PAPER

Dynamic Asset Allocation for Stock Trading Optimized by Evolutionary Computation

Jangmin O[†a)], Jongwoo LEE[††], *Nonmembers*, Jae Won LEE[†††], *Member*, *and* Byoung-Tak ZHANG[†], *Nonmember*

SUMMARY Effective trading with given pattern-based multi-predictors of stock price needs an intelligent asset allocation strategy. In this paper, we study a method of dynamic asset allocation, called the meta policy, which decides how much the proportion of asset should be allocated to each recommendation for trade. The meta policy makes a decision considering both the recommending information of multi-predictors and the current ratio of stock funds over the total asset. We adopt evolutionary computation to optimize the meta policy. The experimental results on the Korean stock market show that the trading system with the proposed meta policy outperforms other systems with fixed asset allocation methods.

key words: *stock market, multi-predictors approach, asset allocation, meta policy, evolutionary computation*

1. Introduction

During the last a few decades, several algorithms have been applied to the stock market problems [5]. But attempts at modeling or predicting the stock market have not been successful in *consistently* beating the market. This is the famous EMH (Efficient Market Hypothesis) saying that the future prices are unpredictable since all the information available is already reflected on the history of past prices [9]. However, if we step back from *consistently*, then we can find several empirical results saying that the market might be somewhat predictable [1]. Especially, the newest algorithms in artificial intelligence, equipped with powerful representational and modeling power, have been applied to the problems of the stock market, such as price prediction, risk management and portfolio optimization.

Many works have been applied to price prediction. Supervised learning such as neural networks, decision trees, and SVMs (Support Vector Machines) are intrinsically well suited to the problem [6], [13]. The risk management and portfolio optimization have been intensively studied in reinforcement learning [7], [10]–[12].

In this paper, our main interest is to trade individual stocks in the market using the proposed asset allocation scheme for portfolio optimization. The works with the stock price prediction based on supervised learning [6], [13] lack considering the risk management and portfolio optimization.

The mechanism of the time delayed learning with the cumulative rewards in reinforcement learning makes it natural to consider the risk management and portfolio optimization. But the researches [11] put simple assumptions on the market to make the problem manageable in the reinforcement learning framework. Also the portfolios of the researches [10] are simple because they focus on switching just between two price series. The works [7], [12] treat trading individual stocks in reinforcement learning but lack asset allocation.

In order to handle both stock price prediction and trading with efficient asset allocation, we divide the problem into two separate parts. For the price prediction, we have adopted a neural networks approach. But instead of an all-purpose predictor, we have built a multi-predictors approach based on some of interesting patterns. For the asset allocation, we focus in this paper how to dynamically distribute whole asset on each predictor's recommendations so that the asset can be used effectively and wisely. To achieve this aim we design a dynamic asset allocation strategy called *meta policy* exploiting the information of recommendations of the multi-predictors and the stock fund ratio over the asset. To optimize meta policy we adopt an optimization method from the evolutionary computation. Through a sophisticated design of representation, crossover, mutation, and fitness function specialized to the meta policy, the meta policy is successfully optimized.

The resulting trading performance is compared with the performances of other fixed asset allocation techniques through a simulation on the Korean stock market.

2. Summary of Predictors

We have constructed the pattern-based predictors. Since stock prices are considered highly nonstationary, it is difficult to make the prediction perfect on all shares over whole spots of their histories. Therefore we are focused on some of interesting price patterns. Equation (1) represents the data patterns of the predictors we are interested in. Here $\mathcal{S}$ is the set of stocks in the market and $\mathcal{T}$ is the training period. These patterns are based on the concept of *moving averages* [4]. D_{bear} and D_{bull} are obtained from stock price data in bear and bull market trends respectively. D_{GC} is constructed from the pattern of *Golden Cross* which is a widely

Manuscript received May 13, 2004.

[†]The authors are with Seoul National University, San 56–1, Shillim-dong, Kwanak-gu, Seoul, 151–742, Korea.

[††]The author is with Sookmyung Women's University, 53–12, Chongpa-dong, Yongsan-gu, Seoul, 140–742, Korea.

[†††]The author is with Sungshin Women's University, Dongsun-dong, Sungbuk-gu, Seoul, 136–742, Korea.

a) E-mail: jmoh@bi.snu.ac.kr

DOI: 10.1093/ietisy/e88–d.6.1217

$$
\begin{aligned}
D_{bear} &= \{feature(s,t) | MA5_t^s < MA10_t^s < MA20_t^s, s \in \mathcal{S}, t \in \mathcal{T}\} \\
D_{bull} &= \{feature(s,t) | MA5_t^s > MA10_t^s > MA20_t^s, s \in \mathcal{S}, t \in \mathcal{T}\} \\
D_{GC} &= \{feature(s,t) | (MA5_{t-1}^s < MA10_{t-1}^s \;\&\&\; MA5_t^s > MA10_t^s \;\&\&\; Grad5_t^s > 0 \;\&\&\; Grad10_t^2 > 0) \\
&\quad || (MA10_{t-1}^s < MA20_{t-1}^s \;\&\&\; MA10_t^s > MA20_t^s \;\&\&\; Grad10_t^s > 0 \;\&\&\; Grad20_t^2 > 0) \\
&\quad || (MA5_{t-1}^s < MA20_{t-1}^s \;\&\&\; MA5_t^s > MA20_t^s \;\&\&\; Grad5_t^s > 0 \;\&\&\; Grad20_t^2 > 0, s \in \mathcal{S}, t \in \mathcal{T}\} \\
D_{TU} &= \{feature(s,t) | (Grad5_{t-1}^s < 0 \;\&\&\; Grad5_t^s > 0) \\
&\quad || (Grad10_{t-1}^s < 0 \;\&\&\; Grad10_t^s > 0) || (Grad20_{t-1}^s < 0 \;\&\&\; Grad20_t^s > 0), s \in \mathcal{S}, t \in \mathcal{T}\}, \qquad (1)
\end{aligned}
$$

Table 1 Parameters of the local policy of an predictor.

Name	Meaning
b_thres	threshold of bid signal
a_thres	threshold of ask signal
h_duration	period of holding a stock

used technical indicator and D_{TU} is from the pattern of *Turn Up*[†]. These patterns cover roughly 80% of stock price series in the Korean stock market. The predictors, *engine_bear*, *engine_bull*, *engine_GC* and *engine_TU* have been developed from these dataset using artificial neural networks.

After each prediction engine was trained, its performance might be evaluated using several criteria. Many evaluation criteria are summarized in [3] but we have adopted a simulation-based evaluation using, $\mathcal{LP}$, called *local policy*.

Definition 1: Given a space Ω = (B_THRES, A_THRES, H_DURATION), a local policy $\mathcal{LP}$ is a set of optimized parameters for each predictor.

$$\mathcal{LP}(\text{predictor}) = (b\_thres, a\_thres, h\_duration).$$

Table 1 shows the meaning of each parameter. We regard the prediction values of an predictor larger than *b_thres* as bid signals. *a_thres* is the threshold for ask signals and *h_duration* is the maximal duration of holding a stock. Using $\mathcal{LP}$, the trading based on each predictor is simulated as follows. The stocks of which predicted values are larger than *b_thres* are retrieved and each purchased stock is sold when its predicted value becomes lower than *a_thres* or its number of holding days expires. For each predictor, its $\mathcal{LP}$ is greedily optimized over the parameter space. For comparison, two metrics, PPT and accuracy, are used. PPT means average profit ratio per each trade and accuracy is defined as;

$$accuracy = \frac{\#\text{of successful trades}}{\#\text{of recommendations}}, \qquad (2)$$

where successful trade means the trade achieving positive profit after subtracting the transaction cost.

Table 2 summarizes the results. The *engine_normal* corresponds to all-in-one predictor. The accuracy of each predictor is improved 12% more than *engine_normal*. In terms of PPT, the predictors are remarkably superior to *engine_normal*. Every predictor achieves almost more than twice PPT over the *engine_normal*. Although our multi-predictors approach covers only 80% of the whole stocks, this result is somewhat meaningful.

Table 2 Performances of the predictors.

Engine	Accuracy (%)	PPT (%)
engine_normal	57.10	0.79
engine_bear	69.42	1.41
engine_bull	73.37	1.99
engine_GC	71.10	1.72
engine_TU	70.97	1.63

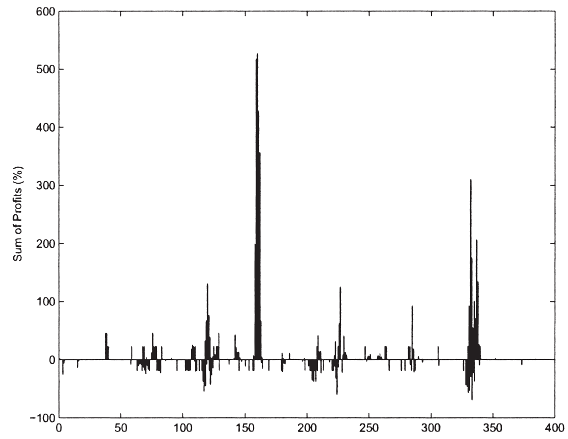


Fig. 1 The tendency of recommendations of *engine_bear*: x-axis represents each month. y-axis represents the total profits induced by recommendations for a day. Up-headed bar is the sum of positive profits and down-headed bar is the sum of negative profits.

3. Need for Meta Policy

We have analyzed the tendencies of our predictors. Figure 1 is the bar graph of the successful or failed recommendations from the trading according to the local policy of *engine_bear*. This figure shows that the profitable recommendations are not equally distributed over the trading days. They are concentrated on some specific periods. It means that profits induced from the trades of a few days dominate others.

To explore the effects of trading with a finite asset, we show how the final asset could be changed according to the purchase money per recommendation. Given an initial asset

[†]For more detailed description about the meaning of Eq. (1), see [4], [8].

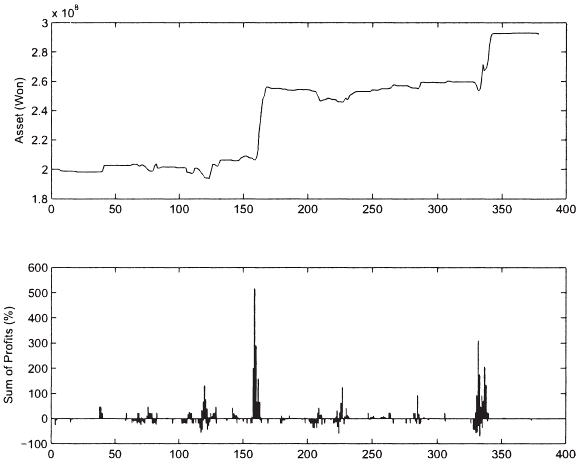


Fig. 2 Funding log and traded recommendations when the purchase money per recommendation is 0.4 million Won.

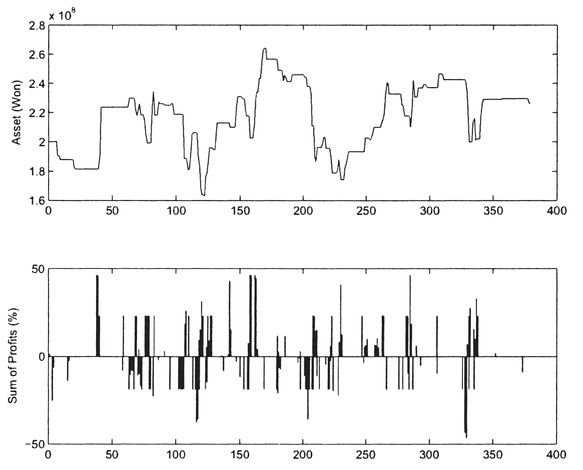


Fig. 3 Funding log and traded recommendations when the purchase money per recommendation is 40 million Won.

200 million Won[††], we set up two tradings one of which purchase money per recommendation is 0.4 million Won and the other 40 million Won. Figure 2 is for the former and Fig. 3 is for the latter. The upper part of each figure shows the history of the asset change and the lower part is the bar graph.

For a small amount of purchase money, we manage to take most of recommendations even when the recommendations are poured out on some days, such as near the 160-th or 340-th trading day. These many profitable trades increase the asset volume. But, when usual days with small number of recommendations, stock fund is much less than ready fund. In those days, though the trade of the recommended stock induces high profit, the asset volume is little influenced by that trade.

In case of the large purchase money per stock, the asset is heavily affected by the result of each trading. The history of the asset in Fig. 3 says the asset is heavily fluctuated according to each trading. Moreover, with the large purchase money the poured out recommendations at the peak days are not manageable, so the bar graph of Fig. 3 looks less profitable.

```
for $t = 1$ to $T$
    for $e = 1$ to $E$
        $S_e$ = retrieve($D_e$, $b\_thres$)
        $N_e$ = numberof($S_e$)
    end for
    $(P_1, \cdots, P_E) = \mathcal{MP}(N_1, \cdots, N_E, \mathcal{SF})$
    for $e = 1$ to $E$
        local_trade($P_e$, $S_e$, $\mathcal{LP}(P_e)$)
    end for
end for
```

Fig. 4 Trading process with a meta policy.

Therefore, it is desirable to adapt the amount of the purchase money for the effective trading. Since we have multiple predictors, we need more complicated asset allocation policy which effectively distributes one finite asset over predictors and their recommendations. Since we want the adaptive asset allocation, we define a *meta policy*:

Definition 2: Let the number of recommendations of multi-predictors be $\mathcal{N} = (N_{bear}, N_{bull}, N_{GC}, N_{TU})$ and the ratio of stock fund over the asset be $\mathcal{SF}$. A *meta policy* $\mathcal{MP}$ is defined as a function over $\mathcal{N}$ and $\mathcal{SF}$,

$$\mathcal{MP}(\mathcal{N}, \mathcal{SF}) := (P_{bear}, P_{bull}, P_{GC}, P_{TU}),$$

where P_{engine} is the proportion of the purchase money of each stock for *engine*.

Figure 4 summarizes the trading process with a meta policy. At t-th day of T trading days, E predictors retrieve their recommendations and $\mathcal{MP}$ determines the purchase money for each predictor of that day according to $N_1, \ldots, N_E$ and $\mathcal{SF}$. The system trades each candidate using $\mathcal{LP}$ of which predictor recommends it.

4. Evolutionary Computation for Meta Policy

Evolutionary computation is probabilistic search method based on the mechanism of natural selection and genetics [2]. Evolutionary computation is a rather systematic iterative procedure consisting of selection and variation procedures. But we must design a chromosome architecture specific to the problem so that intrinsic features of the problem are maximally exploited by the algorithm. Also we must make an artful fitness function as a guide of the search in addition to the objective of the problem. Finally, we provide effective crossover and mutation strategies that searches the problem space using the designed chromosome architecture.

4.1 Representation

Definition 3: A *chromosome* for meta policy is defined as,

$$C = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} & \cdots & r_{1,1280} \\ r_{2,1} & r_{2,2} & r_{2,3} & \cdots & r_{2,1280} \\ r_{3,1} & r_{3,2} & r_{3,3} & \cdots & r_{3,1280} \\ r_{4,1} & r_{4,2} & r_{4,3} & \cdots & r_{4,1280} \end{pmatrix},$$

[††]Won is a monetary unit of Korea.

where a gene $r_{i,j}$ represents the ratio of purchase money per recommendation from the i-th engine when the state is in j-th position.

The number of the recommendations of each engine and the stock fund ratio are the key factors to be considered. Using Table 3 and Table 4, the number of recommendations of each engine and the stock fund ratio are discretized to four bits and to five bits respectively. Total combinations can be exclusively represented by 1280 bits which corresponds to the column length of Definition 3. Each column vector of chromosome C represents the asset allocation rule of four engines.

4.2 Crossover and Mutation

Figure 5 describes how to crossover two chromosomes in this paper. Given parent$_a$ and parent$_b$, the column indices marked as dots are sampled by specified number of η. For parent$_a$, the regions between the marked columns are selected in making genes of the child. For each region, a row index between 0, 1, 2, 3 and 4 is randomly chosen and the selected row is excluded. If selected row index is 0, whole region is accepted such as the rightmost region of parent$_a$. Consequently the black area in the chromosome of child comes from parent$_a$ and the grey area from parent$_b$, which is exclusive of parent$_a$.

In the case of mutation, we randomly choose a j-th column vector. Next each ratio in the column vector is changed by multiplying a random factor between 0.8 and 1.2. If the resulting ratio falls down below a lower limit ϵ, it fixes it to ϵ and if it gets larger than a ρ, it fixes to ρ.

Table 3 Discretization of the number of recommendations.

Number	Discretized Value
0	0001
$[1, 3)$	0010
$[3, 8)$	0100
$[8, \infty)$	1000

Table 4 Discretization of the stock fund ratio.

Stock Fund Ratio (%)	Discretized Value
$[0, 10)$	00001
$[10, 25)$	00010
$[25, 45)$	00100
$[45, 70)$	01000
$[70, \infty)$	10000

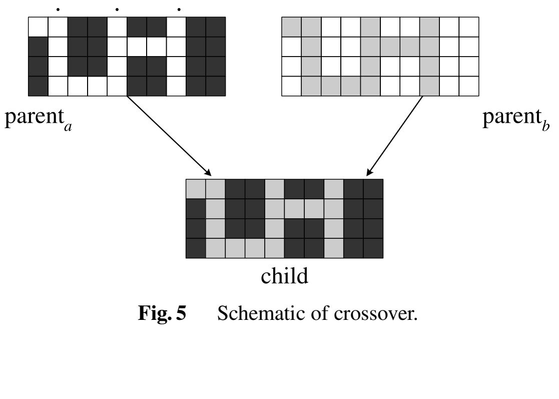


Fig. 5 Schematic of crossover.

4.3 Fitness Function

A big advantage of the evolutionary algorithm is that it has the fitness function which we can actively and flexibly specify the objective to be optimized.

In this paper, we design *fitness function* as

$$fitness(C) = s_1 \times \mathrm{P} - s_2 \times \mathrm{V} - s_3 \times \mathrm{D} \tag{3}$$

where s_1, s_2 and s_3 are the scaling constants of three constituents.

If we assume that there are $1, \ldots, T$ trading days, we can define that P is the *profit ratio*

$$P = 100 * \frac{\mathrm{asset}_T - \mathrm{asset}_0}{\mathrm{asset}_0},$$

where asset$_0$ is initial asset. The relation between asset$_t$ and asset$_{t+1}$ is

$$\mathrm{asset}_{t+1} = \mathrm{trade}(\mathrm{input}_t, \mathrm{asset}_t; C, \mathrm{engines}), \tag{4}$$

where input$_t$ is a sequence of recommendations of the engines on t-th day and asset$_t$ is a sequence including both stock fund and ready fund.

V and D are based on the heuristics of user preference on the performance of the trading system. V is the variance of the profit ratio per month during the trading days. Someone prefers an investment having chances of high return with high risk. But others prefer investments that might produce low return with low risk however they seem more stable than the former.

D is the ratio of the max-draw of the asset [14]. The maxdraw is the worst drawdown the asset. Figure 6 shows what is the maxdraw and D is calculated as

$$\mathrm{D} = \frac{\text{asset at the end of maxdraw}}{\text{asset at the start of maxdraw}}.$$

Larger the ratio is, more unstable and volatile the asset is.

An attractiveness of fitness function is that we can easily modify the objective of the optimization. Although we include just volatility and maxdraw into objective function, other performance measures can be designed and incorporated.

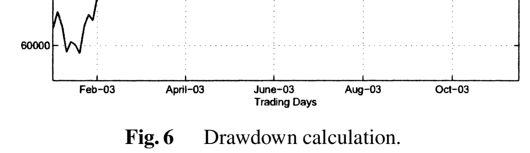


Fig. 6 Drawdown calculation.

5. Experiments

Besides meta policy, named MPG, two other trading systems, namely trader1 and trader2 as shown in Table 5, are used for the performance comparison. Each trading system shares the predictors, which is constructed on the train set from January 1998 to March 2000, but trades on its own asset allocation policy.

5.1 Asset Allocation Policy and Performance of Each System

In case of trader1, as if we had a sub-trading systems per predictor, an initial asset is equally divided into the parts, totally the number of predictors. The stocks recommended by a predictor are traded with the money from the asset for the predictor. During the entire trading period, the divided assets are exclusively utilized to trade by its corresponding predictor. The asset allocation policy of trader1 consists of four constants,

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

where μ_e is the proportion of the asset as purchase money of predictor e. For each predictor the constants are optimized on the period between April 2000 to December 2001.

In trader2, the asset is not divided but managed as united form. Whenever there are any recommendations from predictors, trader2 allocates $1/k_0$ times of asset to purchase, where k_0 is optimized on the same period of trader1.

MPG is constructed by evolutionary computation as described in Sect. 4. It is trained and tuned on the tuning period of trader1 and trader2. η is 20, population size is 500, mutation ratio is 0.1.

Figure 7 shows the comparison of three trading systems on the test period from January 2002 to May 2003. Horizontal axis is the trading day and vertical axis is the asset. Each trading system started with its trading money 25 million Won. The composite index of KOSPI was scaled up so that it looks as if it were 25 million Won at January 2, 2002. Solid line is the history of asset of MPG, dashed-dotted is of trader1, dashed line is of trader2, and dotted line is of composite index.

Clearly, MPG outperforms other two trading systems. After 17 months' trading, its asset rose by nearly 57 million Won, or 230%, to about 82.6 million Won. trader1 shows the worst among three trading systems. There may be some situations where one predictor meets too many recommendations to manage only using its ready fund, while there are much of ready funds in the assets of other predictors.

trader2 also shows limited performance. Since it tries to buy stocks whenever any predictor recommends, it might avoid the ill conditions of trader1. Although it outperforms trader1 over most of periods, trader2 does not consider the relationship between predictors or the ratio of stock fund. During the hard time of August and September 2002, trader1 and trader2 suffer the decline of asset, but MPG endures this hard period. It shows somewhat avoiding the risky declination and getting more chance of level up of the asset using adaptive asset allocation.

Table 5 Policies of the trading systems.

name	description
trader1	Fixed policy, initially partitioned asset
trader2	Fixed policy, united asset
MPG	Adaptive policy, united asset

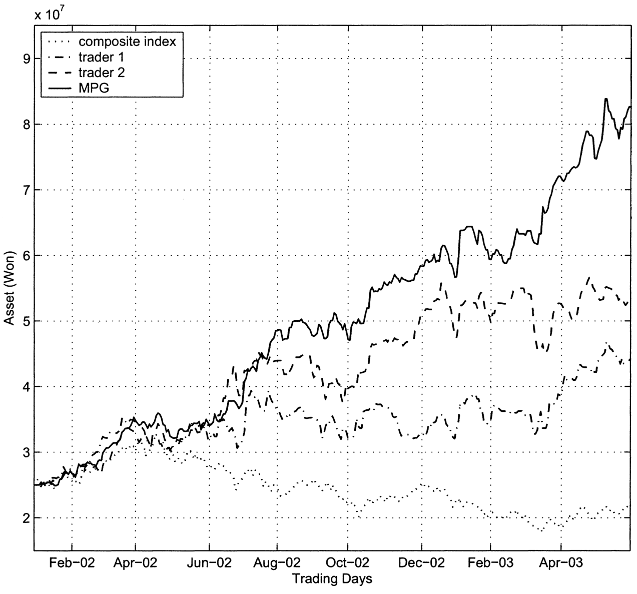


Fig. 7 A comparison of the performances of the trading systems.

6. Discussion

Figure 8 shows a synthetic analysis of MPG including the proportion of stock fund and the information of recommendations. Upper part of the figure is about asset history, middle part is about the proportion of stock fund to asset, and lower part is about the sums of positive or negative profits of each day under the assumption that every recommendation might be purchased. After April 2002, KOSPI is in bear market. From late June 2002 to mid July 2003, there is a sharp rally in KOSPI. *engine_bear* meets its explosive recommending day at June 27, 2002. The asset allocation rule of that day is (3.06, 0.29, 0.2, 0.46), where 3.06 is for *engine_bear*. Stock fund is 5% at June 27, 2002, so we can purchase most of the recommendations. As the result, stock fund is increased over 90% till June 28, 2002.

While KOSPI is in its declining bear at late December 2002, the number of recommendations leading to loss gets increased. At December 20, 2002 *engine_bear* and *engine_bull* recommend 7 and 6 stocks respectively. But their allocation ratios are 0.42 and 0.52. During a few consecutive dangerous days, the small allocation ratios for corresponding predictors prevent the stock fund from being dangerously increased. Although some amount of loss is inevitable, it is more stable than other traders. This situation is more vivid in early March 2003 when `trader2` suffers severe loss of the asset while `MPG` can minimize the fatality.

Figure 9 shows the variations of trading performances according to the change of the scaling constants in the fitness function. Dotted line is the history of asset when $s_1, s_2, s_3 = (1.0, 10.0, 2.0)$. Dashed line is the history on $s_1, s_2, s_3 = (1.0, 0.0, 0.0)$ and Solid line is the history on $s_1, s_2, s_3 = (1.0, 20.0, 5.0)$. The history with no penalty on variance and maxdraw seems to make much money than others. Its profit is always larger than others in most period except May 2003. It suffers a sharp turn in mid May 2003. As we penalize variance and maxdraw, the sharpness in May 2003 gets dull and the volatility gets weak. However as a cost of that stability, the profits in most months might get smaller than an aggressive trading like dashed history.

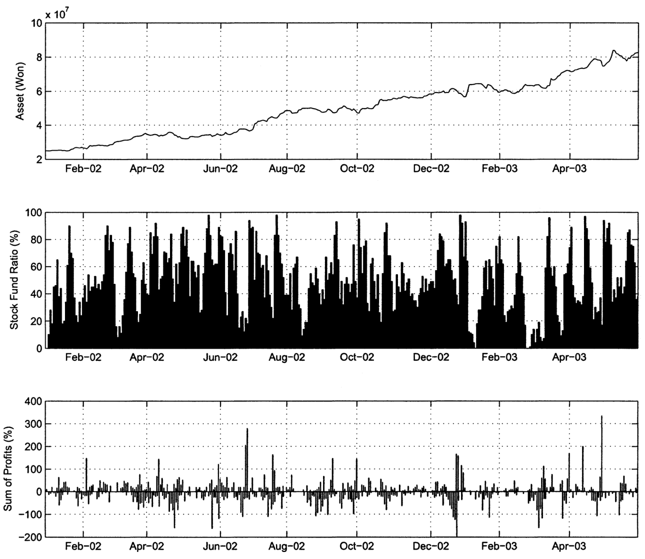


Fig. 8 Upper figure: asset history. vertical axis means Korean monetary unit (Won). Middle figure: The proportion of stock fund to asset as percentage. Lower figure: sums of positive or negative profits recommended a day. It is a virtual summation assuming all the recommendations could be purchased and traded according to local policy of each predictor.

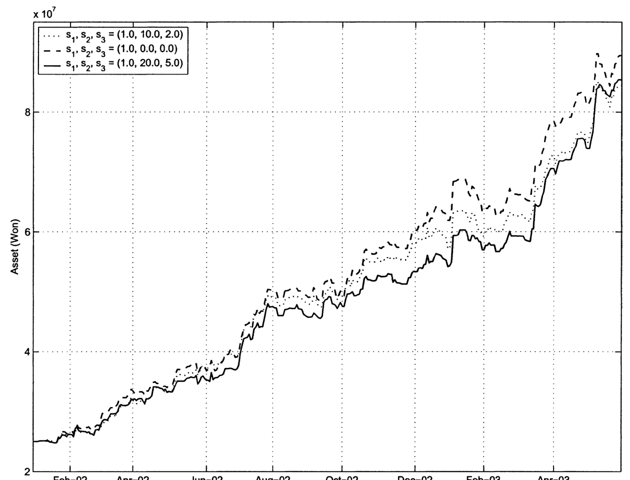


Fig. 9 Performance changes according to variations of coefficients in the fitness function.

7. Conclusion

In this paper, evolutionary computation is used as a methodology to allow a decision maker to combine effectively the learned models of supervised learning. We make an adaptive asset allocator, meta policy, incorporating the information of recommendations of predictors and the stock fund ratio as the part of the chromosome representation.

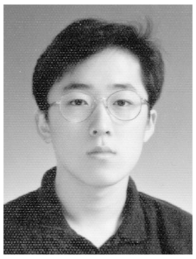
The meta policy optimized by evolutionary computation induces more profits than other fixed asset allocation methods in the simulation on the Korean stock market over specified test period. It can mitigate the loss and maximize the profits by adaptation.

Acknowledgement

This research was supported in part by the Ministry of Education and Human Resources Development under the BK21-IT Program. The RIACT at Seoul National University provides research facilities for this study.

References

[1] E.F. Fama and K.R. French, "Dividend yields and expected stock returns," J. Financial Economics, vol.22, pp.3–26, 1988.
[2] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.
[3] T. Hellström, A Random Walk through the Stock Market, Ph.D. Thesis, Department of Computing Science, Umeå University, 1998.
[4] P.J. Kaufman, The New Commodity Trading Systems and Methods, Wiley, NewYork, 1987.
[5] S.M. Kendall and K. Ord, Time Series, Oxford, New York, 1997.
[6] S.D. Kim, J.W. Lee, J. Lee, and J.-S. Chae, "A two-phase stock trading system using distributional differences," Proc. International Conference on Database and Expert Systems Applications, pp.143–152, 2002.
[7] J.W. Lee and J. O, "A multi-agent Q-learning framework for optimizing stock trading systems," Proc. International Conference on Database and Expert Systems Applications, pp.153–162, 2002.
[8] J.W. Lee, S.D. Kim, J. Lee, and J.-S. Chae, "An intelligent stock trading system based on reinforcement learning," IEICE Trans. Inf. & Syst., vol.E86-D, no.2, pp.296–305, Feb. 2003.
[9] B.G. Malkiel, A Random Walk Down Wall Street, Norton, New York, 1996.
[10] J. Moody and M. Saffell, "Learning to trade via direct reinforcement," IEEE Trans. Neural Netw., vol.12, no.4, pp.875–889, 2001.
[11] R. Neuneier, Risk Sensitive Reinforcement Learning, Advances in Neural Information Processing Systems, pp.1031–1037, MIT Press, Cambridge, 1999.
[12] J. O, J.W. Lee, and B.-T. Zhang, "Stock trading system using reinforcement learning with cooperative agents," Proc. International Conference on Machine Learning, pp.451–458, Morgan Kaufmann, 2002.
[13] E.W. Saad, D.V. Prokhorov, and D.C. Wunsch II, "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks," IEEE Trans. Neural Netw., vol.9, no.6, pp.1456–1470, 1998.
[14] J.S. Zirilli, Financial Prediction Using Neural Networks, Thompson Computer Press, 1997.



Jangmin O is a Ph.D. student of the School of Computer Science and Engineering at Seoul National University. He received BS and MS degrees in the School of Computer Science and Engineering, Seoul National University in 1997 and 1999. His research interests include computational finance, optimization, reinforcement learning and dynamic graphical models.

Jongwoo Lee is an Assistant Professor of Dept. of Multimedia Science at Sookmyung Women's University in Seoul, Korea since 2004. He received his B.S., M.S., and Ph.D. degree in Computer Engineering from Seoul National University in 1990, 1992, and 1996 respectively. From 1996 to 1999, he worked for Hyundai Electronics Industries, Co. He was an Assistant Professor of Division of Information and Telecommunication Engineering at Hallym University in Choончheon, Korea from 1999 to 2002. And then he moved to an Assistant Professor of Computer Engineering at Kwangwoon University in Seoul, Korea from 2002 to 2003. His research interests include storage systems, computational finance, cluster computing, parallel and distributed systems, and system software.

Jae Won Lee has been a full-time Instructor of the School of Computer Science and Information at Sungshin Women's University in Seoul, Korea since 1999. He received his BS, MS, and Ph.D. degrees in Computer Engineering from Seoul National University in 1990, 1992, and 1998. His current research interests include computational finance, artificial intelligence, machine learning, natural langauge processing, and computer music.

Byoung-Tak Zhang has been an Associate Professor of School of Computer Science and Engineering at Seoul National University. He received his BS and MS degrees in Computer Engineering from Seoul National University in 1986 and 1988, and a Ph.D. degree in Computer Science from University of Bonn, Germany in 1992. He had been a research associate at German National Research Center for Information Technology from 1992 to 1995. He serves an Associate Editor of IEEE Transactions on Evolutionary Computation. His research interests are in learning and adaptive systems, evolutionary computation, probabilistic neural networks, and DNA computing.