2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Creating Stock Trading Rules Using Graph-Based Estimation of Distribution Algorithm

Xianneng Li, Wen He, and Kotaro Hirasawa
Graduate School of Information, Production and Systems, Waseda University, Japan
Email: {sennou@asagi., hewen@toki., and hirasawa@}waseda.jp

***Abstract*—Though there are numerous approaches developed currently, exploring the practical applications of estimation of distribution algorithm (EDA) has been reported to be one of the most important challenges in this field. This paper is dedicated to extend EDA to solve one of the most active research problems – stock trading, which has been rarely revealed in the EDA literature. A recent proposed graph-based EDA called reinforced probabilistic model building genetic network programming (RPMBGNP) is investigated to create stock trading rules. With its distinguished directed graph-based individual structure and the reinforcement learning-based probabilistic modeling, we demonstrate the effectiveness of RPMBGNP for the stock trading task through real-market stock data, where much higher profits are obtained than traditional non-EDA models.**

I. Introduction

In recent years, there has been a significant development of Estimation of Distribution Algorithm (EDA) [1], [2], [3], [4]. Different from the traditional evolutionary computation (EC) techniques which simulate the biological evolution for generating the new population stochastically, EDA focuses on building a probabilistic model from the perspective of machine learning (ML). The probabilistic model estimates the probability distribution of the current population, and is used to sample the new population. Currently, it has been demonstrated that EDA is capable of realizing significant speed-up of evolution efficiency comparing with traditional EC techniques when applying to solve many problems, i.e., function optimization [5], [6], bioinformatics [7], [8], multiobjective optimization [9], [10], scheduling [11], dynamic problems [12], continuous optimization [13], [14], program generation [15], [16], [17], [18], [19], information retrieval [20], etc.

Recently, a new type of EDA techniques called probabilistic model building genetic network programming (PMBGNP) [21], [22], [23] has been developed. Different from the existing EDA techniques which apply the concept of probabilistic modeling into bit-string structure based GA [24] and tree structure based GP [25], PMBGNP is dedicated to estimate the probability distribution of a more complex solution structure – graph structure. Accordingly, it can be viewed as a graph-based EDA. PMBGNP applies a recent proposed EC technique, i.e., genetic network programming (GNP) [26], [27], [28], as the base model to construct its individuals. Following the research directions of EDA, the univariate [22], pairwise [29] and continuous PMBGNP [30] were proposed previously, which have also been studied in both theory [31] and applications [32], [33]. Different from the conventional EDA techniques, PMBGNP is dedicated to solve a different sort of problems, that is, the intelligent agent control. In the previous research, it has been successfully applied to the benchmark problems [23] of the intelligent agent control, as well as the real mobile robot control [34], [35], where its superiority has been demonstrated in comparison with the traditional state-of-the-art techniques.

Beyond the much success of EDA in the past studies, it has been reported and accepted that one of its most important challenges is to explore its applications to solve wider range of problems [36], especially its real-world practical usage. Particularly, there have been numerous studies in applying the soft computing techniques to the stock trading task, such as GA [37], GP [38], [39], GNP [40], neural network (NN) [41], etc., which soon becomes one of the most active research problems. However, this problem has been rarely addressed in the EDA literature, where a gap remains to be filled.

In this paper, we are dedicating to develop an EDA-based stock trading model using PMBGNP. Particularly, an advanced version of PMBGNP called Reinforced PMBGNP (RPMBGNP) [23], [42] is utilized as the base technique to efficiently evolve the stock trading rules. RPMBGNP developed a reinforcement learning (RL)-based probabilistic modeling approach to estimate the probability distribution of its directed graphs, which showed efficient evolution ability for generating compact programs with decision-making rules.

The fundamental basis of the proposed model arises from that when developing an intelligent trading system, a large number of technical indices should be considered to track the movement of the real stock market which sometimes causes the dramatic expansion of the search space. However, due to its efficient scalability and evolution ability under large search space, EDA may provide significant improvement of evolution efficiency when comparing with standard EC techniques.

In the next section, the basic framework of the stock trading task is described. Section III introduces the proposed stock trading model based on RPMBGNP. The experimental studies using the real stock data are carried out in section IV. Finally, the conclusions are drawn in section V.

II. Basic Framework of Stock Trading

Generally, the stock trading task has been focused on in two different aspects. One is the fundamental analysis, which focuses on analyzing the stock prices based on the overall state of macroeconomic markets. Another is the technical analysis which assumes that any factor that truly influences the market

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

will immediately show up in the movement of the stock prices. Though there are some theories addressing that no investor can achieve more than average trading advantages based on the historical information, i.e., the efficient-market hypothesis [43], it has been empirically verified that analyzing a large amount of historical data based on the technical indices can relatively capture the movement of the market [44], [45]. In this paper, we are focusing on the technical analysis to determine the timing of buying/selling signals of the stocks by creating trading rules using RPMBGNP.

A. Technical Indices

The foundation of the technical analysis arises from the technical indices, each of which is *a series of data points that are derived by applying a mathematical equation to the price data of the real market*. The public price data of the stock market generally includes the following 5 types, that is, each day's open price, close price, high price, low price and the volume of trades. The technical indices are capable of offering different perspectives from which the underlying movement of prices is analyzed. Among many available choices, this paper applies 9 major technical indices for the stock trading task based on the empirical experience.

1) Rate of deviation (BIAS): BIAS is the rate of deviation from the average price over a period of time, which can estimate the possible correction or rebound of the price volatility. It can be calculated by

$$BIAS(n) = \Big(p - MA(n)\Big)\Big/MA(n), \tag{1}$$

where p is today's close price, and $MA(n)$ is the moving average of the last n days computed by $MA(n) = (p + p_{-1} + ... + p_{-(n-1)})/n$, and p_{-i} is the close price of i days before.

2) Relative strength index (RSI): RSI focuses on capturing how strongly the stock is moving towards its current direction

$$RSI(n) = \frac{\sum_{j=0}^{n-1} p^{'}_{-j}}{\sum_{i=0}^{n-1} \Big|p_{-i} - p_{-(i+1)}\Big|}, \tag{2}$$

where

$$p^{'}_{-j} = \begin{cases} (p_{-j} - p_{-(j+1)}) & p_{-j} > p_{-(j+1)} \\ 0 & \text{otherwise} \end{cases}$$

3) Rate of change (ROC): ROC measures the speed at which the stock price is changing

$$ROC(n) = p/p_{-n}. \tag{3}$$

4) Volume ratio (VR): VR is a powerful metric to identify whether or not the stock is under accumulation

$$VR(n) = \sum_{j=0}^{n-1} v^{'}_{-j} \Big/ \sum_{i=0}^{n-1} v_{-i}, \tag{4}$$

where v_{-i} is the volume of i days before, and

$$v^{'}_{-j} = \begin{cases} v_{-j} & p_{-j} > p_{-(j+1)} \\ v_{-j}/2 & p_{-j} = p_{-(j+1)} \\ 0 & p_{-j} < p_{-(j+1)} \end{cases}$$

5) Rank correlation index (RCI): RCI focuses on the ranking and date of the stock prices to determine whether it is low or high. Let d_{-i} denotes the differences between the rank of the stock price of i days before and its date over a period of time, we can compute

$$RCI(n) = 1 - \frac{6\sum_{i=0}^{n-1} d^2_{-i}}{n(n^2-1)}. \tag{5}$$

6) Stochastic: Stochastic shows the location of each day's close price relative to the high-low range over several periods

$$Stochastic(n) = \Big(K_0(n) + K_{-1}(n) + K_{-2}(n)\Big)\Big/3, \tag{6}$$

where $K_{-i}(n) = (p_{-i} - LOW)/(HIGH - LOW)$. LOW and $HIGH$ are the lowest and highest price of period from i days before to $i + n$ days before.

7) Golden/Dead cross: It uses two MAs of the short-term and long-term periods to indicate the significant signals of buying and selling.

8) Moving average convergence and divergence (MACD): MACD is one of the most famous indices to spot the changes in the strength, direction and momentum of the price trend.

9) Candlestick chart: Candlestick chart has been widely used to indicate the fluctuation of the stock prices, which is utilized by the candles. 8 candlestick chart patterns introduced in [46] are utilized in this paper, which have shown attractive performance to predict the movement of the stock prices.

B. Problem Formulation

The objective of the stock trading task is to *maximize* the profits by a given initial fund.

The input can be represented by the above described technical indices, while the output is the buying or selling action.

The target of this paper is to develop an intelligent trading model to recommend the most remunerative action for an perceived market situation. In other words, we are dedicating to provide the appropriate timing of buying and selling signals, where the recent advanced EDA named RPMBGNP is applied to construct and evolve the model.

III. RPMBGNP-Based Stock Trading Model

Inspired by original GNP, RPMBGNP represents the candidate stock trading models by its individuals with directed graph structures, which are subject to evolution to find the optimal ones to maximize the profits.

A. Individual Structure

One of the main features of RPMBGNP different from traditional EDA techniques is its individual structures, where a distinguished directed graph of GNP is utilized.

The directed graph ensures strong expression ability to allow RPMBGNP to efficiently model the complex systems. The directed graph is composed of two kinds of nodes

- *Judgment nodes*: simulate the sensory functions to judge the environment.

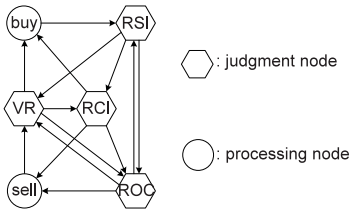


Fig. 1: Directed graph structure

TABLE I: Judgment functions for stock trading

Function	Symbol	#. branches	Contents of branches
J_1–J_3	BIAS with $n = 5, 13, 26$	5	$[-1, -0.1]$, $(-0.1, -0.05]$, $(-0.05, 0.05]$, $(0.05, 0.1]$, $(0.1, \infty)$
J_4–J_6	RSI with $n = 5, 13, 26$	3	$[0, 0.2]$, $(0.2, 0.8]$, $(0.8, 1]$
J_7–J_9	ROC with $n = 5, 13, 26$	3	$[0, 0.9]$, $(0.9, 1.1]$, $(1.1, \infty)$
J_{10}–J_{12}	VR with $n = 5, 13, 26$	3	$[0, 0.3]$, $(0.3, 0.7]$, $(0.7, 1]$
J_{13}–J_{15}	RCI with $n = 5, 13, 26$	3	$[-1, -0.7]$, $(-0.7, 0.7]$, $(0.7, 1]$
J_{16}–J_{18}	Stochastic with $n = 5, 13, 26$	3	$[0, 0.3]$, $(0.3, 0.7]$, $(0.7, 1]$
J_{19}	Golden/Dead cross	3	golden cross, dead cross, the others
J_{20}	MACD	3	golden cross, dead cross, the others
J_{21}	Candlestick chart	8	8 candlestick chart patterns

- *Processing nodes*: include the processing functions for the action determination.

Each node i includes a set of parameters: $NT_i \in \{0, 1\}$ is the node type (0 for judgment nodes and 1 for processing nodes); NF_i denotes the node function read from a problem specific *LIBRARY*; $C_{i1}, C_{i2}, ...$ denotes the connected node from branch 1, 2,... of node i; TD_i and td_i represents the time delays spent on executing node i or transiting its branches, respectively.

Each judgment node consists of multiple branches, each of which corresponds to a specific input perceived from the environment, while processing node does not consist of conditional branches, in other words, with only C_{i1}.

TD and td play the roles to model the agents with human like brain that needs the time for thinking. With time delays, the program can count the needed time units of executing its directed graph. In this paper, $TD = 5$ or 1 for judgment or processing nodes, respectively, and $td = 0$. A trading day ends when 5 or more time units reached. In other words, the programs can execute "5 judgment nodes" or "less than 5 judgment nodes with 1 processing node" per each trading day.

B. LIBRARY (Judgment/Processing Functions)

In order to develop the stock trading model, the *LIBRARY* is prepared to define the judgment and processing functions based on the descriptions of section II.

The judgment functions are defined based on the listed 9 technical indices. However, as for the first 6 technical indices, n is set to 5, 13 and 26 to track the relatively short-, medium-, and long-term movements of the stock prices, respectively. The number of branches in judgment nodes are determined based on the corresponding judgment functions and their perceived sensory results from the stock prices. Accordingly, they are total 21 judgment functions as listed in Table I.

Two processing functions are existed in the model, corresponding to the actions of buying or selling, respectively.

C. Performance Component (Workflow of the Directed Graph)

Each stock trading model, represented by a directed graph, works as an intelligent agent. Initially, the model starts its execution from a predefined start judgment node, and the transition of the model continues depending on the judgment results and node connections.

When a judgment node is executed, it applies its judgment function to judge the current situation of the stock prices, and transits to another node by selecting a branch based on the perceived value matching with Table I. After a number of judgment nodes, a processing node might be executed, leading to the trading behavior. During the transition of the directed graph, the time units are incrementally counted based on the time delays, which simulates the stream of trading days. The model is terminated if the final trading day arrives. Afterwards, the fitness value can be evaluated.

1) Importance Index: Since different technical indices have different ranges, Importance Index (IMX) [47] is used to transform the returned judgment values of the technical indices to a normalized interval $[-1, 1]$. The IMX functions for each technical index used in this paper is based on the ones described in [47].

2) Creating trading actions: The processing nodes consist of the candidate trading actions, i.e., buying and selling, which are executed based on the node transitions and judgment results.

Each processing node i consists of a processing function NF_i and a numerical variable v_i. NF_i could be 1 (buying) or 2 (selling), while v_i denotes the trading signal of node i.

When processing node i is transited, the strategy of stock trading of our model is carried out as follows:

1) First, we calculate the average IMX value A of the judgment nodes transited from the previous processing node to the current processing node i (the set of these judgment nodes is denoted by I). That is,
$$A = \sum_{j \in I} IMX(j) \Big/ |I|. \tag{7}$$
2) Second, if $NF_i = 1$ (buying),
   - if $A \geq v_i$, buy stocks as much as possible.
   - otherwise, just do nothing.
3) But, if $NF_i = 2$ (selling),
   - if $A < v_i$, sell stocks as much as possible.
   - otherwise, just do nothing.

D. Evolution Space

Each directed graph is encoded with set N_j of judgment nodes and set N_p of processing nodes, where the nodes can be connected arbitrarily to model complicated combinations of judgment and processing. In other words, it searches the optimal combination of technical indices to construct the stock trading model generated by a sequence of decision-making trading rules.

In this problem, evolution bias is applied to evolve the node connections and trading signal of each processing node. In other words, $C_{i1}, C_{i2}, ...$ for each node $i \in N_j \cup N_p$ and v_j for each processing node $j \in N_p$ are subject to evolution to determine the optimal stock trading models. Accordingly,

Definition 1: The number of *search variables* Θ of GNP is defined by

$$\Theta = \mathcal{B} + \mathcal{V}. \tag{8}$$

where $\mathcal{B}$ is the total number of branches of the directed graph, and $\mathcal{V}$ is the number of processing nodes, that is, $|N_p|$.

It is clear that the $\mathcal{B}$ variables are discrete ones, while $\mathcal{V}$ variables are in a continuous range, which result in a mixed optimization problem.

E. Evolving $\mathcal{B}$ by RPMBGNP

In order to evolve the $\mathcal{B}$ discrete variables, a reinforcement learning (RL)-based EDA proposed in [23] is utilized. The proposed technique, named Reinforced PMBGNP (RPMBGNP), is dedicated to learn knowledge/experience, i.e., Q values, using RL [48] to measure the quality of node connections of the directed graph, where the Q values are used to construct the probabilistic model of EDA to estimate a precise model with implicit multivariate interactions.

In RPMBGNP, each candidate stock trading model, i.e., a directed graph, is viewed as a policy of RL, where the node transitions are defined as an *episode*. We define a branch of the directed graph as the *state* of RL, where the *action* of RL is denoted as the selection of a node to be connected by a given branch (state). Accordingly, each model can be factorized by a sequence of state-action pairs (node connections)

$$(S, A) = \{(s_1, a_1), (s_2, a_2), ..., (s_L, a_L)\}, \tag{9}$$

where L is the length of the episode.

Each candidate model n can be substituted to the above form $(S, A)_n$ of state-action pairs after its execution. An on-policy RL technique called Sarsa Learning (Sarsa) is applied to update the Q values of state-action pairs. Suppose the current state and action at time step t are $b(i)$ and j, respectively, which means the current state-action pair (s_t, a_t) is formed by node connection $(b(i), j)$ [1], and the state-action pair in the next time step is $(s_{t+1}, a_{t+1}) = (b(j), k)$. Then, the Q value of $(b(i), j)$ is updated by:

$$Q(b(i), j) \leftarrow Q(b(i), j) + \alpha \left[r_j + \gamma Q(b(j), k) - Q(b(i), j) \right], \tag{10}$$

where,

α, γ: learning rate and discount factor.

r_j: reward of choosing node j at branch $b(i)$ of node i, and,

1) In the case of judgment nodes, $r_j = 0$.
2) In the case of processing nodes, r_j is given after processing node j. Particularly, r_j is given when a selling action is carried out,

$$r_j = \text{selling price} - \text{buying price}, \tag{11}$$

which shows the profit of one buying-selling pair. If the selling action is not executed, $r_j = 0$.

[1] $b(i)$ denotes branch $b(i)$ of node i, and j is $C_{iB(i)}$ based on the description of individual structure.

Algorithm 1 Updating of Q values

1: $n \leftarrow 1$;
2: **for** $n \leq N$ **do**
3: execute candidate model n, and obtain the episode (sequence of state-action pairs) $(S, A)_n$;
4: update the corresponding Q values of $(S, A)_n$ using Eq. (10);
5: $n \leftarrow n + 1$;
6: **end for**

The procedure of updating Q values in each generation is shown in Algorithm 1. The Q values are updated based on the execution of the best individuals (truncation selection with size N). As a result, we can collect the experience of the promising individuals into one Q table which consists of all $Q(S, A)$ values. It is expected that the good state-action pairs will be rewarded with high Q values, and vice-versa. Accordingly, the quality of node connections can be explicitly captured, which is used to estimate the probabilistic model of RPMBGNP.

RPMBGNP constructs its probabilistic model by a set of connection probabilities. For example, $P(b(i), j)$ denotes the probability that branch $b(i)$ is connected to node j. As a result, the optimal directed graph structures can be sampled if we learn the accurate connection probabilities. Each $P(b(i), j)$ is calculated by

$$P(b(i), j) = \frac{\exp\left(\frac{Q(b(i),j)}{T}\right)}{Z(b(i))}. \tag{12}$$

The normalization function $Z(b(i))$ is calculated by

$$Z(b(i)) = \sum_{j' \in N_j \cup N_p} \exp\left(\frac{Q(b(i), j')}{T}\right), \tag{13}$$

and temperature parameter T is obtained by Eq. (14).

$$T = \frac{\tau}{t+1}, \tag{14}$$

where τ is a coefficient, and t is the current generation number.

F. Evolving $\mathcal{V}$ by Continuous RPMBGNP

As for the trading signals $\mathcal{V}$ in the processing nodes with continuous intervals, the continuous RPMBGNP proposed in [30] is applied. In our model, Gaussian distribution is used to model the continuous variables, where each trading signal v_i is represented by a unidimensional Gaussian distribution $\mathcal{N}(\mu_i, \sigma_i^2)$. The mean value μ_i and standard deviation σ_i^2 of each v_i is subject to evolution. Continuous RPMBGNP develops a RL-based probabilistic modeling approach using Actor-Critic (AC) technique [48] combined with gradient search to evolve the probability density function (pdf) of each v_i.

In order to update the pdf of each trading signal v_i, the following steps are carried out sequentially.

Step 1: Calculate gradients

Lemma 1: Given a sampled value x_i of trading signal v_i, the gradients of μ_i and σ_i are calculated by

$$\nabla(\mu_i; x_i) = x_i - \mu_i, \tag{15}$$

$$\nabla(\sigma_i; x_i) = \frac{(x_i - \mu_i)^2}{\sigma_i^2} - 1. \tag{16}$$

The proof of *Lemma 1* can be found in [30]. The gradients denote the updating directions of μ_i and σ_i given a sampled value x_i.

Step 2: Calculate scalar reinforcement signal

After obtaining the gradient of each parameter, continuous RPMBGNP incorporates a RL technique named AC to guide the updating direction.

Similarly to the concept of RL in discrete RPMBGNP, the quality of sampled value x_i from a given variable v_i can be measured by RL. In this case, we define a trading signal of a processing node, i.e., v_i, as the *state*, where the *action* is represented by a sampled value from the pdf of v_i, i.e., x_i. Accordingly, we are capable of incorporating AC efficiently, where the Gaussian distribution is known as the *actor* since it is used to select actions, and the state-value function can be formulated as the *critic*.

Each time processing node i is executed, AC evaluates the next new processing node j to determine whether the current sampled value is better or worse than expected. The evaluation is formulated by a temporal-difference (TD)-error δ as

$$\delta_i = r_i + \gamma_{AC} V(j) - V(i), \tag{17}$$

where,
$V(i)$: state value function of processing node i.
γ_{AC}: discounted factor of AC.

After obtaining the TD-error, it is sent back to update the state-value function V by:

$$V(i) \leftarrow V(i) + \alpha_{AC} \delta_i, \tag{18}$$

where α_{AC} is the learning rate of AC.

If δ_i is positive, it suggests that the tendency to sample x_i should be strengthened, and vice-versa. Accordingly, δ_i can be used to formulate a scalar reinforcement signal θ_i to indicate whether the updating direction of this sampled value should be strengthened or weakened.

$$\theta_i = \begin{cases} -1, & \text{for } \delta_i < 0 \\ 0, & \text{for } \delta_i = 0 \\ 1, & \text{for } \delta_i > 0 \end{cases} \tag{19}$$

Step 3: Update the Gaussian distribution

Combining the scalar reinforcement signal with the gradient search, continuous RPMBGNP updates the pdf of Gaussian distribution for each variable as follows

$$\mu_i \leftarrow \mu_i + \alpha_\mu \nabla(\mu_i; x_i) \theta_i, \tag{20}$$

$$\sigma_i \leftarrow \sigma_i + \alpha_\sigma \nabla(\sigma_i; x_i) \theta_i, \tag{21}$$

where α_μ and α_σ are the learning rates.

IV. Experimental Study

The proposed RPMBGNP-based stock trading model is applied to 9 real stock data obtained from the Tokyo Stock Exchange market. The stock data is divided into two periods: *training* and *testing*. The training data are used to evolve the stock trading models represented by the directed graph of RPMBGNP, while the best model obtained in the last generation of the training period is applied to the testing data. The experimental data used in this paper are as follows

- *Training data*: January 4, 2001 – December 30, 2003
- *Testing data*: January 5, 2004 – December 30, 2004

Initial funds with 5 million Japanese Yen are provided to the trader, where the final fitness value for each candidate model is the total profits obtained during the training period. All the results reported in this paper are the average values of 30 independent experiments.

A. Parameter Settings

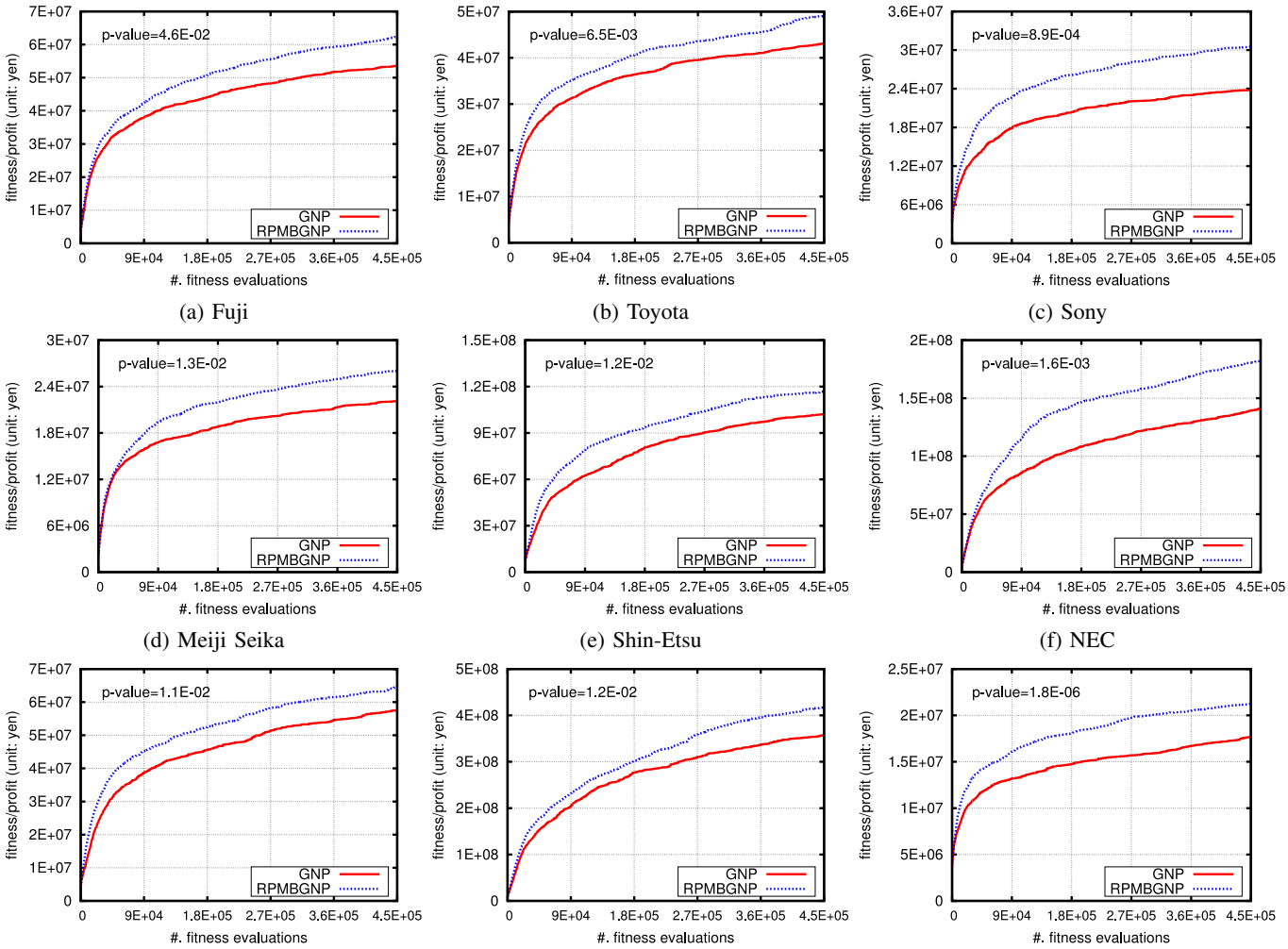
Since our purpose is to demonstrate the effectiveness of EDA for the stock trading task, the comparisons are mainly carried in comparison with the EC techniques with standard crossover and mutation, i.e., standard GNP [40].

The directed graph of RPMBGNP is defined as follows: $|N_j| = 42$ (2 nodes for each judgment function listed in Table I), and $N_p = 10$ (5 nodes for each processing function, i.e., buying/selling).

In standard GNP, uniform crossover and mutation with their rates of $p_c = 0.1$ and $p_m = 0.02$ are performed. On the other hand, RPMBGNP neglects these parameters of genetic operators. Instead, the parameters of RPMBGNP for its probabilistic modeling are defined as follows: $\alpha = 0.2$ and $\gamma = 0.9$ for Sarsa; $\tau = 1000$; $\alpha_{AC} = 0.1$ and $\gamma_{AC} = 0.9$ for AC; $\alpha_\mu = 0.05$ and $\alpha_\gamma = 0.05$ for Gaussian distribution. The population size of GNP is set at 300 with 1 elite individual, 120 crossover individuals and 179 mutation individuals. On the other hand, RPMBGNP requires a relatively large population size in order to avoid its local convergence [23]. Therefore, the population size of 1500 is used in this paper. Truncation selection with size $N = 750$ is used. The terminal condition for each experiment is based on the maximal number of fitness evaluations, where $450,000$ is used in this paper.

B. Experimental Results in the Training Period

1) Evolution efficiency: The fitness curves of RPMBGNP for the training period are plotted in Fig. 2 for the comparison with standard GNP, which are represented by the profits obtained by the best individuals at each generation. It is found that the directed graph is suitable for constructing the stock trading model, where high profits can be obtained through evolution. On the other hand, the results of 9 studied stocks confirm that RPMBGNP can achieve the significant improvement of the evolution efficiency to find better stock trading models than standard GNP with crossover and mutation.



(a) Fuji (b) Toyota (c) Sony

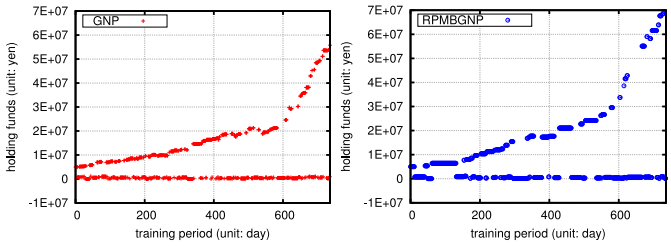
(d) Meiji Seika (e) Shin-Etsu (f) NEC

(g) Mitsubishi (h) KDDI (i) Tokyo Gas

Fig. 2: Comparison of fitness (profit) curves over 9 selected stocks in the training period

2) Statistical analysis: Two-tailed/paired t-test is applied to analyze the fitness values obtained by GNP and RPMBGNP in the training period. We calculate the p-values between the two algorithms for the 9 stocks, where the p-value smaller than 0.05 indicates a statistically significant difference between them. Fig. 2 includes the t-test result calculated for each stock. It is confirmed that RPMBGNP outperforms the standard GNP in all experiments with statistical meaning.

3) Change of holding funds: In a micro-level aspect, Fig. 3 shows the typical trajectories of the change of the holding funds during the training period of Mitsubishi stock. The dramatic decrease of holding funds indicates a buying action, while the significant increase of holding funds is caused by the selling action. It is shown that RPMBGNP has different buying/selling *timing* when comparing with GNP, which causes the different profits obtained in the final trading day.



(a) GNP (b) RPMBGNP

Fig. 3: Typical trajectories of holding funds on each trading day in the training period (Mitsubishi stock)

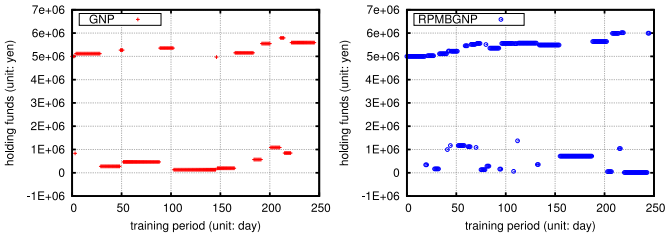
C. Experimental Results in the Testing Period

1) Amount of profits: After the training period, the best model obtained in the last generation of the evolution is applied to make trading in the testing period. Table II reports the profits obtained by different algorithms[2]. The testing profits demonstrate that RPMBGNP can evolve better stock trading

[2]GA and Buy&Hold are selected as the baselines to evaluate this study. GA denotes the method that applies GA to evolve the continuous variables $\mathcal{V}$, while Buy&Hold is a strategy that buys as much stocks as possible on the start day and sells them all on the last day.

TABLE II: Profits (unit: yen) obtained in the testing period (profit rate [unit:%])

Stock	RPMBGNP	GNP	GA	Buy&Hold
Fuji	−122,000 (−2.4)	−140,000 (−2.8)	173,000 (3.5)	−189,000 (−3.8)
Toyota	407,000 (8.1)	640,000 (12.8)	507,000 (10.1)	520,000 (10.4)
Sony	138,000 (2.8)	−36,000 (−0.7)	112,000 (2.2)	150,000 (3.0)
Meiji Seika	180,000 (3.6)	287,000 (5.7)	−185,000 (−3.7)	451,000 (9.0)
Shin-Etsu	156,000 (3.1)	54,000 (1.1)	−539,400 (−10.8)	−264,000 (−5.3)
NEC	−164,000 (−3.3)	−292,000 (−5.8)	−531,000 (−10.6)	−1,026,000 (−20.5)
Mitsubishi	832,000 (16.6)	551,000 (11.1)	174,000 (3.5)	664,000 (13.3)
KDDI	−53,000 (−1.1)	−103,000 (−2.1)	−273,000 (−5.5)	−576,000 (−11.5)
Tokyo Gas	362,000 (7.2)	225,000 (4.5)	729,850 (14.6)	372,000 (7.4)
Total profits	1,736,000	1,186,000	167,450	102,000



(a) GNP (b) RPMBGNP

Fig. 4: Change of holding funds on each trading day in the testing period (Mitsubishi stock)

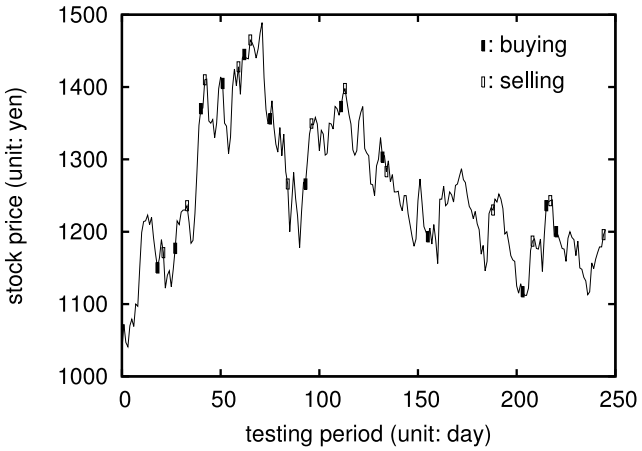


Fig. 5: Timing of buying/selling recommended by RPMBGNP in the testing period (Mitsubishi stock)

models than the other compared algorithms. Comparing with standard GNP, RPMBGNP can obtain higher profits, though in some stocks, i.e., Toyota and Meiji Seika, worse results are obtained which might be due to the overfitting problem.

2) Change of holding funds: Fig. 4 shows the change of the holding funds during the testing period of Mitsubishi stock, which shows that RPMBGNP has different buying/selling *timing* when comparing with GNP to obtain higher profits in the final trading day.

3) Timing of buying/selling: Fig. 5 shows a typical example of the buying/selling timing made by RPMBGNP which is plotted in the stock movement of the testing period. It is found that in most cases, RPMBGNP can buy stocks at lower prices and make selling decision at relatively higher prices.

Considering the results of the training and testing periods as a whole, we demonstrate the effectiveness of the proposed RPMBGNP-based stock trading model comparing with the other non-EDA models, which clarifies the superiority of the EDA-based probabilistic modeling in this problem.

V. Conclusions and Future Work

In this paper, an EDA-based stock trading model has been proposed, where a recent proposed algorithm called RPMBGNP is applied. With its directed graph structure and RL-based probabilistic modeling, RPMBGNP shows attractive expression and evolution ability to create stock trading rules. The experimental results clarified that the proposed model can obtain better evolution efficiency and higher profits when comparing with the traditional non-EDA models. In the future, further improvements of the research using variable size GNP [49], [50] and rule-based systems [51], [52] will be studied. Self-adaptive method to reduce the parameters is also an another direction.

References

[1] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive leaning," Carnegie Mellon University, Tech. Report CMU-CS-94-163, 1994.

[2] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions i. binary parameters," in *Proc. of the Conf. on Parallel Problem Solving from Nature*, 1996, pp. 178–187.

[3] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.

[4] Q. Zhang and H. Mühlenbein, "On the convergence of a class of estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 127–136, 2004.

[5] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "Linkage problem, distribution estimation, and Bayesian networks," *Evol. Comput.*, vol. 8, no. 3, pp. 311–341, 2002.

[6] A. Brownlee, J. McCall, and Q. Zhang, "Fitness modeling with markov networks," *IEEE Trans. Evol. Comput.*, vol. 17, no. 6, pp. 862–879, 2013.

[7] T. K. Paul and H. Iba, "Identification of informative genes for molecular classification using probabilistic model building genetic algorithm," in *Proc. of the Genetic and Evol. Comput. Conf.*, 2004, pp. 414–425.

[8] R. Santana, P. Larrañaga, and J. A. Lozano, "Protein folding in simplified models with estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 12, no. 4, pp. 418–438, 2008.

[9] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 41–63, 2008.

[10] A. Zhou, Q. Zhang, and Y. Jin, "Approximating the set of pareto-optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1167–1189, 2009.

[11] J. Ceberio, E. Irurozqui, A. Mendiburu, and J. Lozano, "A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem," *IEEE Trans. Evol. Comput.*, 2013, (early access).

[12] S. Yang and X. Yao, "Population-based incremental learning with associative memory for dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 12, no. 5, pp. 542–561, 2008.

[13] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña, "Optimization by learning and simulation of Bayesian and gaussian networks," Intelligent Systems Group, Dept. of Comput. Sci. and Artif. Intell., University of the Basque Country, Tech. Report EHU-KZAA-IK-4-99, 1999.

[14] P. A. N. Bosman and D. Thierens, "Numerical optimization with real-valued estimation-of-distribution algorithms," in *Scalable Optimization via Probabilistic Modeling*, M. Pelikan, K. Sastry, and E. Cantú-Paz, Eds. Berlin, Germany: Springer-Verlag, 2006, ch. 5, pp. 91–120.

[15] R. P. Salustowicz and J. Schmidhuber, "Probabilistic incremental program evolution," *Evol. Comput.*, vol. 5, no. 2, pp. 123–141, 1997.

[16] K. Yanai and H. Iba, "Estimation of distribution programming based on Bayesian network," in *Proc. of the IEEE Congress on Evol. Comput.*, 2003, pp. 1618–1625.

[17] Y. Shan, R. I. McKay, R. Baxter, H. Abbass, D. Essam, and N. X. Hoai, "Grammar model-based program evolution," in *Proc. of the IEEE Congress on Evol. Comput.*, 2004, pp. 478–485.

[18] Y. Hasegawa and H. Iba, "A Bayesian network approach to program generation," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 750–764, 2008.

[19] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming*. Published via http://lulu.com/ and freely available at http://www.gp-field-guide.org.uk/, 2008, (With contributions by J. R. Koza).

[20] H. Sato, D. Bollegala, Y. Hasegawa, and H. Iba, "Learning non-linear ranking functions for web search using probabilistic model building GP," in *Proc. of the IEEE Congress on Evol. Comput.*, 2013, pp. 3371–3378.

[21] X. Li, S. Mabu, H. Zhou, K. Shimada, and K. Hirasawa, "Genetic network programming with estimation of distribution algorithms and its application to association rule mining for traffic prediction," in *Proc. of the ICCAS-SICE*, 2009, pp. 3457–3462.

[22] ——, "Genetic network programming with estimation of distribution algorithms for class association rule mining in traffic prediction," in *Proc. of the IEEE Congress on Evol. Comput.*, 2010, pp. 2673–2680.

[23] X. Li, S. Mabu, and K. Hirasawa, "A novel graph-based estimation of distribution algorithm and its extension using reinforcement learning," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 98–113, 2014.

[24] M. Pelikan, D. E. Goldberg, and F. G. Lobo, "A survey of optimization by building and using probabilistic models," *Computational Optimization and Applications*, vol. 21, no. 1, pp. 5–20, 2002.

[25] Y. Shan, R. I. McKay, D. Essam, and H. A. Abbass, "A survey of probabilistic model building genetic programming," in *Scalable Optimization via Probabilistic Modeling*, M. Pelikan, K. Sastry, and E. Cantú-Paz, Eds. Berlin, Germany: Springer-Verlag, 2006, ch. 6, pp. 121–154.

[26] K. Hirasawa, M. Okubo, H. Katagiri, J. Hu, and J. Murata, "Comparison between genetic network programming (GNP) and genetic programming (GP)," in *Proc. of the IEEE Congress on Evol. Comput.*, 2001, pp. 1276–1282.

[27] S. Mabu, K. Hirasawa, and J. Hu, "A graph-based evolutionary algorithm: Genetic network programming (GNP) and its extension using reinforcement learning," *Evol. Comput.*, vol. 15, no. 3, pp. 369–398, 2007.

[28] X. Li, W. He, and K. Hirasawa, "Genetic network programming with simplified genetic operators," in *Neural Information Processing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 8227, pp. 51–58.

[29] X. Li, B. Li, S. Mabu, and K. Hirasawa, "A novel estimation of distribution algorithm using graph-based chromosome representation and reinforcement learning," in *Proc. of the IEEE Congress on Evol. Comput.*, 2011, pp. 37–44.

[30] X. Li, S. Mabu, H. Zhou, K. Shimada, and K. Hirasawa, "A continuous estimation of distribution algorithm by evolving graph structures using reinforcement learning," in *Proc. of the IEEE Congress on Evol. Comput.*, 2012, pp. 2097–2104.

[31] X. Li, S. Mabu, and K. Hirasawa, "Towards the maintenance of population diversity: A hybrid probabilistic model building genetic network programming," *Trans. of the Japanese Society for Evol. Comput.*, vol. 1, no. 1, pp. 89–101, 2010.

[32] ——, "Use of infeasible individuals in probabilistic model building genetic network programming," in *Proc. of the Genetic and Evol. Comput. Conf.*, 2011, pp. 601–608.

[33] X. Li, S. Mabu, H. Zhou, K. Shimada, and K. Hirasawa, "Analysis of various interestingness measures in class association rule mining," *SICE Journal of Control, Measurement, and System Integration*, vol. 4, no. 4, pp. 295–304, 2011.

[34] X. Li, S. Mabu, M. K. Mainali, and K. Hirasawa, "Probabilistic model building genetic network programming using multiple probability vectors," in *Proc. of the IEEE Region 10 Conf. (TENCON)*, 2010, pp. 1398–1403.

[35] X. Li, S. Mabu, and K. Hirasawa, "An extended probabilistic model building genetic network programming using both of good and bad individuals," *IEEJ Trans. on Electrical and Electronic Engineering*, vol. 8, no. 4, pp. 339–347, 2013.

[36] R. Santana, P. Larrañaga, and J. A. Lozano, "Research topics in discrete estimation of distribution algorithms based on factorizations," *Memetic Computing*, vol. 1, no. 1, pp. 35–54, 2009.

[37] F. Allen and R. Karjalainen, "Using genetic algorithms to find technical trading rules," *Journal of financial Economics*, vol. 51, no. 2, pp. 245–271, 1999.

[38] H. Iba and T. Sasaki, "Using genetic programming to predict financial data," in *Proc. of the IEEE Congress on Evol. Comput.*, 1999, pp. 244–251.

[39] J.-Y. Potvin, P. Soriano, and M. Vallée, "Generating trading rules on the stock markets with genetic programming," *Computers & Operations Research*, vol. 31, no. 7, pp. 1033–1047, 2004.

[40] Y. Izumi, T. Yamaguchi, S. Mabu, K. Hirasawa, and J. Hu, "Trading rules on the stock markets using genetic network programming with candlestick chart," in *Proc. of the IEEE Congress on Evol. Comput.*, 2006, pp. 2362–2367.

[41] K. K. Ang and C. Quek, "Stock trading using rspop: A novel rough set-based neuro-fuzzy approach," *IEEE Trans. Neural Netw.*, vol. 17, no. 5, pp. 1301–1315, 2006.

[42] X. Li, *Study on Probabilistic Model Building Genetic Network Programming*. Waseda University Press, 2013.

[43] E. F. Fama, "The behavior of stock-market prices," *The Journal of Business*, vol. 38, no. 1, pp. 34–105, 1965.

[44] A. Hirabayashi, C. de Castro Aranha, and H. Iba, "Optimization of the trading rule in foreign exchange using genetic algorithm," in *Proc. of the Genetic and Evol. Comput. Conf.*, 2009, pp. 1529–1536.

[45] A. Ghandar, Z. Michalewicz, M. Schmidt, T.-D. To, and R. Zurbrugg, "Computational intelligence for evolving trading rules," *IEEE Trans. Evol. Comput.*, vol. 13, no. 1, pp. 71–86, 2009.

[46] Y. Chen, S. Mabu, K. Shimada, and K. Hirasawa, "A genetic network programming with learning approach for enhanced stock trading model," *Expert Systems with Applications*, vol. 36, no. 10, pp. 12 537–12 546, 2009.

[47] ——, "Trading rules on stock markets using genetic network programming with sarsa learning," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 12, no. 4, pp. 383–392, 2008.

[48] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[49] B. Li, X. Li, S. Mabu, and K. Hirasawa, "Variable size genetic network programming with binomial distribution," in *Proc. of the IEEE Congress on Evol. Comput.*, 2011, pp. 973–980.

[50] ——, "Evolving graph-based chromosome by means of variable size genetic network programming with binomial distribution," *IEEJ Trans. on Electrical and Electronic Engineering*, vol. 8, no. 4, pp. 348–356, 2013.

[51] X. Li and K. Hirasawa, "Extended rule-based genetic network programming," in *Proc. of the Proc. of the Genetic and Evol. Comput. Conf. Companion*, 2013, pp. 155–156.

[52] ——, "A learning classifier system based on genetic network programming," in *Proc. of the IEEE Int'l Conf. on Syst., Man, Cybern.*, 2013, pp. 1323–1328.