Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Sciences 176 (2006) 2121–2147
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

Adaptive stock trading with dynamic asset allocation using reinforcement learning

Jangmin O [a,*], Jongwoo Lee [b], Jae Won Lee [c], Byoung-Tak Zhang [a]

[a] *School of Computer Science and Engineering, Seoul National University, San 56-1, Shillim-dong, Kwanak-gu, Seoul 151-742, Republic of Korea*
[b] *Department of Multimedia Science, Sookmyung Women's University, Chongpa-dong, Yongsan-gu, Seoul 140-742, Republic of Korea*
[c] *School of Computer Science and Engineering, Sungshin Women's University, Dongsun-dong, Sungbuk-gu, Seoul 136-742, Republic of Korea*

Received 4 December 2003; received in revised form 11 October 2005; accepted 14 October 2005

Abstract

Stock trading is an important decision-making problem that involves both stock selection and asset management. Though many promising results have been reported for predicting prices, selecting stocks, and managing assets using machine-learning techniques, considering all of them is challenging because of their complexity. In this paper, we present a new stock trading method that incorporates dynamic asset allocation in a reinforcement-learning framework. The proposed asset allocation strategy, called meta policy (MP), is designed to utilize the temporal information from both stock recommendations and the ratio of the stock fund over the asset. Local traders are constructed with pattern-based multiple predictors, and used to decide the purchase money per recommendation. Formulating the MP in the reinforcement learning framework is achieved by a compact design of the environment and the learning agent. Experimental results

* Corresponding author. Tel.: +82 2 880 1847; fax: +82 2 875 2240.
E-mail addresses: jmoh@bi.snu.ac.kr (J. O), bigrain@sookmyung.ac.kr (J. Lee), jwlee@cs.sungshin.ac.kr (J.W. Lee), btzhang@cse.snu.ac.kr (B.-T. Zhang).

0020-0255/$ - see front matter
doi:10.1016/j.ins.2005.10.009

using the Korean stock market show that the proposed MP method outperforms other fixed asset-allocation strategies, and reduces the risks inherent in local traders.

Keywords: Stock trading; Reinforcement learning; Multiple-predictors approach; Asset allocation

1. Introduction

In the stock market, people invest some of their assets in stocks or derivatives via the market, and companies may raise their business funds from the market. An ultimate goal of investors is to make profits, and, at the same time, to satisfy constraints on, or interests in, their investments. During the last few decades, various algorithms have been applied to stock market problems [11]. Some of them formulate the market via mathematical equations, putting heavy assumptions on the market's characteristics to achieve compactness and soundness in their models. However, since the stock market is a complicated compound of capitalism, attempts to model or predict the stock market have not been successful in *consistently* 'beating the market'. This is the famous efficient market hypothesis (EMH), which says that future prices cannot be predicted, since all of the information available is already reflected in the history of past prices [16]. However, if we step back from the word *consistently*, then we can find several empirical results indicating that the market might be *somewhat* predictable [5,6]. Indeed, the last decade has witnessed an abundance of such new approaches to financial analysis, both from academia and from industry.

Machine-learning approaches to stock market modeling can be classified using various criteria. First, considering the target for trading, we might focus on either a single price series such as the S&P 500 index, or all of the shares in the market. Second, considering the purpose of the algorithm, there are some approaches that predict future prices, and others that manage the asset and its portfolios.

Much work has been done on price prediction. State–space models, generalizing from traditional dynamic equation models such as autoregressive integrated moving average (ARIMA), are devoted to single series modeling, and their statistical properties are well established [3,11]. The prediction problem is an aim of supervised learning, which models the relationship between the input and desired-output pair [4]. In particular, there has been much interest in the application of artificial neural networks to predict prices, and this popularity continues mainly because neural networks do not require a parametric system model, and because they are relatively insensitive to unusual data patterns [1,15,23]. Regarding approaches to the selection of stocks, classification algorithms have been used as well as neural networks-based classifiers. Fan et al. used support vector machines (SVMs) as a classifier on stocks in the

Australian stock market, by formulating the problem as a binary classification: it is either a high-return stock or it is not [7]. Kim et al. used a decision-tree method, with the concept of distributional differences used to decide whether patterns of stock-price changes belong to up trends or down trends [12].

Approaches to price prediction and stock selection based on supervised learning do not cover asset management issues. Supervised-learning-based approaches aim to minimize errors between the input and the desired output, without considering the effect of the original decision.

The mechanism of time-delayed learning with cumulative rewards in reinforcement learning makes it natural to consider decision-making [24,26]. Asset management, including risk avoidance and portfolio optimization, has been intensively studied in terms of reinforcement learning. Neuneier formulated the financial market as a simplified artificial market by regarding it as a Markov decision process (MDP) [20]. He also modified the Q-learning method, by adding some preferences to risk-avoiding tendencies [17]. Gao et al. considered a method for portfolio-management scheduling, namely how to switch between two price series within a Q-learning framework [8]. Moody et al. also treated portfolio management in a similar way to that of [8], but they formulated it using direct reinforcement, instead of the conventional Q-learning method [19].

However, reinforcement learning requires the representation of the problem to be simplified. The work of [17,20] makes simple assumptions about a market to make the problem manageable in a reinforcement-learning framework. Some portfolio-management issues are treated in [8,19], but they focus only on how to switch between a few two-price series.

The approaches described above have treated separate problems of stock market analysis in isolation. In this work, by contrast, we aim to solve the stock trading problem by taking into account the sub-problems in a consistent framework, such as [21]. In particular, we are interested in trading the shares of the stock market as well as managing the asset efficiently. We present a new method, incorporating both stock selection and asset management, which is manageable by current reinforcement learning. We build local traders, based on pattern-based multiple predictors, which undertake the stock selection. This has the effect of reducing the whole problem to a smaller decision-making problem, namely an asset management problem, which makes full use of the local traders to maximize the trading performance. To achieve this, we propose a dynamic asset-allocation strategy called meta policy (MP). MP decides the purchase money for a recommendation of each local trader, by exploiting two types of temporal information, namely, the summaries of the recommendations of local traders, and the stock fund ratio over the asset.

It seems natural to formulate the whole of stock trading in terms of reinforcement learning, but this is hindered by the exponentially growing state space needed to describe this complex task, especially the stock-price modeling. Though there has been work on integrating the stock selection by

reinforcement learning with the sophisticated design of the state space, such as [13,14,22], its predictive power is not comparable to conventional supervised learning methods. In our work, optimizing the stock trading problem corresponds to formulating an MP and maximizing its performance in a reinforcement-learning framework. This makes it possible to trade the hundreds of shares of the Korean stock market in an efficient way. The superiority of the proposed method with respect to trading performance is supported by experimental results using the Korean stock market.

The remainder of this paper is organized as follows: Section 2 summarizes the local traders for the stock selection, Section 3 describes the need for MP, Section 4 develops a reinforcement-learning framework for stock trading with MP, Sections 5 and 6 explain the experimental environment and discuss results, and Section 7 draws conclusions.

2. Local traders

For stock selection, it is not easy to use traditional dynamic equations to model all the shares in the market. In an ordinary stock market, there are many stocks to trade. For example, there are more than 1600 shares in the Korean stock market. Thus we choose, as an alternative, to use pattern-based supervised learning so that we can predict and select candidates from among the hundreds of stocks in the market. Rather than using an all-in-one predictor, we have focused on four meaningful patterns. For each pattern, we have built a local trader that consists of a predictor and a local policy. Predictors estimate the future price of shares, and local policies specify how to use the predicted values given by its predictor.

$$
\begin{aligned}
D_{\text{bear}} &= \{(\mathbf{x}_{s,t}, o_{s,t}) | MA5^s_t < MA10^s_t < MA20^s_t, s \in \mathscr{S}, t \in \mathscr{T}\} \\
D_{\text{bull}} &= \{(\mathbf{x}_{s,t}, o_{s,t}) | MA5^s_t > MA10^s_t > MA20^s_t, s \in \mathscr{S}, t \in \mathscr{T}\} \\
D_{\text{GC}} &= \{(\mathbf{x}_{s,t}, o_{s,t}) | ([MA5^s_{t-1} < MA10^s_{t-1} \&\& MA5^s_t > MA10^s_t] \\
&\quad \&\& [Grad5^s_t > 0 \&\& Grad10^s_t > 0]) \\
&\quad \| ([MA10^s_{t-1} < MA20^s_{t-1} \&\& MA10^s_t > MA20^s_t] \\
&\quad \&\& [Grad10^s_t > 0 \&\& Grad20^s_t > 0]) \\
&\quad \| ([MA5^s_{t-1} < MA20^s_{t-1} \&\& MA5^s_t > MA20^s_t] \\
&\quad \&\& [Grad5^s_t > 0 \&\& Grad20^s_t > 0]), s \in \mathscr{S}, t \in \mathscr{T}\} \\
D_{\text{TU}} &= \{(\mathbf{x}_{s,t}, o_{s,t}) | (Grad5^s_{t-1} < 0 \&\& Grad5^s_t > 0) \\
&\quad \| (Grad10^s_{t-1} < 0 \&\& Grad10^s_t > 0) \\
&\quad \| (Grad20^s_{t-1} < 0 \&\& Grad20^s_t > 0), s \in \mathscr{S}, t \in \mathscr{T}\}
\end{aligned} \tag{1}
$$

From among the hundreds of patterns, we adopt these four patterns: *bearish*, *bullish*, *Golden Cross*, and *Turn Up*, which are defined in Eq. (1).[1] Here, $\mathscr{S}$ is the set of shares in the market, and $\mathscr{T}$ is the trading period. The patterns are based on the concept of *moving averages* [10]. For window sizes 5, 10, and 20, MA is a moving average and *Grad* is an increment in MA. For example, for a window of size 5, $MA5^s_t$ and $Grad5^s_t$ are calculated as

$$MA5^s_t = \frac{1}{5}\sum_{k=0}^{4} \mathrm{close}^s_{t-k}, \quad Grad5^s_t = \frac{MA5^s_t - MA5^s_{t-1}}{MA5^s_{t-1}},$$

where $\mathrm{close}_s(t)$ is the closing price on day t. The input vector of predictors, $\mathbf{x}_{s,t}$, comprises 138 elements. There are 80 continuous and 58 discrete features to represent the arrangement of moving-average lines and heuristics from technical analysis.[2] The value $o_{s,t}$, a desired output of $\mathbf{x}_{s,t}$, is defined as a cumulative rate of change:

$$o_{s,t} = \sum_{k=1}^{k=T} \gamma^{k-1} RC_s(t+k), \quad 0 \leqslant \gamma \leqslant 1, \tag{2}$$

where γ is a discounting factor and T is the future interval under consideration. RC refers to the *rate of change* of the daily price:

$$RC_s(t) = 100 \times \frac{\mathrm{close}_s(t) - \mathrm{close}_s(t-1)}{\mathrm{close}_s(t-1)}. \tag{3}$$

For pattern e, the local trader LT_e comprises $\{NN_e, LP_e\}$, the predictor and the local policy, respectively. LT_e is constructed in two steps.

The first step is to optimize the predictor. For each pattern e, an artificial neural network NN_e is trained. First, the training data set T_e and the validation data set V_e are collected as

$$T_e = \{(\mathbf{x}_{s,t}, o_{s,t}) | (\mathbf{x}_{s,t}, o_{s,t}) \in D_e \quad \forall s, t = tday_1, \ldots, tday_{|T|}\},$$
$$V_e = \{(\mathbf{x}_{s,t}, o_{s,t}) | (\mathbf{x}_{s,t}, o_{s,t}) \in D_e \quad \forall s, t = vday_1, \ldots, vday_{|V|}\},$$

where *tday* and *vday* are the index sets of the training period and the validation period, respectively. $|T|$ and $|V|$ are the sizes of the sets. On these data sets, the learning of NN_e is formulated in terms of an optimization that minimizes the quadratic error:

$$\arg\min_{\mathbf{w}_e} \frac{1}{2} \sum_{(\mathbf{x}_{s,t}, o_{s,t}) \in T_e} \{NN_e(\mathbf{x}_{s,t}; \mathbf{w}_e) - o_{s,t}\}^2, \tag{4}$$

[1] For a more detailed description of the meaning of Eq. (1), see [10,13].
[2] Additional explanation can be found in [13].

where $\mathbf{w}_e$ is the model parameter vector of NN_e. We confined the structure of the neural networks to two hidden layers with a hypertangent activation function on each neuron. The learning is batch-processed by the scaled conjugate gradient (SCG) method and is stopped when the performance on the validation set has degenerated, following the principles of [2]. This process is repeated over the various configurations of hidden neurons, and the best is chosen as the final predictor.

The second step in constructing the local trader is to optimize a local policy. The time series of stock prices is highly noisy. Therefore, although reducing the prediction error is important, the trading performance can be changed by using the local traders. We introduce a local policy as follows. Given a space $\Omega = (\text{B\_THRES}, \text{A\_THRES}, \text{H\_DURATION})$, and a local trader LT_e, a local policy $LP(LT_e)$ comprises parameters for a local trader:

$$(b\_thres_e, a\_thres_e, h\_duration_e) := LP(LT_e).$$

Table 1 shows the meaning of each parameter. $LP(LT_e)$ determines: b_thres_e, the criterion for the bid signal; a_thres_e, the criterion for the ask signal; and $h\_duration_e$, the maximum number of days to hold the purchased stock. If the predicted value $NN_e(\mathbf{x}_{s,t})$ of the share s on day t is greater than b_thres_e, a bid signal for the share is generated. If $NN_e(\mathbf{x}_{s,t})$ is lower than a_thres_e, an ask signal is produced. To optimize the local policy, the following simulation-based approach is adopted:

```
procedure PPT(NN, 𝒟, b_thres, a_thres, h_duration)
    profit = 0
    for s, t in 𝒟
        if NN(x_{s,t}) > b_thres
            sell the stock based on a_thres and h_duration.
            profit = profit + (ask_{s,t} − bid_{s,t} − tr_{s,t})/bid_{s,t}
        end if
    end for
    return average(profit).
```

Table 1
Parameters of local policy

Name	Meaning
b_thres	Threshold of bid signal
a_thres	Threshold of ask signal
h_duration	Period of holding a stock

Table 2
Performances of local predictors

Local traders	Accuracy (%)	PPT (%)	Structure	Coverage
LT_{normal}	57.10	0.79	72×16	100.0
LT_{bear}	69.42	1.41	60×15	20.3
LT_{bull}	73.37	1.99	58×18	19.7
LT_{GC}	71.10	1.72	25×14	9.7
LT_{TU}	70.97	1.63	30×10	12.0

Given the predictor NN and the exclusive data set $\mathscr{D}$, the simulation calculates the profit per trade (PPT). Here, $\text{ask}_{s,t}$, $\text{bid}_{s,t}$ and $\text{tr}_{s,t}$[3] are the asking price, the bidding price, and the transaction cost, respectively, for the trading of share s at day t. In this simulation, the optimization of the local policy is represented as:

$$\underset{b\_thres, a\_thres, h\_duration \in \Omega}{\arg\max} \text{PPT}(NN, \mathscr{D}, b\_thres, a\_thres, h\_duration). \tag{5}$$

In contrast to Eq. (4), which is supervised learning that minimizes the error between the desired output and the predicted value, the objective function in Eq. (5) has no desired output and is not differentiable. To find the solution to Eq. (5), we restrict the search space Ω to the form of a grid. B_THRES consists of values from 0.25 to 0.54 in increments of 0.01, A_THRES is from -0.54 to -0.25 in 0.01 increments, and H_DURATION is from 5 to 15 in unit increments. For this case, the number of possible combinations is 9900. This is manageable by a greedy search method, i.e. enumerating each configuration of the parameters, calculating the PPT for this configuration, and selecting the configuration that maximizes the PPT.

Table 2 summarizes the results. The *accuracy* is defined as:

$$accuracy = \frac{\#\text{ of successful trades}}{\#\text{ of recommendations}}, \tag{6}$$

where a successful trade is a trade achieving positive profit after subtracting the transaction cost. The LT_{normal} corresponds to the case of an all-in-one predictor. The fourth column shows the optimal structure of the learned neural networks, where 72×16 means that the first hidden layer has 72 neurons, and the second layer has 16 neurons. The coverage of each pattern is given in the last column. D_{bear}, D_{bull}, D_{GC}, and D_{TU} cover 20.3%, 19.7%, 9.7%, and 12% of the stock-price series in the Korean stock market, respectively. Although our multiple-predictor approach covers only 70% of all stocks, this result is considered meaningful. The improved accuracy of each local trader is 12% more than

[3] In this paper, a transaction cost is fixed at 0.7% with respect to the selling price, including the commission money and the price slippage.

LT_{normal}. In terms of the PPT, the local traders are remarkably superior to LT_{normal}. Every local trader achieves nearly twice the PPT of LT_{normal}.

3. Need for a meta policy

We have analyzed the tendencies of our local traders. Fig. 1 is a bar graph showing the successful or failed recommendations of local trader LT_{bear} when the local policy of LP_{bear} is applied. Three hundred and eighty trading days are represented along the horizontal axis. The vertical axis represents the total sum of profits, in percentage terms, induced by selling the purchased stocks on each trading day. The sum is calculated without considering the asset limitations, that is, it is the simple sum of the trading results under the assumption that any recommendation could be bought. Viewing the recommendations from this bearish pattern, we can see that the profitable recommendations are not equally distributed over the entire trading period. The trades concentrated into some specific periods have a major effect on the resulting PPT of 1.41 for LT_{bear}.

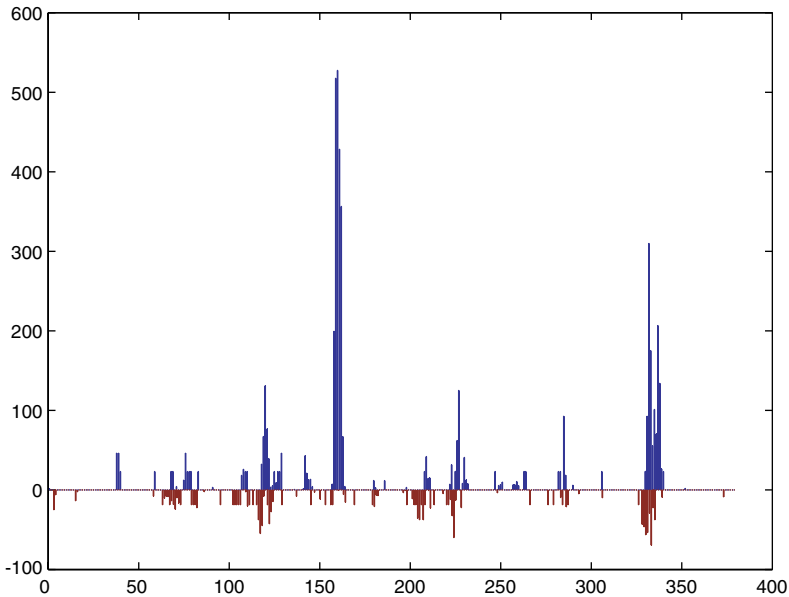

Fig. 1. Tendency of recommendations of LT_{bear}. The x-axis represents training days, and the y-axis represents the sum of profits, in percentage terms, induced by recommendations for a day. An upward bar means that positive sums dominate negatives for that day. A downward bar means the opposite.

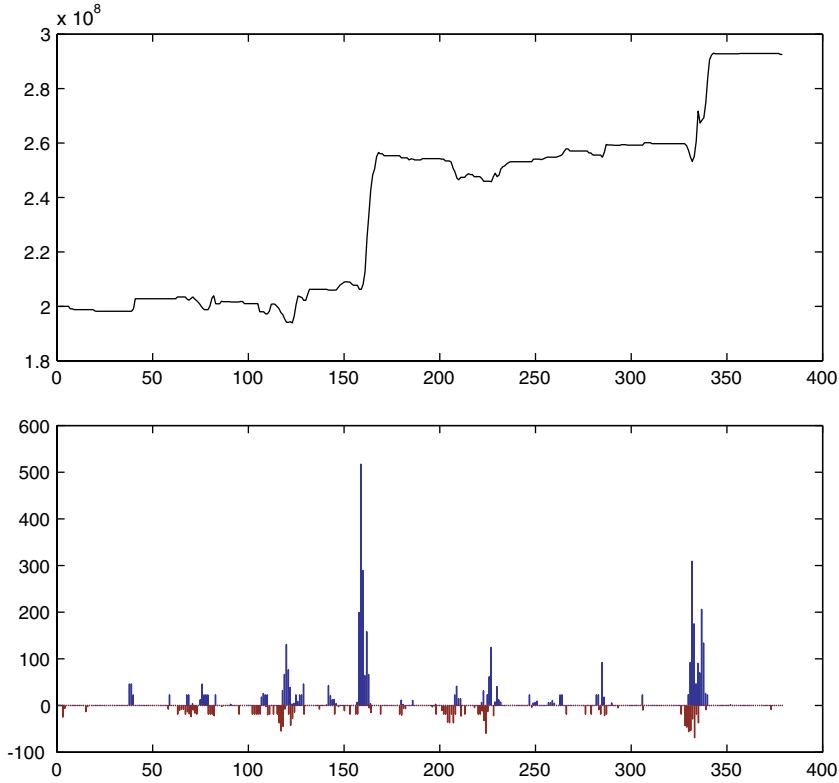

Fig. 2. Funding history and traded recommendations when the purchase money per recommendation is fixed at 0.4 million Won. The upper figure shows the change in asset over a trading period. Whenever there is a candidate share, the fixed amount of asset is used to trade it. The asset volume is shown in the vertical axis in monetary units, i.e., Won. The horizontal axis represents trading days. The lower figure shows the sum of the profits this trading scheme induces for the trading period, as a percentage.

Since asset limitations apply to real investments, we have experimented further to show how the trading results vary according to the purchase money per recommendation (PMR). Given an initial asset of 200 million Won,[4] we set up two configurations, with PMRs of 0.4 million Won and 40 million Won, respectively. Fig. 2 shows the results for the former configuration, and Fig. 3 shows the results for the latter. The upper part of each figure shows the history of the asset growth in Won. The lower part shows the sum of profits as a percentage. When the purchased stock is sold, its induced profit is added to the sum for the day it is recommended.

[4] The Won is a Korean monetary unit.

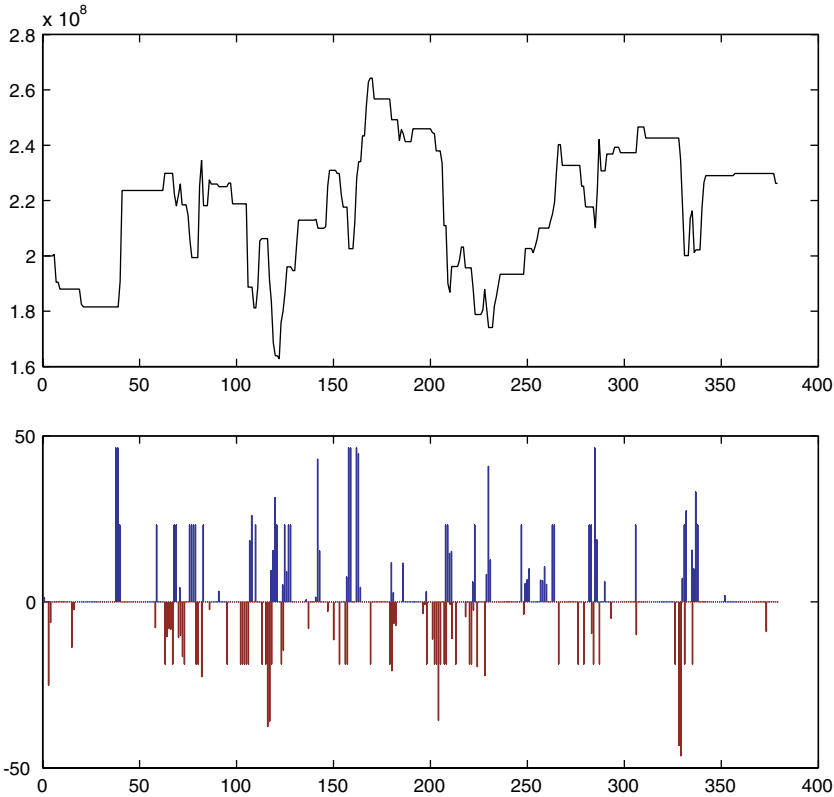

Fig. 3. Funding history and traded recommendations when the purchase money per recommendation is fixed at 40 million Won.

For a small PMR, we can manage most recommendations, even when the recommendations are dramatically increased for some days, e.g., around the 160th or 340th trading day. It is clear that these unusually profitable days contribute to the asset volume. However, for more usual days, with a small number of recommendations, a small quantity of asset is used as the stock fund. For those days, although a trade may induce a high profit ratio, it has little influence on the asset volume. For a large PMR, the asset is heavily affected by the result of each trading, as shown in Fig. 3. Even a small number of purchases can lead to the entire asset being exhausted. Moreover, from the variation in the profits of recommendations, it may happen that the purchased stocks induce a loss, while the stocks that are recommended but cannot be bought might have been profitable.

Therefore, if we use different PMRs for variable numbers of recommendations, the trading might be more effective. However, since we have several local

```
for t = 1 to T
   for e = 1 to E
      ({S_e}, N_e) = retrieve(NN_e, LP_e)
   end for
   N = (N_1, ··· , N_E)
   (PMR_1, ··· , PMR_E) = MP(N, SF)
   for e = 1 to E
      local_trade(PMR_e, {S_e}, LP_e)
   end for
end for
```

Fig. 4. The stock trading process with MP.

traders, the adaptation becomes more complicated. To complete the stock trading with this idea included, we define a *meta policy* (MP), an asset-allocation strategy that distributes a limited asset over the recommendations of the local traders:

Definition 1. Let the number of recommendations of local traders be $N = (N_{\text{bear}}, N_{\text{bull}}, N_{\text{GC}}, N_{\text{TU}})$, and the ratio of the stock fund over the asset be SF. A meta policy MP is a $\mathbb{R}^4 \times \mathbb{R} \mapsto \mathbb{R}^4$ function on N and SF and is defined as

$$(\text{PMR}_{\text{bear}}, \text{PMR}_{\text{bull}}, \text{PMR}_{\text{GC}}, \text{PMR}_{\text{TU}}) := MP(N, SF),$$

where PMR_e is the PMR of local trader e for its recommendations.

Fig. 4 shows the stock trading process with MP. At the tth day of T trading days, E local traders retrieve their recommendations $\{S_e\}$. MP determines the PMR for each local trader. The decisions are based on two types of information, namely the summaries of the recommendations of local traders and the stock fund ratio over the asset. Recommended stocks are bought and sold according to the LP_e with PMR_e, which is conducted by local_trade. In the next section, we present a reinforcement-learning framework in which stock trading with MP is formulated.

4. Reinforcement learning of meta policy

Reinforcement learning is a computational approach to understanding and automating goal-directed learning and decision-making. We introduce reinforcement learning using the notations of Sutton [24]. In a reinforcement-learning framework, especially in an MDP, an agent and an environment interact with each other on a discrete time scale $t = 0, 1, 2, \ldots, T$. The agent selects an action $a_t \in \mathscr{A}$ from its policy π, based on the state $s_t \in \mathscr{S}$ of the environment. If an action s_t is taken by the agent, the environment changes its state to s_{t+1} in response, and generates a reward $r_{t+1} \in R$ to the agent.

If one-step state-transition probabilities and one-step expected-reward models were available, the environment could be completely described as

$$p^a_{ss'} = \Pr\{s_{t+1} = s' | s_t = s, a_t = a\},$$
$$r^a_s = E\{r_{t+1} | s_t = s, a_t = a\}$$

for all s, $s' \in \mathscr{S}$ and $a \in \mathscr{A}$. The objective of the agent is to learn an optimal policy, i.e. a mapping from states to action. This maximizes the expected discounted future reward from the state–action pair (s,a), which is called the action-value function $Q^\pi(s,a)$. Given an episodic task, which is defined as the history of interactions from the starting state s_0 to the terminal state s_T, $Q^\pi(s,a)$ is defined as

$$Q^\pi(s,a) = E_\pi\{r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t-1} r_T | s_t = s, a_t = a\},$$

where $0 \leqslant \gamma \leqslant 1$ is a discount-rate factor that determines the effect of future rewards on the present value. A reward receiving k time-steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately. Then the optimal policy, denoted Q^*, can be defined as

$$Q^*(s,a) = \max_\pi Q^\pi(s,a)$$

for all s and a.

There are many kinds of algorithms for learning optimal policy, and they can be grouped into three categories according to the construction of their backups. First, dynamic programming uses full backup, can always converge, but requires exact modeling of the environment. Second, the Monte Carlo technique uses only sample backups of the entire episode. It does not require a model, but needs enormous episodes for convergence. Last, the temporal difference (TD) method is a compromise between these algorithms, using both n-step samples and bootstrapping, namely a currently learned value function. In practice, the TD method is widely used for its ease of handling, but much caution is needed regarding its convergence.

In the early stages of reinforcement-learning theory, deep understanding about the convergence of the TD method, with the value function represented as a table, was widely studied [9,24]. Since many practical applications of reinforcement learning have too many states to manage its reward model with the table method, it is common to use a linear or nonlinear function approximator as the value function model. However, several studies have shown that a simple linear approximator can diverge [2,25]. As a result, designers must pay attention to the convergence of their models.

4.1. Stock trading with MP by reinforcement learning

In this work, all the decision-making to be formulated via reinforcement learning is reduced to the MP, an asset-allocation strategy. Fig. 5 shows the

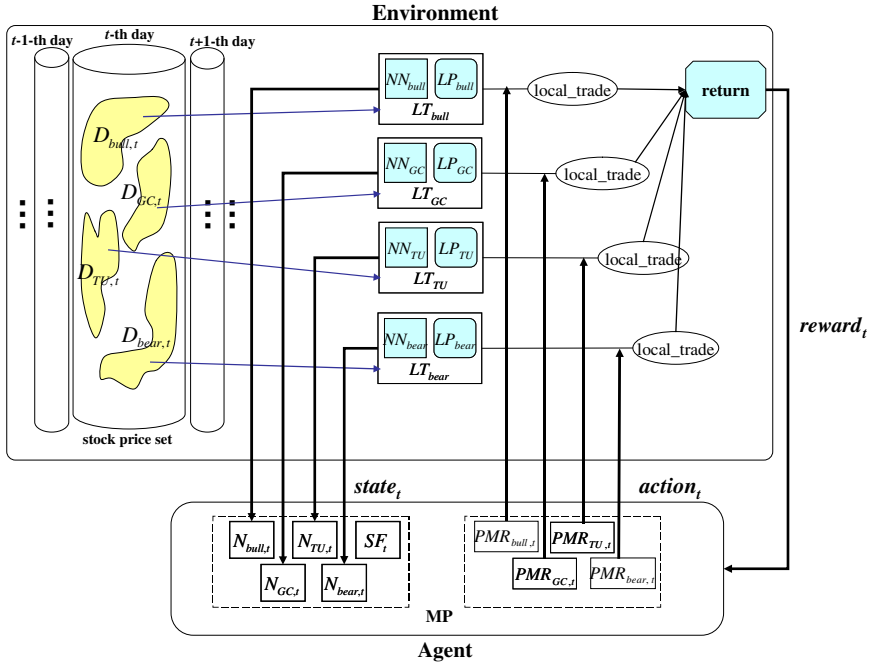

Fig. 5. The overall architecture for stock trading in a reinforcement-learning framework. The stock market corresponds to the environment, and the MP corresponds to the agent. $N_{e,t}$ is the number of candidates the local trader LT_e retrieves. SF_t is the stock fund ratio. $\mathrm{PMR}_{e,t}$ corresponds to the action that is the PMR for LT_e. Buying and selling by local traders are conducted by local_trade. Reward is the profit ratio of Eq. (8).

overall architecture for stock trading. The environment includes the unknown dynamics of the stock market. It also contains the local traders as components. The MP corresponds to the learning agent for reinforcement learning. On the tth day, local traders retrieve their candidate stocks. The state comprises the number of recommendations of each local trader $N_{e,t}$ and the stock fund ratio over the asset SF_t. The action of the agent is the PMR for each local trader. The reward is the profit ratio of Eq. (8), which is defined later in this subsection.

One of the most important keys to achieving reasonable performance in machine learning is the representation of the input space. In particular, reinforcement learning is the art of the design of state, action and reward. Here, we describe how we design them to be manageable in a Q-learning framework.

The state s_t is a state vector at time t that the environment gives to the agent. We define s_t as

$$s_t = (N^{\mathrm{bits}}_{\mathrm{bear}}(t), N^{\mathrm{bits}}_{\mathrm{bull}}(t), N^{\mathrm{bits}}_{\mathrm{GC}}(t), N^{\mathrm{bits}}_{\mathrm{TU}}(t), SF^{\mathrm{bits}}(t)).$$

The state vector comprises two parts: the information vector about the local traders, and the stock fund ratio. Each $N_e^{\text{bits}}(t)$ is a bit vector that specifies the number of recommendations by LT_e at time t. Although the number of recommendations is an unbounded integer, we restrict it so that it can be represented in a 10-bit orthogonal vector, as shown in Table 3.

$SF^{\text{bits}}(t)$ is a bit vector representing the ratio of the stock fund over the asset. Since the ratio can range from 0% to 100%, we discretize it into 20 intervals, and express it as a 20-bit orthogonal vector, as shown in Table 4.

The action a_t on s_t is the set of PMRs for local traders. Although it is natural to use real-valued ratios, it is difficult to envision a real-valued action in Q-learning. We have to appropriately discretize, so as to reduce the cardinality of the action. If we restrict the ratio to having one of four values from $(0.5, 1.0, 3.0, 5.0)$, the action can be written as

$$a_t = (\text{PMR}_{\text{bear}}^{\text{bits}}(t), \text{PMR}_{\text{bull}}^{\text{bits}}(t), P_{\text{GC}}^{\text{bits}}(t), P_{\text{TU}}^{\text{bits}}(t)),$$

where each $\text{PMR}_e^{\text{bits}}(t)$ is one of four-bit orthogonal vectors that specify the corresponding PMR value. In total, there are 256 possible actions.

If we use a table representation of the action-value function for the Q-learning, we might need $10^4 \times 20 \times 4^4 = 51.2$ million entries. This is a substantial table size, but it is not unmanageable with current computing power. Therefore, we do not adopt any function approximators.

Fig. 6 describes our Q-learning framework for the MP. To produce episodes over the training period, both the start and the end days of an episode are uniformly sampled over the total training period. Over the course of an episode,

Table 3
Bit vector representation of recommendations

# of recommendations	Bit vector
0	0000000001
1	0000000010
...	...
8	0100000000
9∼	1000000000

Table 4
Bit vector of the stock fund ratio

Ratio of the stock fund	Bit vector
[0, 5)	00000000000000000001
[5, 10)	00000000000000000010
...	...
[90, 95)	01000000000000000000
[95, 100)	10000000000000000000

Initialize $Q(s, a)$ as zero.
Repeat (for specified number of episodes):
  Make an episode by sampling T_1 and T_n.
  Environment initializes $asset_{T_1}$ and s_{T_1}.
  for $t = T_1$ to T_{n-1}
    $a_t \longleftarrow Action(s_t)$
    $\text{asset}_{t+1} \longleftarrow Trade(\text{asset}_t, a_t)$
    if $t < T_{n-1}$ then
      $r_t \longleftarrow 0$
    else
      $r_t \longleftarrow$ profit ratio
    end if
    Environment produces s_{t+1}
    $\delta_t \longleftarrow r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)$
    $Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha \times \delta_t$
  end for

Fig. 6. The Q-learning algorithm for meta policy. The function *Action* chooses the action by Eq. (7) and the function *Trade* simulates a trading day.

the agent chooses an action from the function *Action*, trades stocks with the chosen action, is given rewards, and updates the tables of Q-values. The function $Action(s_t)$ has the form:

$$a_t \leftarrow \begin{cases} \text{proportional sampling} & \text{if } u < 1 - \epsilon, \\ \text{random sampling} & \text{otherwise,} \end{cases} \tag{7}$$

where u is a uniformly sampled random number between $[0, 1)$. With probability $1 - \epsilon$, $Action(s_t)$ chooses an action proportional to the values of each action for s_t. Since the search space is quite large, we do not adopt a greedy policy, which would mean that $a_t \leftarrow \text{argmax}_a Q(s_t, a)$. The parameter ϵ is a parameter to control the search behavior of the agent. The smaller the value of ϵ, the more exploitative is the search, which means that the agent chooses its action based primarily on the current Q-value. The alternative is explorative search, whereby the agent has a chance to search more widely. The function $Trade(\text{asset}_t, a_t)$ simulates a one-day trading process, according to a_t with an initial asset_t. The output of *Trade* becomes the initial asset for the next day. The parameter α determines how much Q-value is updated by δ_t, similar to the learning-rate parameters of other iterative-learning algorithms.

As the reward function, we choose the profit ratio on the trading period, which is defined as

$$\text{profit ratio} = 100 \times \frac{\text{asset}_{T_n} - \text{asset}_{T_1}}{\text{asset}_{T_1}}. \tag{8}$$

The reward for each action is zero inside of the trading period and is given only on the final day of the episode.

5. Experimental results

In this section, we experimentally analyze the performance of the stock trading with MP optimized by *Q*-learning. We refer to this as a meta policy generator (MPG). Two other trading systems, namely policy1 and policy2, are used for a performance comparison with MPG. Each trading system is briefly summarized in Table 5. The trading systems use the same local traders, but have their own asset-allocation policy. Table 6 shows how the data are organized for training and testing.

5.1. Korean stock market

In Korea, there are two stock markets, the Korea Stock Exchange market and the KOSDAQ market. Since the birth of the KOSDAQ market in 1995, the Korea Stock Exchange market has been called KOSPI, literally referring to the benchmark Korea Composite Stock Price Index. The KOSPI market comprises about 840 shares, and the KOSDAQ market comprises about 890 shares, as of July 2005. The KOSDAQ market has a higher turnover rate and is more volatile than the KOSPI market. In this paper, we focus on evaluating the proposed method for the KOSPI market. However, the simple trading performance for the KOSDAQ market is also provided to show the validity of the method.

Table 5
Policies of the trading systems

Name	Description
policy1	Fixed policy, initially partitioned asset
policy2	Fixed policy, unified asset
MPG	Adaptive policy, unified asset

Table 6
Partitions of the data

Group	Period
Training set for each local trader	January 1998–March 2000
Training set for MPG	April 2000–December 2001
Test set for trading systems	January 2002–May 2003

5.2. The asset-allocation policy of each trading system

Figs. 7 and 8 show how the assets of `policy1` and `policy2` are managed. For `policy1`, it is as if we had a sub-trading system for each local trader, with the initial asset being divided by the number of local traders into equal parts. During the entire trading period, the divided assets are exclusively used by their corresponding local traders. The asset-allocation policy of `policy1` comprises four parameters:

$$\mu_{\text{bear}}, \mu_{\text{bull}}, \mu_{\text{GC}}, \mu_{\text{TU}},$$

where μ_e is the PMR for local trader e. For each local trader, the parameters were optimized so that it could achieve maximal profit on simulated trading over the period between April 2000 and December 2001. The simulation is identical to the process of Fig. 4 except that μ_{bear}, μ_{bull}, μ_{GC}, and μ_{TU} are used to calculate PMRs, rather than MP.

In `policy2`, the asset is not divided, but is managed in a unified form. Whenever there is a recommendation from local traders, `policy2` allocates

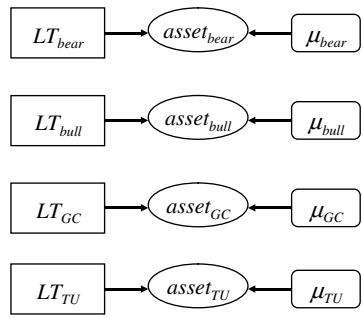

Fig. 7. The asset-allocation strategy of `policy1`.

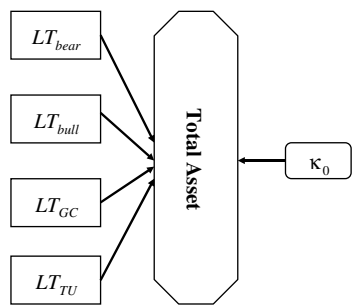

Fig. 8. The asset-allocation strategy of `policy2`.

k_0 times the asset to the purchase, where k_0 was tuned on the same simulation as `policy1`.

5.3. Training the MPG

The `MPG` was constructed by Q-learning as described in Section 4. It was trained over the period from April 2000 to December 2001, the same period as the tuning period for `policy1` and `policy2`. There is no algorithmic way to decide the learning parameters of Q-learning, but we specified the parameters using conventional practice. The discount-rate factor γ was set to 0.9. In Q-learning, it is a convention to set α small enough that the update of Q-value proceeds gradually, so we chose 0.02 for α. A controlling parameter for explorative versus exploitative, ε, was set to 0.2.

The high uncertainty of the stock market makes overfitting on the training period inevitable. Therefore, for the training set of `MPG`, we excluded the last six months of the training period as a validation period, which we used to monitor the degree of overfitting. For this purpose, after every 10 episodes are experienced, the trading performances of Eq. (8) over the training period and the validation period are estimated by the process of Fig. 4. Fig. 9 shows the

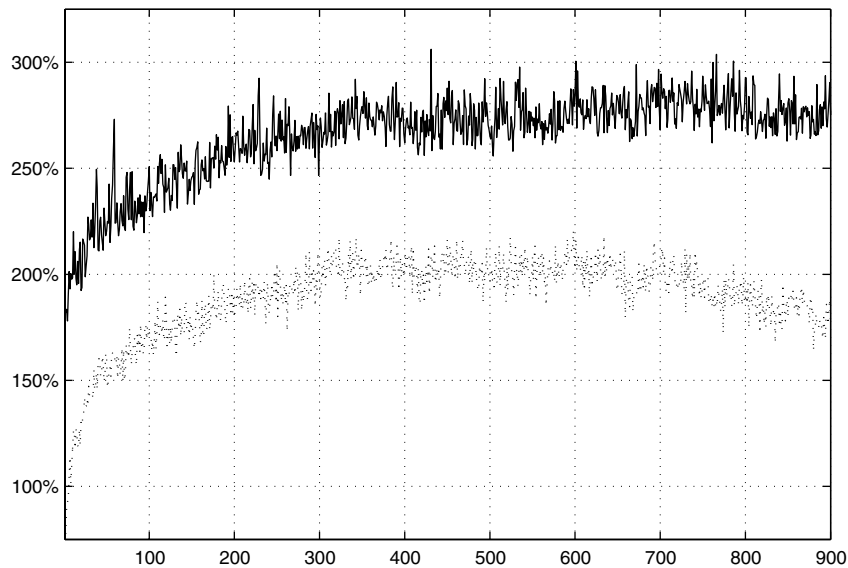

Fig. 9. The tendency of Q-learning. The profit ratios are estimated by the simulation in Fig. 4 for the training period and the validation period. The horizontal axis represents the estimating points after every 1000 episodes. The vertical axis shows the profit ratios as a percentage. The solid line is for the training period and the dotted line is for the validation period.

results. The points for every 1000 episodes are shown on the horizontal axis for visibility. The vertical axis represents the profit ratios corresponding to these points. The solid line shows the profit ratios for the training period and the dotted line shows the profit ratios for the validation period. The performance of the validation period reaches its maximum of about 210% after 500,000 episodes. After this, experiencing more episodes makes the performance decline, which means that additional updating of Q-values would lead to overfitting. Thus, we stop further training, and use this result as the MP.

5.4. Trading test

Fig. 10 shows the comparison of the three trading systems over the test period from January 2002 to May 2003. The horizontal axis is trading days and

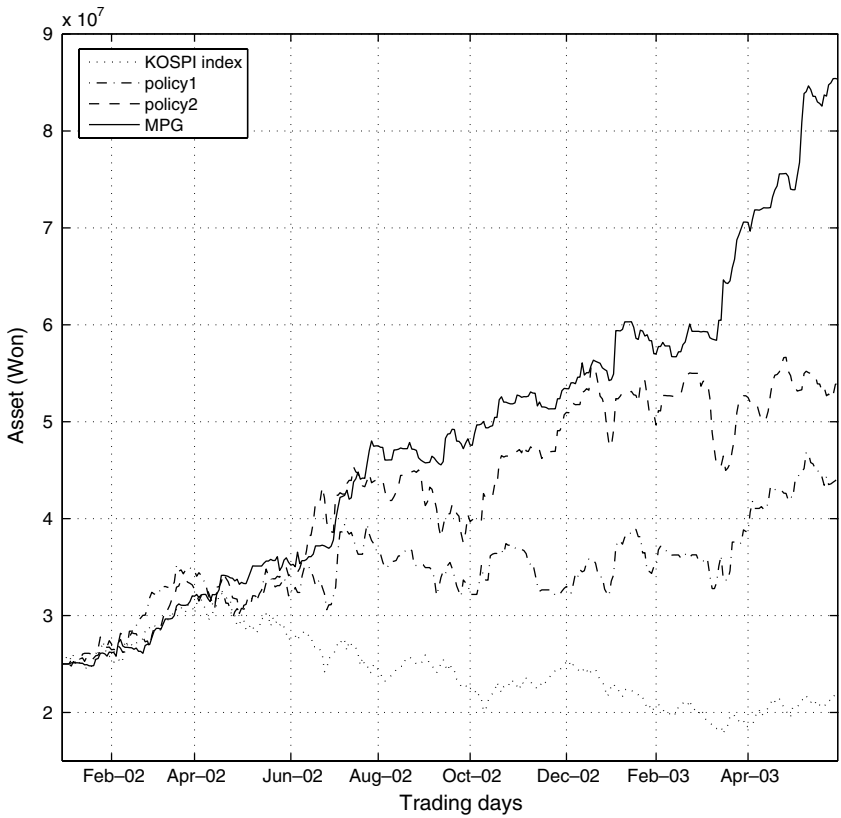

Fig. 10. A comparison of the performances of the three trading systems. The horizontal axis shows days from January 2002 to May 2003. The vertical axis shows the asset volume in Won. Lines are: dotted (KOSPI index), dashed-dotted (`policy1`), dashed (`policy2`), and solid (`MPG`).

the vertical axis shows the assets in Won. Each trading system started with its initial asset being 25 million Won. The benchmark KOSPI index is a composite index of the Korean stock market, representing the tendency of the KOSPI market, similar to America's Dow Jones Average (DJA) index or Japan's NIKKEI index. The KOSPI index was scaled up so that we could visualize it as an imaginary asset of 25 million Won on January 2, 2002. The solid line is the history of the asset of `MPG`, the dashed-dotted line is for `policy1`, the dashed line is for `policy2`, and the dotted line is for the KOSPI index.

Table 7 summarizes the performance induced by each trading system. The profit ratio is calculated via Eq. (8) and the relative profit ratio is calculated by

$$R(\text{asset}) = \frac{\text{asset}_T - \text{composite}_T}{\text{composite}_T}, \tag{9}$$

which is the relative profit compared with the benchmark composite index on day T, at the end of the trading period.

Clearly, `MPG` outperforms the other two trading systems. After 17 months' trading, its asset rose by nearly 65 million Won, or 257.76%, to about 90 million Won. Here, `policy1` is seen to be the worst of the three trading systems. There may be a situation when a local trader has insufficient funds to afford its newly recommended stocks, while other traders have sufficient money in reserve.

In addition, `policy2` also shows a limited performance. Since it tries to buy stocks regardless of which local trader recommends them, it might avoid the problem of `policy1` mentioned above. Although it does outperform `policy1` over most of the period, `policy2` does not consider the relationship between the local traders or the stock fund ratio information. During the difficult period of August and September 2002, `policy1` and `policy2` suffer a decline of the asset, but `MPG` survives this difficult period.

To examine the generality of the proposed method, additional experimental results are presented for the KOSDAQ market. Fig. 11 shows the performances of the three trading systems on this market. The training, validation, and test period of the local traders and the policies were the same as for the KOSPI market. (The reason why we have focused on KOSDAQ, the other Korean stock market, is that we have encountered difficulties in acquiring full informa-

Table 7
The profit induced by each trading system in the KOSPI market

Trading systems	Profit ratios (%)	Relative profit ratios (%)
`policy1`	76.92	102.49
`policy2`	115.74	146.92
`MPG`	257.76	309.46

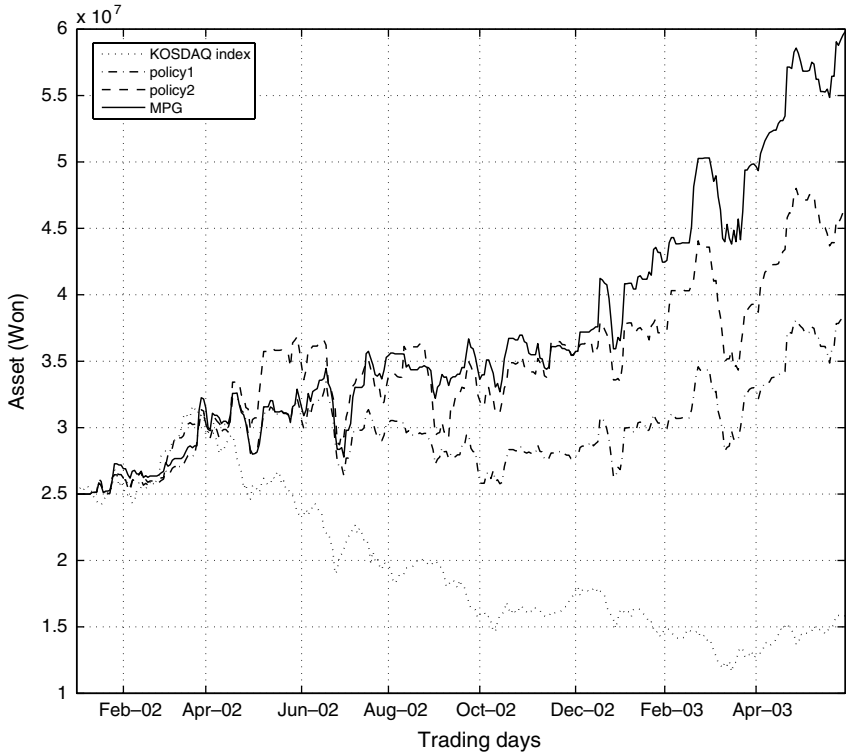

Fig. 11. A comparison of the performances of the three trading systems. The test was conducted on the KOSDAQ market. Lines are: dotted (KOSDAQ index), dashed-dotted (`policyl`), dashed (`policy2`), and solid (`MPG`).

tion about stocks in the markets of other nations, which is a necessary element of the proposed method.)

The KOSDAQ market shows such high volatility that the KOSDAQ index dropped, by 63.27%, from its maximum on March 18, 2002, to its minimum on March 11, 2003. In the KOSPI market, the drop from its maximum to its minimum was only 45.70%. The trading systems suffered more severe fluctuations in their asset than those for KOSPI. Even `MPG` failed to maintain the asset in late June, 2002, which was achieved for the KOSPI case. The trading performances are summarized in Table 8. The earning profit ratios are small compared with the KOSPI case, while the relative profit ratios are comparable. Although the `MPG` for KOSDAQ did not avoid as many hazards as the `MPG` for KOSPI, it is superior to the other two conventional policies. This additional experiment increases confidence in the assertion that the adaptive policy of `MPG` can induce better performance than the conventional fixed asset-allocation schemes.

Table 8
Profit induced by each trading system on the KOSDAQ market

Trading systems	Profit ratios (%)	Relative profit ratios (%)
policy1	52.23	140.51
policy2	84.69	191.59
MPG	139.27	277.96

6. Discussion

To investigate the characteristics of MPG, an internal analysis of MPG, which includes the stock fund ratio and the information about recommendations, is given in Fig. 12. The upper part of Fig. 12 shows the asset history. The middle part shows the stock fund ratio over the entire asset. The lower part shows an imaginary sum of the profits, assuming that the recommended stocks could be bought according to their local policy, without considering asset limitations. After April 2002, KOSPI was in bear-market mode. From late June 2002 to mid-July 2002, there was a sharp rally in KOSPI. LT_{bear} made dramatic recommendations on June 27, 2002. The PMRs for the next day were $(5.0, 0.5, 0.5, 1.0)$, where 5.0 is for LT_{bear}. The recommended stocks of LT_{bear} were purchased with more PMR than the others. Since the stock fund ratio was 5.3% on June 27, 2002, there was little difficulty in affording the many recommendations. As the result of the purchases, the stock fund ratio on June 28, 2002 was increased to about 91.5%.

In December 2002, KOSPI suffered the end of a short bull market, and the beginning of another bear market. After mid-December 2002, the number of recommendations had increased, which might have caused losses in asset. On December 20, 2002, LT_{bear} and LT_{bull} recommended four and six stocks, respectively. However, the PMRs were $(0.5, 0.5, 1.0, 0.5)$ with 0.5 for both LT_{bear} and LT_{bull}. For a few consecutive days following that day, with those PMRs, the stock fund had been prevented from making a dangerous increase. Although some loss was inevitable, it was more stable than for other trading systems. This situation was more vivid in early March 2003, when policy2 suffered from a severe loss of asset, while MPG minimized the hazard.

However, MPG does not always win. For example, during late January 2003, the stock fund increased from wrong recommendations, which caused a loss in the asset. Since the predictors of local traders are not perfect, it is hardly possible to induce constantly winning trades. However, the higher-level consideration of several local traders, as well as the stock fund ratio of MPG, makes trading more profitable and less risky than the conventional asset-allocation strategies for the local traders. It is a consequence of reinforcement learning

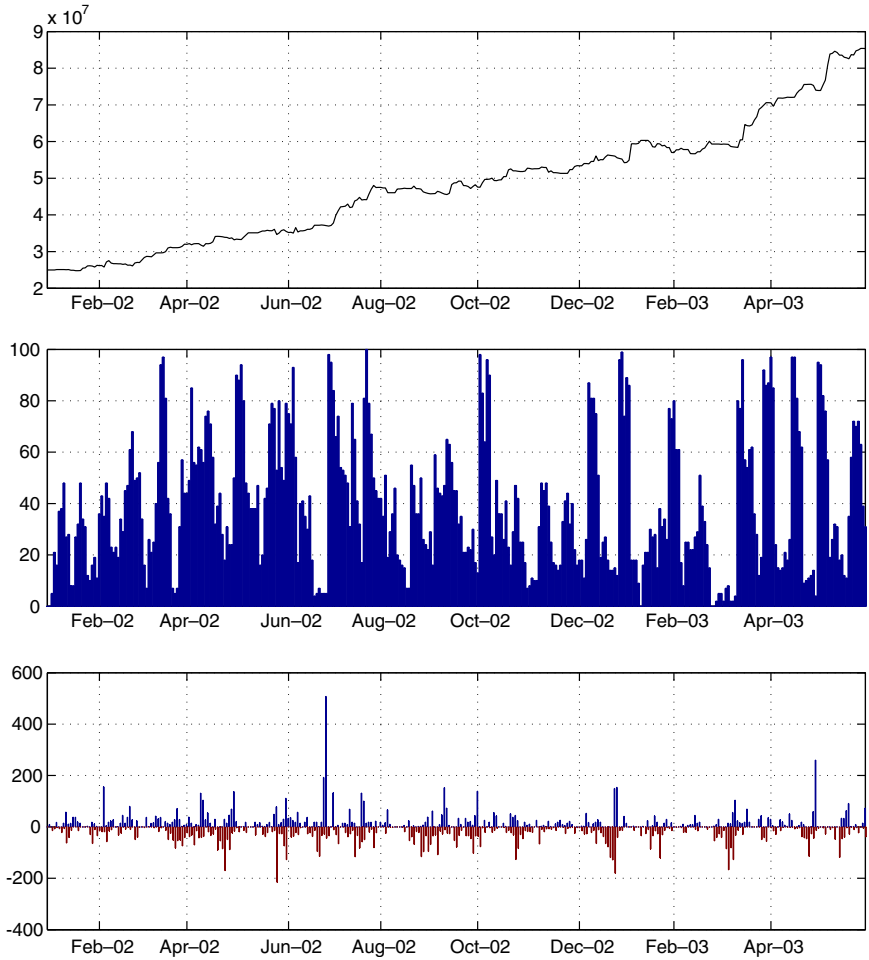

Fig. 12. Upper figure: asset history. Middle figure: the proportion of stock fund to asset. Lower figure: sums of positive and negative profits recommended per day. It is an imaginary summation, which assumes that all the recommendations were purchased and traded according to the local policy.

that induces this efficient trading. Time-delayed learning makes it possible to consider the result of temporal decisions that influence the asset. Therefore, although the local traders were fixed before constructing the asset allocation, the resulting trading performances were improved by an MP optimized by reinforcement learning. The risks, which are inevitable and not considered in the case of fixed asset-allocations, are mitigated or avoided by the MP.

There is an additional consideration. In Section 4, we discretized both the state and the action, which is inevitable in a Q-learning framework. Between these two, it is the cardinality of the action that has the major impact on the properties of the Q-value table. The greater the increase in cardinality, the more precise the representation becomes. At the same time, however, the table size increases exponentially. Therefore, we must maintain a trade-off between the resolution and the size of the table. In addition to the cardinality, the discretization level of the action influences the learned policy. In fact, there is no algorithmic way to do this, and we have to choose it by trial and error. Fig. 13 shows the trading performances of two different discretization levels. The solid line shows the result for the level we finally chose. The dotted line is for when the level is $(1.5, 3.0, 4.5, 6.0)$. The latter causes purchases with larger PMRs than the former. Thus, the resulting asset history becomes more volatile than

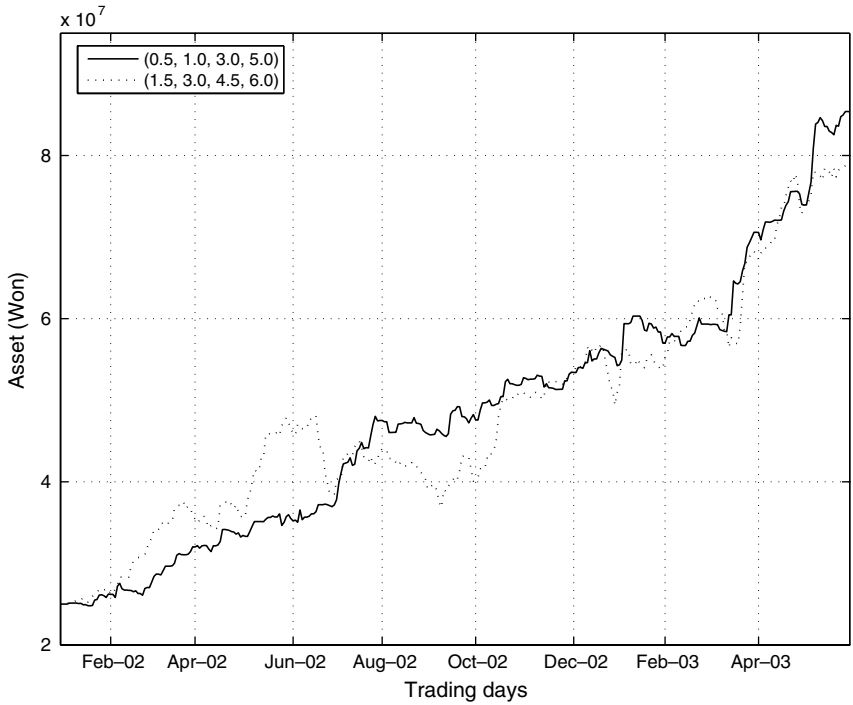

Fig. 13. A comparison of the performances when the discretization level of action is changed. The solid line shows the result of $(0.5, 1.0, 3.0, 5.0)$, and the dotted line shows the result of $(1.5, 3.0, 4.5, 6.0)$.

the former. Although the final asset of the former is slightly larger than the asset of the latter, some may prefer the aggressive strategy of the latter. Therefore, it is the designer's heuristics that determine the characteristics of the learned policy.

7. Conclusions

We have presented a stock trading method that integrates stock selection and asset management. Meta policy makes it possible to reduce a large decision-making problem to an asset management problem, i.e. deciding the purchase money per recommendation for the stock selection. With the meta policy designed as a compact representation, stock trading is formulated and optimized in a reinforcement-learning framework. In a simulation over a test period for the KOSPI market, trading with the meta policy induces about 258% profits, which is more than two times the profits of other fixed asset-allocation strategies. In addition, some risks that are unavoidable with the naive use of local traders are reduced by the proposed method. Our results show that reinforcement learning associated with meta policy is useful in solving a stock trading problem that is too complex to be solved directly.

Future research may address the issue of the increasing cardinality of the action, PMR in our case. Q-learning forces the real-valued action to be discretized, which causes the cardinality problem of the action. In this work, we restricted the cardinality to four for tractability reasons. If we increase the cardinality, more precise control of action can be achieved. However, this may compromise the tractability. Therefore, we are trying to maximize the cardinality while maintaining tractability. This involves adopting the extensions to reinforcement learning that can treat continuous action, such as [18].

Acknowledgement

This research was done in collaboration with NATURE SOLUTIONS. It was supported in part by the Ministry of Education and Human Resources Development, under the BK21-IT Program. The RIACT at Seoul National University provided research facilities for this study.

References

[1] G. Armano, M. Marchesi, A. Murru, A hybrid genetic-neural architecture for stock indexes forecasting, Information Sciences 170 (1) (2005) 3–33.

[2] R. Caruana, S. Lawrence, L. Giles, Overfitting in neural nets: backpropagation conjugate gradient, and early stopping, Advanced in Neural Information Processing Systems (2001) 402–408.
[3] C.-S.J. Chu, G.J. Santoni, T. Liu, Stock market volatility and regime shifts in returns, Information Sciences 94 (1/4) (1996) 189–190.
[4] R.O. Duda, P.E. Hard, D.G. Stork, Pattern Classification, Wiley-Interscience, New York, 2000.
[5] E.F. Fama, K.R. French, Dividend yields and expected stock returns, Journal of Financial Economics 22 (1988) 3–26.
[6] E.F. Fama, K.R. French, The cross-section of expected stock returns, Journal of Finance 47 (1992) 427–465.
[7] A. Fan, M. Palaniswami, Stock selection using support vector machines, in: Proceedings of the International Joint Conference on Neural Networks, 2001, pp. 1793–1798.
[8] X. Gao, L. Chan, An algorithm for trading and portfolio management using *Q*-learning and sharpe ratio maximization, in: Proceedings of the International Conference on Neural Information Processing, 2000, pp. 832–837.
[9] T. Jaakkola, M. Jordan, S. Singh, On the convergence of stochastic iterative dynamic programming algorithms, Neural Computation 6 (6) (1994) 1185–2201.
[10] P.J. Kaufman, The New Commodity Trading Systems and Methods, Wiley, New York, 1987.
[11] S.M. Kendall, K. Ord, Time Series, Oxford, New York, 1997.
[12] S.D. Kim, J.W. Lee, J. Lee, J.-S. Chae, A two-phase stock trading system using distributional differences, in: Proceedings of the International Conference on Database and Expert Systems Applications, 2002, pp. 143–152.
[13] J.W. Lee, S.D. Kim, J. Lee, J.-S. Chae, An intelligent stock trading system based on reinforcement learning, IEICE Transactions on Information and Systems E86-D (2) (2003) 296–305.
[14] J.W. Lee, J. O, A multi-agent *Q*-learning framework for optimizing stock trading systems, in: Proceedings of the International Conference on Database and Expert Systems Applications, 2002, pp. 153–162.
[15] R.S.T. Lee, iJADE stock advisor: an intelligent agent based stock prediction system using hybrid RBF recurrent network, IEEE Transactions on Systems, Man, Cybernetics, Part A: Systems and Humans 34 (3) (2004) 421–428.
[16] B.G. Malkiel, A Random Walk Down Wall Street, Norton, New York, 1996.
[17] O. Mihatsch, R. Neuneier, Risk sensitive reinforcement learning, Machine Learning 49 (2/3) (2002) 267–290.
[18] J.D.R. Millán, D. Posenato, E. Dedieu, Continuous-action *Q*-learning, Machine Learning 49 (2/3) (2002) 247–266.
[19] J. Moody, M. Saffell, Learning to trade via direct reinforcement, IEEE Transactions on Neural Networks 12 (4) (2001) 875–889.
[20] R. Neuneier, Enhancing *Q*-learning for optimal asset allocation, Advances in Neural Information Processing Systems, MIT Press, Cambridge, 1998, pp. 936–942.
[21] J. O, J. Lee, J.W. Lee, B.-T. Zhang, Dynamic asset allocation for stock trading optimized by evolutionary computation, IEICE Transactions on Information and Systems E88-D (6) (2005) 1217–1223.
[22] J. O, J.W. Lee, B.-T. Zhang, Stock trading system using reinforcement learning with cooperative agents, in: Proceedings of the International Conference on Machine Learning, Morgan Kaufmann, 2002, pp. 451–458.
[23] E.W. Saad, D.V. Prokhorov, D.C. Wunsch II, Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks, IEEE Transactions on Neural Networks 9 (6) (1998) 1456–1470.

[24] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, 1998.
[25] J.N. Tsitsiklis, B. Van Roy, Regression methods for pricing complex American-style options, IEEE Transactions on Neural Networks 12 (4) (2001) 694–703.
[26] C.J.C.H. Watkins, Learning from delayed rewords, Ph.D. thesis, Cambridge University, 1989.