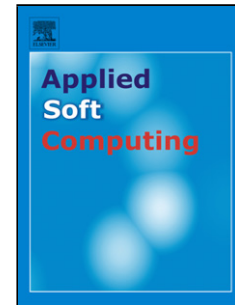


Accepted Manuscript

Title: A League Championship Algorithm Equipped with Network Structure and Backward Q-learning for Extracting Stock Trading Rules

Authors: Muhammad Reza Alimoradi, Ali Hussein-zadeh Kashan



PII: S1568-4946(18)30184-4
DOI: <https://doi.org/10.1016/j.asoc.2018.03.051>
Reference: ASOC 4802

To appear in: *Applied Soft Computing*

Received date: 23-7-2017
Revised date: 7-3-2018
Accepted date: 30-3-2018

Please cite this article as: Muhammad Reza Alimoradi, Ali Hussein-zadeh Kashan, A League Championship Algorithm Equipped with Network Structure and Backward Q-learning for Extracting Stock Trading Rules, *Applied Soft Computing Journal* <https://doi.org/10.1016/j.asoc.2018.03.051>

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

A League Championship Algorithm Equipped with Network Structure and Backward Q-learning for Extracting Stock Trading Rules

Muhammad Reza Alimoradi

Department of Industrial and Systems Engineering, Tarbiat Modares University, Tehran, Iran.

Email: m.alimoradi@modares.ac.ir

Ali Husseinzadeh Kashan*

Department of Industrial and Systems Engineering, Tarbiat Modares University, Tehran, Iran.

Email: a.kashan@modares.ac.ir

*Corresponding Author: Email: a.kashan@modares.ac.ir

Tel: +98 21 82884398

Fax: +98 21 82884392

highlights

- A league championship algorithm is heavily adapted to extract stock trading rules
- A network structure representation scheme and a backward Q-learning is developed
- Each individual learns from strengths/weakness of others to extract better rules
- An average matching degree approach is introduced to create trinary signals
- The hybrid method shows a very good performance specially for sharp price uptrends

Abstract

Buying and selling stocks are the main activities of investment in the financial markets. During several years, financial experts have designed various indices and indicators to develop intelligent methods to decide whether buy or sell or no trade a specific stock. In this paper, the league championship algorithm (LCA) is adapted and equipped with a network structure (which provides the implicit memory function, compact structure and gives the ability of reusing nodes) for stock trading rule extraction process. The proposed algorithm is able to extract and save various stock trading rules for various kinds of stock market conditions. Each individual in LCA makes role as a sport team which could learn from weaknesses and strengths of others to enhance its

performance. The network structure used for each individual is equipped with reinforcement learning to strength the searching power and extracting better rules. The backward Q-learning (BQL), which is a combination of Sarsa learning and Q-learning algorithms, is also hybridized with LCA to increase the learning speed to extract better rules. A trend clustering technique is also developed to determine uptrends and downtrends and to remedy delay in the trend direction reflection along with moving average of stock prices. Finally, to use the trading rules extracted and saved in the rule pools, a new approach through average matching degree is introduced which could create trinary signals of buying, selling, and no trade. The effectiveness of the proposed approach is tested on a set of 20 companies, which have high liquidity and large market capitalization, chosen from various fields of Tehran Stock Exchange. The used historical price data was adjusted through factors of dividends and capital increase. Computational tests and simulations indicate that our hybrid method clearly offers higher profits and lower losses than those of genetic network programming method with rule accumulation (GNP-RA) on all stocks and those of the Buy&Hold strategy on 17 out of 20 instances (85%), especially when the price uptrends are sharp.

Keywords: Stock trading rule extraction, Technical analysis, Reinforcement learning, Q-learning, Genetic network programming, League championship algorithm.

1. Introduction

The ideal strategy for all stock investors is to buy at a low price and selling at a higher price. It is hard and usually impossible to obtain the ideal strategy, because the stock price is a function of known and unknown factors and almost seems to behave unpredictably. Generally, two approaches, used for predicting stock prices and determining the timing of buying and selling stock, are Fundamental analysis and Technical analysis. Fundamental analysis wants to determine the intrinsic value of a security such as a stock based on a set of economic and noneconomic factors. On the other hand, Technical analysis studies market actions via the use of charts to predict future price trends [1]. Technical analysts assume that historical price patterns would repeat in the future, therefore these patterns are able to forecast the price movement. Both methods aim to forecast the price movement from different perspectives.

Contrary to stock market participants, a group of academic researchers has criticized the profitability of Technical analysis. The criticisms have been based on two factors, efficient market hypothesis, and random walk theory. Efficient market hypothesis states that “the price always reflects all available information.” This hypothesis suggests that any attempt at gaining profit with available information is vain [2]. Cootner [3] showed that the stock price behavior is a random walk and therefore market fluctuations cannot be predicted based on pre-designed rules. People who are one of the important components of a market are omitted from random walk theory. People remember the previous prices and act accordingly. Prices are seen to understand people’s reactions, but prices also affect people’s reactions [4].

Regardless of the results of initial studies [2, 3, 5-8] on applying the technical trading rules and explaining the ineffectiveness of the technical trading rules, traders continued to use these rules until 1990s when a new trend of research on the application of technical trading rules were carried out, which their results rejected the related results of previous studies about the ineffectiveness of technical rules [9-13].

After preliminary investigations which used simple and traditional methods to exploit trading rules in the late of 1990s, new groups of studies emerged and used more advanced computational methods to extract trading rules. Bauer [14] used genetic algorithm to automatically extract technical trading rules and investigated them in the US exchange market. His results showed positive excess returns. Neely, Weller [15] used genetic programming to identify optimal trading rules. They extracted trading rules as decision trees and investigated them on six currencies. Neely and Weller [16] used genetic programming to extract trading rules for four European Monetary System currencies. They could identify and examined profitable technical trading rules for three of four currencies.

In recent years, researchers have been more interested in studying soft computing techniques such as genetic algorithm [17, 18], genetic programming [19, 20], and artificial neural network [21, 22] to predict stock price or to trade stock.

For the first time, Izumi, Yamaguchi [23] used genetic network programming (GNP) to extract stock trading rules. They employed candlestick chart patterns and exploited stock trading rules based on them. Their results represented superiority of their proposed model over the three layers Neural Network and the Multi-Branch Neural Network. Chen, Mabu [24] enhanced the model presented by Izumi, Yamaguchi [23], with adding Sarsa learning as one of reinforcement learning methods. They introduced sub-nodes into each node of the network structure without any change in the network size. They applied not only candlestick charts but also a group of technical indicators as input variables of their model in order to get more effective stock price information from the stock market. Based on their results, it was clarified that their model obtains more profits than GNP-Actor Critic, GNP candlestick chart, GA, and Buy&Hold.

Mabu, Hirasawa [25] created a structural change in Chen, Mabu [24]'s model and proposed genetic network programming with rule accumulation (GNP-RA) for extracting trading rules. They believed that a model with rule accumulation which extracted and saved a lot of rules can understand and interpret a greater variety of events and conditions and therefore had a higher ability in decision-making than models without rule accumulation. Their results revealed that GNP-RA acts better than GNP without rule accumulation and Buy&Hold strategy.

Mabu, Obayashi [26] utilized ensemble learning, a process in which several classifiers or experts are strategically generated and combined to solve a special computational intelligence problem. They combined Multi-Layer Perceptron (MLP) and GNP-RA based on Sarsa learning in order to enhance the performance of the stock trading system. They divided the whole training data into several sub-periods through the market trends. Stock trading rules were extracted for each sub-period separately and saved in rule pools related to that sub-period. Technical indicators and candlestick chart patterns which belong to each sub-period were used to learn MLP where, the

outputs of MLP were the similarity values to each rule pool set (trend). In the testing period, the rule sets, based on the similarity calculation between testing and training data determined by MLP, are selected and used for decision-making. They compared their results with the results of GNP-RA and Buy&Hold and concluded that their model has a better performance than the other two methods.

One of the requirements to deal in the capital market is that each trader in a market requires a trading plan which should be written and acted. Emotions are the significant challenges for traders which cause the traders ignore their trading rules and do something such as not to close their long position when the price falls, to deal at an unsuitable time, and to do an unplanned deal which is against their trading rules which enforce they lose their money. Thus, creating a trading system which is able to deal automatically and away from emotions can be important for investors.

The purpose of this paper is to propose a rule-based trading system based on an enhanced league championship algorithm (LCA) to determine the suitable time of buying and selling stocks. LCA is adapted such that each artificial team (essentially an individual in terms of evolutionary computation) is represented by a network structure and considered as an evolving rule generator. The resulting algorithm is called LCA-N. The most important difference between LCA-N and GNP is that each individual of LCA-N could learn something from weaknesses and strengths of other individuals to enhance its performance. But in GNP, an individual is randomly altered or combined with others, thus, its performance may improve or not. The major advantage of LCA-N over probabilistic model building genetic network programming (PMBGNP) [27, 28] is that there is no need to estimate a joint probability distribution associated with the database of selected individuals from the previous generation which is very complex and computationally expensive. Reinforcement learning is applied to LCA-N to enhance the intensified search and online learning abilities. The backward Q-learning is applied to increase the speed of learning and to obtain a better sequence of actions. However, the most important point is to extract a large number of stock trading rules by an exploration ability of reinforcement learning with ϵ -greedy policy. An artificial team in LCA-N with backward Q-learning (LCA-NBQL) has a number of sub-nodes in each node of the network structure. Under the network structure representation, reinforcement learning (RL) which chooses sub-nodes based on search experience, is used to select technical indices or to make buying or selling decisions. The proposed rule-based trading system is composed of two steps. At the first step (the training period), LCA-NBQL is used to extract a large number of stock trading rules combining technical indices and accordingly, suitable extracted rules are stored in different rule pools based on their result and the price trend. The trend direction is usually determined by using a moving average of stock prices. A moving average showing the average price of a stock over a certain time period is a function of price and market trends. Thus, it has a delay for showing trend direction. So a trend clustering technique is used to determine uptrend and downtrend. At the second step (the testing period), to make a decision on buying or selling stocks, a new approach which could create trinary signals of buying, selling, and no trade, is introduced based on the average matching degree using the stored rules and current day information. To confirm the effectiveness of our proposed method, we compare the results obtained by LCA-NBQL, based on extensive computational efforts, with those obtained by GNP-RA and Buy&Hold methods.

The paper is organized as follows. The proposed league championship algorithm, the adapted backward Q-learning, and the manner by which the market trend is determined via trend clustering technique are explained in Section 2. Answers to questions such as how to extract trading rules, how to store the rules in the rule pools and how to decide based on the stored rules are all provided in Section 3. Section 4 deals on the effectiveness of the proposed approach, wherein the simulation environment of stock trading is explained and simulation results are analyzed and compared with existing stock trading strategies. Section 5 concludes the paper.

2. The LCA-NBQL method: empowering LCA with network representation and backward Q-learning

LCA is a population-based algorithmic framework, inspired by sport league competitions in the real world, for global optimization over a continuous search space. LCA was introduced first by Husseinzadeh Kashan [29] for numerical global optimization. During the iterations of the algorithm, different solutions which can be given to a problem, are compared and improved based on their strength.

LCA tries to imitate a championship environment wherein individuals (or teams in the sport terms) compete in an artificial league for several weeks (iterations). Given the league schedule in each week, individuals compete in pairs and their competition outcome is determined in terms of win-loss-tie, given the playing strength (fitness value) along with each individual. Before proceeding the next iteration, each individual is changed to yield an offspring by imitating an artificial post-match SWOT analysis, and required changes are made in different dimensions of the individual array. In this way, the championship goes on for a number of seasons (stopping condition). Beside the nature, culture, politics, etc. as the typical sources of inspiration of various algorithms, the metaphor of sporting competitions is used for the first time in LCA. Since its introduction in 2009, LCA has been motivated by several sport inspired algorithms. For a greater details on LCA and its different modules, the interested reader may refer to Husseinzadeh Kashan [30], Husseinzadeh Kashan [31]. Through the paper, we may use the following terminologies alternately when describing LCA-NBQL algorithm. “League” stands for the population; “team” which is considered as an evolving rule generator, stands for an individual in the population; each individual/team in the population generates a number of rules in each iteration. The “rule pools” are therefore the solutions; “Team formation” stands for a graph structure related to an individual; “week” stands for “iteration”; “playing strength” stands for the “fitness value”.

In this paper, instead of the array representation typically used in LCA for the individuals, we adopt a network structure for the reasons that it provides a compact structure, enables to reuse nodes, and has an implicit memory function of the past actions. Such structural characteristics are useful to deal with dynamic environments such as stock markets. The network structure is equipped with reinforcement learning in order to enhance searching ability and extracting better rules. The rule accumulation method is also applied to enable the model to percept and interpret more different events and conditions. Such considerations call for relatively massive changes made in the traditional LCA.

2.1. The network structure of LCA individuals

LCA is adapted to extract stock trading rules via developing a network structure for individual (team) representation. Fig. 1 shows the basic structure of an individual. The network structure consists of a start node and a collection of judgment and processing nodes whose number is determined in advance based on the complexity of the problem. Each judgment node has an if-then branch decision function and each processing node has an action function.

Each of the judgment nodes includes a technical index as a decision function, each of the processing nodes determines buying or selling actions, and the function of the start node is to choose the first node to be executed. The node transition in the network begins from the start node and thereafter, a chain of judgment and processing nodes are executed according to the connection among them.

Fig. 2 shows the structure of a node in the network. Each node has some attribute such as the node type, the node function and the node connection to other nodes.

In Fig. 2, NT_i represents the node type for node i , 0 is allocated to the start node, 1 is allocated to the judgment nodes, and 2 is allocated to the processing nodes which provide the final decision. d_i is the time delay spent on making judgment or process at node i . In this paper, following Chen, Mabu [24], Mabu, Hirasawa [25], Mabu, Obayashi [26], Chen and Wang [32], for judgment nodes we set $d_i=1$ and for the processing nodes we set $d_i=5$. Each action step is 5 judgment nodes or less than 5 judgment nodes and a processing node which is executed in a trading day. Q_{i1} and Q_{i2} are associated with reinforcement learning and estimate the sum of the discounted rewards obtained in the future due to the use of each sub-node i . ID_{i1} and ID_{i2} determine the node function of each sub-node i . For example, if node i is a judgment node and $ID_{i1} = 2$, it implies that the function of sub-node 1 of node i is J_2 . Each technical index assigned to each judgment node is listed in the library and recalled by the network with a code which is ID_{ij} . $C_{i1}^A, C_{i1}^B, C_{i2}^A$ and C_{i2}^B are numerical values which represent the nodes to whom node i is connected. Superscripts A and B correspond to judgment results.

2-2. Sub-nodes in the judgment and processing node

Because of the use of reinforcement learning, each judgment and processing node in the network has some sub-nodes which have their own Q value and function. These sub-nodes are added to the network in order to enhance its performance and also to accelerate convergence. The number of sub-nodes is bounded by two, because numerous sub-nodes would enlarge the search space and therefore reduce the search speed [24]. Fig. 3 shows the structure of the nodes of the network equipped with reinforcement learning. Fig. 3(a) shows the structure of a judgment node with 2 sub-nodes. ID_{i1} and ID_{i2} are technical indices and Q_{i1} and Q_{i2} are Q values for sub-node 1 and sub-node 2, respectively. Each sub-node of a judgment node is connected to two other nodes. Fig. 3(b) represents the structure of a processing node with 2 sub-nodes. ID_{i1} and ID_{i2} are buying and selling actions and Q_{i1} and Q_{i2} are Q values for sub-node 1 and sub-node 2, respectively. Each sub-node of a processing node is connected to a judgment node.

2.3. Backward Q-learning

The Q-learning algorithm introduced by Watkins and Dayan [33] is one of the simplest and most well-known reinforcement learning algorithm. One-step Q-learning which is the simplest form of Q-learning is defined by Eq. 1.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R(s_t) + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (1)$$

Where $Q(s_t, a_t)$ indicates the Q value after applying action a_t in state s_t . Subscript t indicates the previous period and subscript $t+1$ indicates the current period. $\alpha \in (0,1]$ is a learning rate which shows the magnitude of the effect of future periods. $R(s_t)$ is the immediate reward received from environment through taking a_t in s_t at time t . $\gamma \in (0,1]$ is the discount rate.

The Sarsa learning algorithm introduced by Sutton [34] and Sutton and Barto [35], is an on-policy algorithm and an improved version of Q-learning algorithm. In this method, the value of the $Q(s_t, a_t)$ is updated by Eq. 2.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R(s_t) + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (2)$$

Sutton and Barto [35] and Wiering and Van Hasselt [36] stated that on-policy method guarantees the convergence, while Boyan and Moore [37] and Gordon [38] showed that Q-learning might be divergent. Although, Sarsa has a faster convergence characteristic but it may trap into local minima. Q-learning exhibits a better final performance but usually requires a longer time to learn [39].

With respect to the mentioned points, the Backward Q-Learning [39], which is a combination of Sarsa and Q-learning, is used to enhance the learning speed and improve the rule extraction performance. The architecture of the standard reinforcement learning and the architecture of the reinforcement learning integrated with backward Q-learning has been shown in Fig. 4 and Fig. 5, respectively. While an agent interacts with its environment, the agent gains knowledge. The more the agent interacts with the environment, the more accurate knowledge the agent acquires. The agent can use this knowledge to enhance learning speed and to balance exploration-exploitation. Following the true experience of the agent to update the Q values, Sarsa learning calculates Q values when an agent goes through states, elects actions, and acquires rewards. The agent simultaneously records previous state, action, reward, and current state as information of the previous step. When the agent reaches the goal state, the agent employs the recorded information of steps to update Q values by Q-learning again. It is not an ordinary Q-learning and is backward updated, thus this called backward Q-learning [39].

2.3.1. Node transition rule and its reinforcement learning

According to Sarsa learning, node transitions are carried out in the following way. Passing the start node, a judgment node in connection with the start node is selected. One of its sub-nodes is chosen based on ϵ -greedy policy. For example, a sub-node with the largest Q value is selected with probability of $1-\epsilon$ or one of sub-nodes is selected randomly with probability of ϵ . Then, the corresponding node function is executed and based on the execution results and the selected connection, the current node is passed towards the next node. When the current node is a

processing node, one of its sub-nodes is selected same as the case of a judgment node. The action of the selected sub-node is executed and the next node is determined based on the selected connection. The Q value is updated by Eq. 3 after each node transition.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R(s_t) + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (3)$$

Where $Q(s_t, a_t)$ shows the Q value of the selected sub-node a_t at node s_t . $R(s_t)$ is a reward obtained after executing node s_t .

2.3.2. Required changes to conform the backward Q-learning (BQL) with trading rule extraction problem

Given that the aim of trading is whether to buy or sell stocks, we should devise for some changes in BQL. The forward node transition in our proposed LCA-NBQL approach has been already described in the previous section. If the portfolio contains no stock and a rule with buying action was generated, all states (s_t), actions (a_t) and rewards are recorded in order to update Q values in the backward learning after the last previous processing node, until a rule with selling action being generated. To explain in more detail, we assume that the portfolio contains no stock and a rule with buying action has been generated at the current state (s_{t-n+4}). s_{t-n} is defined as the first state after the previous processing node and s_{t+1} is defined as a state that a rule with selling action will be generated. Therefore, each node transition from s_{t-n} until s_{t+1} is recorded as a four-dimensional vector like $(s_t^i, a_t^i, R_t^i, s_{t+1}^i)$ in M^i , where that superscript i is the node transition number. The Q value will be backward updated based on the information stocked in M^i as follow:

$$Q(s_t^i, a_t^i) \leftarrow Q(s_t^i, a_t^i) + \alpha_b[R^i(s_t) + \gamma_b \max_a Q(s_{t+1}^i, a_{t+1}^i) - Q(s_t^i, a_t^i)] \quad (4)$$

Where $i=n, n-1, \dots, 1$. α_b and γ_b are the learning rate and discount factor for BQL. If one action step contains 5 judgment nodes to update Q value in backward learning, a punishment is considered for the fifth judgment node.

It is worth to mention that BQL is used for each set of trades, i.e. one - buy and one sell in the current week. For example, if a rule with buying action has been generated and then a rule with selling action has not been generated in the current week, BQL has not been used in this week. Fig. 6 shows how BQL is used in one set of trades.

Now we have finished descriptions on the BQL mechanism of the LCA-NBQL method. In what follows we give a detailed description on different modules of LCA part, adapted to our problem, namely generating the league schedule, identifying the winner/loser individuals and setting up a new team formation.

2.4. Generating the league schedule

To generate the league schedule, at first it is assumed that both teams only played once with each other in every season. By “team” we addressed an individual in the population. Through the paper, both terminologies are alternately used. For a sport league composed of L teams, each team has $L-1$ competitions and it means that totally $L(L-1)/2$ matches are done during a season. The scheduling algorithm is simple and illustrated with a league of 8 teams ($L = 8$). Let assign a number to each team and pair them off in the first week (Fig. 7(a)). As shown in Fig. 7(a), team 1 plays with 8 and team 2 plays with 7 and so on. For the second week, one team is held, i.e. team 1, and the others are rotated clockwise (Fig. 7(b)). In this week, team 1 plays with 7, team 8 plays with 6 and so on. This procedure is continued until the $7^{th}(L-1)$ week. If the number of teams (L) was odd, a dummy team is added. The rival of the dummy team has no play and gets rests in that week.

The league schedule can have an important role on the exploration and exploitation ability of the LCA. Conducting an artificial game every time between each individual and high-quality solutions may improve the exploitation ability. However, the exploration ability is also important to avoid the search getting stocked. In this way in LCA we allow that all individuals have a chance to play with each other under the single round-robin rational. In this way each individual will experience the game with both high and less quality solutions to learn from both getting close to high quality solutions (exploitation) or retreat from less quality solutions (exploration).

2.5. Identifying the winner/loser individual

The winner of a match is determined based on comparing a random number with value of a function which has a linear relation with fitness strength of competing teams. Let's assume, individual i played with j at week t . The chance of team i to beat team j at week t is defined as p_i^t and is calculated by Eq. 5 (p_j^t can be defined accordingly). A random number in $[0, 1]$ is generated. If it was less than p_i^t , individual i is announced as winner and otherwise it is announced as loser. Let also \hat{f} be an ideal value (a lower bound on the fitness value for a minimization problem) till that moment and f_i^t and f_j^t be the fitness value of individuals i and j , p_i^t could be calculated with Eq. 5.

$$p_i^t = \frac{f_i^t - \hat{f}}{f_i^t + f_j^t - 2\hat{f}} \quad (5)$$

2.6. Setting up a new team formation

To generate a new individual from the currently available individuals in the population, LCA simulates an artificial post-match analysis followed by coaches. The simulation is based on SWOT analysis. Analyzing performance of a team in previous week (in week/iteration t) is studied as internal evaluation (strengths/weaknesses) and analyzing performance of its rival which plays with it the next week ($t+1$) is studied as external evaluation (opportunities/threats). The details of how to generate a new individual solution for iteration $t+1$, from the population at iteration t for numerical optimization can be found in Husseinzadeh Kashan [30], Husseinzadeh Kashan [31] and Husseinzadeh Kashan and Karimi [40]. In these papers, authors have proposed four updating equations (namely, S/T, S/O, W/T and W/O equation) following the SWOT rational. However, as stated earlier, unlike array representation used in conventional LCA, for stock trading rule extraction problem we use the network structure as our representation scheme.

In Modeling an artificial match analysis for team i to create a new team formation for week $t+1$, if team i has won the team j in week t , we suppose that this success (failure) has been obtained because of the strengths (weaknesses) of team i and consequently the weaknesses (strengths) of team j directly. Now, based on the league schedule in week $t+1$, let us suppose that the next match of the team i is with the team l . If team l had been the winner (loser) over team k in week t , so this success (failure) and its relevant team formation maybe a direct threat (opportunity) for the team i . This success (failure) has been obtained by strengths (weaknesses). Focusing on strengths (weaknesses) of the team l , give team i an intuitive method to avoid potential threats (to gain possible opportunity advantages). Alternately, team i can focus on the weaknesses (strengths) of the team k . Based on the previous week events, the possible action for the team i is gained from the artificial match analysis summarized in SWOT matrix in Table 1.

Learning from strength of the winner team: let the number of action steps be m in each week, so an integer n among $\{1, \dots, m\}$ and a vector $(step_{m_1}, \dots, step_{m_n})$ which is a vector of action steps executed by LCA-NBQL, associated to the team i in week t , are chosen randomly. The vector $(step_{m_1}, \dots, step_{m_n})$ is a vector of steps which are used by team i in order to remove its weaknesses by analogous steps of a winner team. For each step m_p ($step_{m_p}$), $p = 1, \dots, m$, an integer v_{m_p} is chosen randomly given that $0 < v_{m_p} \leq \min(length_{step_{m_p}}^i, length_{step_{m_p}}^{winner})$ and is defined as the number of changes made in $step_{m_p}$. v_{m_p} nodes are randomly chosen among nodes which are available in $step_{m_p}$ of team i . Two cases would happen for each picked node. First, the node function of the picked node does not exist among the node functions in analogous step of the winner team. In this case, a node is randomly chosen among nodes in $step_{m_p}$ of the winner team and its node function is settled in the node function of the picked node of the $step_{m_p}$ of team i . When choosing the nodes in $step_{m_p}$, repetition is not allowed. In this case, when the node function

has been transferred from the winner team to team i , the Q value of the picked node of $step_{m_p}$ of team i is determined as follow:

$$Q_i^{selected} = \max\{Q_i^{selected}, Q_{winner}^{selected}\} \quad (6)$$

In the second case, the node function of the picked node exists among the node functions in analogous step of the winner team. In this case, the picked node learns connections from the chosen node of $step_{m_p}$ of the winner team. In Fig. 8, an example of learning in $step_{m_p}$ of team i with respect to $step_{m_p}$ of the winner team has been shown. Procedure of learning from the fortes of the winner team has been shown in Fig. 9.

Learning from weaknesses of the loser team: let the number of action steps be m in each week, so an integer n among integers $\{1, \dots, m\}$ and a vector $(step_{m_1}, \dots, step_{m_n})$ which is a vector of action steps executed by LCA-NBQL, associated to the team i in week t , are randomly chosen. The vector $(step_{m_1}, \dots, step_{m_n})$ is a vector of steps which are used by team i in order to remove its weaknesses through analogous steps of a loser team. For each step m_p ($step_{m_p}$), $p = 1, \dots, m$, an integer v_{m_p} is randomly chosen given that $0 < v_{m_p} \leq \min(length_{step_{m_p}}^i, length_{step_{m_p}}^{loser})$ and is defined as the number of changes made in each $step_{m_p}$. Nodes from $step_{m_p}$ of team i which have as common node functions as nodes in analogous step of loser team are chosen and their corresponding node functions are replaced with other functions from the library. The maximum number of chosen nodes is v_{m_p} . When choosing the nodes in $step_{m_p}$, repetition is not allowed. So, the weaknesses of the loser team are resolved from the formation of team i . If the number of changes in $step_{m_p}$ is less than v_{m_p} , the size of the remaining changes, nodes are chosen among nodes in $step_{m_p}$ which have not been chosen in the current week and their corresponding connections are changed with probability of p_q . In Fig. 10, an example of learning in $step_{m_p}$ of team i with respect to $step_{m_p}$ of the loser team has been shown. Procedure of learning from the weaknesses of the loser team has been shown in Fig. 11.

The best team formation for the i^{th} member of the population (i.e., team i) till week t is defined as a team formation having the highest fitness value till week t . To create a new team formation for team i in week $t+1$, we consider the best team formation of team i till week t and do the required changes based on learning from strengths of the winner team or learning from weaknesses of the loser team.

Let us introduce the following indices:

l = Index of the team that will play with team i ($i = 1, \dots, L$) at week $t+1$ based on the league schedule.

j = Index of the team that has played with team i ($i = 1, \dots, L$) at week t based on the league schedule.

k = Index of the team that has played with team l at week t based on the league schedule.

The four possible cases are as follow:

1. Both team i and team l have won their competition against team j and team k . Here, first team i learns from team j via “learning from weaknesses of team j ” and its relevant team formation changes. Then, team i enhances its new team formation through “learning from weaknesses of the team k ”.
2. Team i has won and the team l has lost its competition against team j and team k . Here, first team i learns from team j via “learning from weaknesses of team j ” and its relevant team formation changes. Then, team i enhances its new team formation via “learning from strengths of team k ”.
3. Team i has lost and the team l has won its competition against team j and team k . Here, first team i learns from team j via “learning from strength of team j ” and its relevant team formation changes. Then, team i enhances its new team formation via “learning from weaknesses of team k ”.
4. Both team i and team l have lost their competition against team j and team k . Here, first team i learns from team j via “learning from strength of team j ” and its relevant team formation changes. Then, team i enhances its new team formation through “learning from strength of the team k ”.

2.7. Trend determination

A trend is the general direction of a market or the price of an asset. Moving averages are defined as the average price of a security over a certain time period. So, moving averages represent the consensus views and expectations of shareholder over a period of time in the past. Moving averages are trend-following or lagging indicators because they are based on past prices and do not predict price direction. The most common application of moving averages is to identify the trend direction. A moving average is a function of price and lags behind the price. So, the idea of stock price clustering introduced by Wu, Chou [41] is used for determining the trend in order to eliminate the lag. Stock prices are clustered via employing a standard genetic algorithm and using the concept of trend value and rate of change which are explained more in the following. The stock prices are clustered in uptrends and downtrends. Trend value is the difference between price and moving average and is calculated by Eq. 7.

$$T_n = \frac{P_n - MA_n}{MA_n} \quad (7)$$

Where p_n is calculated by Eq. 8.

$$P_n = \frac{O_n + H_n + L_n + C_n}{4} \quad (8)$$

Where, p_n is stock price average in n^{th} day, MA_n is the moving average in n^{th} day, O_n, H_n, L_n and C_n are opening price, the highest price, the lowest price and the last price in n^{th} day, respectively. The shorter the length of a moving average, the more unstable the price direction and

the more difficult it is to realize the trend. In contrast, the longer the length of a moving average, the more stable the price direction. It, however, is less sensitive to the short-term trend action. To calculate the trend value in this paper, similar to Mabu, Hirasawa [25], a 25-day moving average, which nearly exhibits the monthly movements of the price, is used.

The rate of change is applied to mathematically describe the percentage of the change in value over a specified time frame which can be calculated by Eq. 9.

$$\text{Rate of change} = \frac{\text{current price} - \text{previous price}}{\text{previous price}} \quad (9)$$

The time frame here is a trading day just similar to Wu, Chou [41]. The volume theory states that the greater volume, the greater force behind the move. As long as volume increases, the current price trend should continue. If volume declines as price trend progresses, there is less reason to believe that the trend will continue [4]. The volume can be applied to measure the force of price movement. Therefore, in the remainder of the paper, the volume will be considered as one of criterion for the stock price clustering.

3. Extracting and accumulating trading rules

Fig. 12 shows the flowchart of rule extraction in training period. First, teams are randomly created. Then each team enters to training period and its fitness value is calculated based on its performance. Given that good rules are extracted from the best team every week, in addition to the extracted rules from the best team having the highest fitness value, the best rules extracted by the second best team are also stored. During iterations of LCA-NBQL algorithm, teams play in pair based on the league schedule and their team formation changes by which new rules are extracted. This procedure is repeated until the last iteration and rule accumulation is stopped when the training period ends.

We use the accumulation method introduced by Wang, Mabu [42]. We store the extracted rules from the best team and a number of the best rules extracted by the second best team in each iteration in rule pools. If an extracted rule has not been stored in any rule pool, in previous iterations, it would be stored in rule pools of the current iteration. Let $\{X_1, X_2, X_3, \dots\}$ be the set of input attributes and $\{Y\}$ be a decision class attribute. For example, a rule could be defined as follow:

$$\text{if } X_1 = A \text{ and } X_2 = B \text{ and } X_3 = C \text{ then } Y = D \quad (10)$$

Where A, B, C , and D are the attribute values. If the antecedent part of each rule is satisfied, the decision is made by the consequent part of the rule. For example, considering the above rule, if the attribute value X_1 was A , the attribute value X_2 was B , and the attribute value X_3 was C , then Y must be D . The way by which rules are extracted has been schematically shown in Fig. 13.

Actually, the antecedent part of a rule consists of a sequence of successive judgment nodes with their judgment results, and consequence part of a rule consists of succeeding processing node. The rules are stored in buying-up rule pool, selling-up rule pool, buying-down rule pool and selling-down rule pool based on the resulting action of the rule and the stock price trend. Let us consider the following example to better explain how to extract trading rules on different days and divide them between rule pools based on stock price trend and resulting actions of the rules. Table 2 shows the example data at time t , $t+1$, $t+2$, $t+3$. The node transitions which is executed at time t , $t+1$, $t+2$, $t+3$ are shown in Fig. 14.

At time t , the node transition is started from the start node. Since there is one connection from the start node so, the next node is G/D which is a judgment node. G/D technical index is recalled and its result is checked. One branch is selected based on G/D's result and the next node is MACD. The value of MACD is A as shown in Table 2. So, the branch A is opted and the node is transferred to VR. As the previous judgment node, the value of VR is checked which is A . The node is transferred to the next node which is a processing node. The action "sell" is decided at time t . The trend is checked which is an "up-trend" at time t . The rule "if $G/D=B$ and $MACD=A$ and $VR=A$ then sell" is generated and stored in sell-up rule pool. At time $t+1$, the node transition is continued from the last node which has been used at time t . Since the last node was a processing node the node is transferred to the next node which is CANDLE. In this way rule extraction continues for $t+1$, $t+2$, $t+3$.

To define a fitness value for each extracted rule, if portfolio contains no stock and a rule with buying action was generated, the fitness value of this rule and rules generated later are equal when a rule with selling action is generated, and the fitness value is calculated by Eq. 11. The fitness values of other rules are zero. Fig.15 shows how the fitness value of the extracted rules are determined.

$$fitness\ value = selling\ price - buying\ price \quad (11)$$

The rules extracted by the best team and the extracted rules of the second best team whose fitness values are strictly positive are stored in rule pools.

3.1.Trading signal generation by matching calculation

In the training period, trading rules are extracted by the best teams and are stored in the rule pools. In the testing period, trading signals are generated using the extracted trading rules. This procedure is explained as follows.

3.1.1. Matching degree

To generate trading signals, the stock data d on a certain day is matched with all stock data recorded in stock trading rules in each rule pools. The matching degree of the stock data d , which consists of technical indices and their level on a certain day, with rule r in the rule pool k is calculated as follows:

$$Match_k(d, r) = \frac{N_k(d, r)}{N_k(r)} \quad (12)$$

where $N_k(d, r)$ is the number of matched judgment results in the antecedent part of the stock trading rule r with the stock data d . $N_k(r)$ is the number of judgments in the antecedent part of the stock trading rule r . The average matching degree of stock data d with all stock trading rules in rule pool k is calculated by Eq. 13. R_k is the set of suffixes of stock trading rules in rule pool k .

$$m_k(d) = \frac{1}{|R_k|} \sum_{r \in R_k} Match_k(d, r) \quad (13)$$

For each day relevant to training data, the average matching degree is calculated by Eq. 13 for each rule pool. The mean and the standard deviation of the matching degree of all training data are calculated as follows:

$$Mean_k = \frac{1}{|D_{training}|} \sum_{d \in D} m_k(d) \quad (14)$$

$$Std_k = \sqrt{\frac{1}{|D_{training}|} \sum_{d \in D} (m_k(d) - Mean_k)^2} \quad (15)$$

3.1.2. Trading signal generation

After calculating the mean and standard deviation of the matching degree for four rule pools for each day in testing period, the rule pools are divided into two groups based on the trend. The first one is devoted to the buying and selling rule pools for uptrends and the second one is devoted to buying and selling rule pools for downtrends. The procedure has been shown in Fig. 16. For each stock data d , the average matching degrees of both selected rule pools are calculated by Eq. 13.

If the portfolio contains a stock, first consider whether holding the stock is a suitable decision or not. If the average matching degree ($m_{k_1}(d)$) keeps relation (16) true, holding the stock is a suitable decision. If (16) was not true, consider whether selling the stock is a suitable decision or not. To check this, it is considered whether the matching degree ($m_{k_2}(d)$) holds (17) true or false. If it is true then selling the stock is a suitable decision. Finally, if none of (16) and (17) were true, no trade is advised.

Now, if the portfolio contains no stock, first consider whether the stock data belongs to selling class or not. In other words, consider whether holding cash is a suitable decision or not. If the stock data d belongs to the selling class, no actions is done. That is, the average matching degree ($m_{k_2}(d)$) keeps (18) true. If the stock data d did not belong to the selling class, consider whether stock buying is a suitable decision or not. So, if the average matching degree ($m_{k_1}(d)$) holds (19)

true, then buying stock is suitable. If the average matching degree was not true for (18) and (19), no trade is done.

In fact, if the described method is used for generating trading signals, the trading signals would be extended from binary trading signals (buying or selling signal) to trinary trading signals (buying or selling or no trading signals). Sometimes in a stock market, the best action done by a trader is to do nothing. So, the mentioned method for generating trading signals enabled LCA-NBQL method to make decisions as stock traders.

$$m_{k_1}(d) \geq Mean_{k_1} + \eta_1 Std_{k_1} \quad (16)$$

$$m_{k_2}(d) \geq Mean_{k_2} + \eta_2 Std_{k_2} \quad (17)$$

$$m_{k_2}(d) \geq Mean_{k_2} + \eta_3 Std_{k_2} \quad (18)$$

$$m_{k_1}(d) \geq Mean_{k_1} + \eta_4 Std_{k_1} \quad (19)$$

In inequalities (16)-(19), η ($\eta_1 = \eta_3 = .1$, $\eta_2 = \eta_4 = .4$) is a coefficient which adjusts the matching sensitivity in generating trading signals. If the coefficient is large, market conditions which match many rules cause to generate trading signals. Of course, strict policy can make to miss some trading opportunities. If the coefficient is small, market conditions which is matched with fewer rules generates trading signals, and hence more trades are done by the model. If un-strict policy is applied some trades would be done at unsuitable time.

The strict policy ($\eta_2 = \eta_4 = .4$) is used when the model wants to buy or sell stocks and the un-strict policy ($\eta_1 = \eta_3 = .1$) is used when the model wants to check the last previous decision which was made for buying or selling stocks in the previous days.

In case of GNP-RA, Mabu, Hirasawa [25], the rules are extracted more than once during the period used to generate trading signals. The method used by Mabu, Hirasawa [25] is used to generate trading signals.

4. Computational experiments and simulations

The stock trading problem considered in this research is described as follows:

- The initial fund is 10 million Iranian Rials for buying stocks.
- Stock data includes volume, open price, high price, low price, last price and close price on each day. Given that buying or selling is not possible at open price, last price, high price or low price, the close price is used when stocks are traded. The decision of buying and selling is made every trading day before the trading session.
- The basic and simple indicators calculated based on stock prices and volume on each trading day are used. A simple trading rule cannot be expected to anticipate all price trends and should combine simple trading rules together to get a complex trading strategy [43], so simple indicators are used and combined with each other to create a collection of good trading rules.

- The stock trading problem is defined on Tehran Stock Exchange (TSE), where an investor only goes long on an investment. Therefore, the reward for reinforcement learning is given when stocks are sold.

$$\text{reward} = \text{selling price} - \text{buying price} \quad (20)$$

- The playing strength (fitness value) of team i , $i=1, \dots, L$, is calculated by Eq. 21 in t^{th} week.

$$f_i^t = \text{total profit in } t^{\text{th}} \text{ week} \quad (21)$$

- To update the value of Q in the backward learning, the punishment value for the fifth judgment node in a step which only contains five judgment nodes, is calculated by Eq. 22.

$$\text{Punishment}_t = -0.1 \times \text{stock price} \quad (22)$$

where the stock price is considered as the close price at t^{th} day.

Table 3 shows the transaction costs on TSE according to the TSE's regulation. The transaction costs are calculated based on the buying and selling price and the trade volume.

20 stocks from TSE were selected from various fields such as car, pharmaceuticals, metal, etc. We selected companies with large market capitalization and relatively high liquidity.

Table 4 shows the parameters used in the experiments. For the comparison, the experiments are carried out by LCA-NBQL GA, and GNP-RA which is one of the most successful algorithms for stock trading rule extraction. The number of individuals is determined considering the balance between the calculation time and performance in the training phase. The parameters for backward learning were selected experimentally to show a good performance in the training phase in terms of the fitness value. The parameters used for GNP-RA, are also shown in Table 4, which were obtained from Mabu, Hirasawa [25]. The parameters used for GA were determined based on its performance in the training phase.

4.1. Technical indices

Our study uses simple technical indicators because the purpose of the rule based trading model is to make complex trading rules from simple trading rules. The technical indices used in our simulations are shown in Table 5. All technical indices, except golden/dead cross, MACD and candle chart, are calculated based on three calculation periods.

Table 6 shows the judgment results of each technical index used as judgment functions in each sub-node. The judgment result for each index is determined in this way: first the value of index is calculated using stock data, then its value is compared with Table 6 to determine the interval it lies inside as the result of the judgment node. Fig. 18 shows the eight chart patterns which are used to determine the candle chart judgment results. We use the judgment results introduced by Mabu, Hirasawa [25] as the judgment results for technical indicators used in GNP-RA.

Data set

The historical stock data is divided into two parts: training and testing data. Our study considers data of 1000 trading days leading up to 20 August 2016 as stock data, because sometimes some stocks were not allowed to trade. The training and testing period consists of 750 and 250 trading days, respectively. The training and testing data include not only up but also down trends. So, these are suitable periods to evaluate the performance. Since the companies are opted from several field, their stock prices exhibit various trends. The training period is divided into smaller periods to face the model with various market conditions and extract a lot of trading rules in an effective way. Hence, similar to Mabu, Hirasawa [25], the training period is divided into 30 smaller periods; each of them consists of 25 trading days. Fig. 19 shows schematically how the training period is divided. Then, the trading rule extraction is carried out for each sub-period. First, LCA-NBQL is trained for 38 iterations (or weeks in LCA terminology) using stock data of the first sub-period and the trading rules are stored in the rule pools. After finishing 38 iterations, training starts with a portfolio which only contains cash, and LCA-NBQL is trained using stock data of the second sub-period and the trading rules are stored in rule pools. This procedure is repeated for all sub-periods.

4.2. Simulation results

Table 7 shows the profits obtained by LCA-NBQL, GNP-RA, and Buy&Hold for 20 stocks. Buy&Hold is a passive investment strategy which is applied through buying stocks via all the cash in portfolio on the first day in the testing period and selling them on the last day. This strategy is in opposition to absolute market timing and is often considered as a benchmark of stock trading simulations.

Comparing the results, it can be seen that GNP-RA performs worse than the other two models. GNP-RA has done a lot of trades which their high transaction costs have diluted money. GNP-RA continued to hold losing stocks and sold winning stocks too soon. These are some reasons that why GNP-RA had not been able to perform well.

The rule-based trading model proposed by LCA-NBQL performs better than Buy&Hold in 17 out of 20 cases, which shows the rule-based trading system and active investing have had a higher profit than passive trading strategy known as Buy&Hold strategy. The proposed model performs as good as Buy&Hold in sharp uptrends and avoids large losses in severe downtrend.

4.3. Trading process

Some trading processes are analyzed to study the performance of LCA-NBQL better. Changes in the amount of the asset obtained by Buy&Hold strategy can show the direction of stock price changes. Fig. 20 shows the changes in the amount of assets obtained by LCA-NBQL and Buy&Hold strategy in the trade of MARK. The price trend is down at the beginning of the testing period. LCA-NBQL has not done any trade and the portfolio contains cash during this period. LCA-NBQL has bought shares at the end of downtrend like a stock trader and the portfolio contains shares during the sharp uptrend. LCA-NBQL gets more profit than Buy&Hold strategy.

It can be seen from Fig. 21 that for example for the ROOI stock, the price trend is a sideways trend. LCA-NBQL has done one pair of trade and obtained profit. The price is getting lower, but LCA-NBQL avoids losses by holding cash in its portfolio. In the middle of testing period, LCA-NBQL buys shares when the sideways trend turns into an uptrend. The portfolio thus contains shares during the sharp uptrend. LCA-NBQL gains more profit than Buy&Hold. Fig. 22 shows the behavior of algorithms for when trading NAFT. The price trend is a severe downtrend and the portfolio contains cash during this period. LCA-NBQL buys stock when the downtrend is turned into a trendless phase. The portfolio contains shares during the uptrend. LCA-NBQL performs better than Buy&Hold in this case. Fig. 23 depicts the algorithm's trades for the case of MKBT. The price trend is an uptrend. LCA-NBQL buys shares at the first day of the testing period. The portfolio contains shares during the uptrend. LCA-NBQL sells shares when the uptrend has turned to a sideways trend. LCA-NBQL cannot get profit from the last high rally. Buy&Hold strategy gets a little higher profit than LCA-NBQL.

As mentioned in section 3.1.2, for generating a trading signal in testing phase, the strict policy is used to eliminate false signal which is the result of stock price fluctuations and to pay lower transaction costs since they are significant amounts at Tehran Stock Exchange. Therefore, LCA-NBQL has done a few trades in some of the studied stocks. For example, in case of MKBT, LCA-NBQL has done only one set of trades, i.e. one time buying and one time selling, and gets profit from this set. So, the performance of LCA-NBQL cannot be evaluated based on performance measurement metrics such as risk adjusted return measures which uses value at risk (VaR) or conditional value at risk (CVaR). Winning trades versus losing trades is not a suitable performance measure [44]. Therefore, the performance of the proposed LCA-NBQL has been evaluated by the profit it provides. However, the main points of this paper are to introduce a new algorithm for extracting stock trading rules and to enhance decision making based on the extracted rules. So, it is one of the remaining works to be studied in the future to consider many kinds of parameters for creating a more realistic stock trading model.

5. Conclusion

In this paper, the LCA-NBQL algorithm was introduced to extract stock trading rules. It works based on the rationale of the league championship algorithm, network structure and backward Q-learning. Each individual has a network structure, which enables it to reuse nodes as a compact structure, and enhances its performance through learning from weaknesses and strengths of other individual teams. The algorithm was equipped with backward Q-learning to increase learning speed and consequently extract better stock trading rules. The trend clustering was also used to determine the stock price trend which categorizes stock prices into uptrend and downtrend. LCA-NBQL was encountered with various market conditions in the training period and showed that it can remember them in the testing period and can generate more creditable signals. In some market situations, the proposed method can generate no-trade signal and do nothing. The simulation results showed that the proposed method performs better than GNP-RA and Buy&Hold. It showed a suitable performance in sharp uptrend. The ϵ -greedy policy used in Sarsa learning may choose smaller Q value, and some unimportant stock trading rules may be extracted and stored in the rule pools. Therefore, for future research, these unimportant stock trading rules can be pruned from the

rule pools in order to improve the quality of the rule pools. In addition, a combination of Fundamental analysis and Technical analysis can be used to extract stock trading rules because most of stock traders not only consider technical analysis but also care about fundamental factors.

References

1. Murphy, J.J., *Technical analysis of the financial markets : a comprehensive guide to trading methods and applications*. 1999, New York: New York Institute of Finance.
2. Malkiel, B.G. and E.F. Fama, *Efficient capital markets: A review of theory and empirical work*. The journal of Finance, 1970. **25**(2): p. 383-417.
3. Cootner, P.H., *The random character of stock market prices*. 1967, Cambridge, Mass.: M.I.T. Press.
4. Nison, S., *Japanese candlestick charting techniques : a contemporary guide to the ancient investment techniques of the Far East*. 2001, New York: New York Institute of Finance.
5. Cunningham, S., *The predictability of British stock market prices*. Applied Statistics, 1973: p. 315-331.
6. Dryden, M.M., *Filter tests of UK share prices*. Applied Economics, 1970. **1**(4): p. 261-275.
7. Jensen, M.C. and G.A. Benington, *Random walks and technical theories: Some additional evidence*. The Journal of Finance, 1970. **25**(2): p. 469-482.
8. Fama, E.F. and M.E. Blume, *Filter rules and stock-market trading*. The Journal of Business, 1966. **39**(1): p. 226-241.
9. Bessembinder, H. and K. Chan, *The profitability of technical trading rules in the Asian stock markets*. Pacific-Basin Finance Journal, 1995. **3**(2): p. 257-284.
10. Brock, W., J. Lakonishok, and B. LeBaron, *Simple technical trading rules and the stochastic properties of stock returns*. The Journal of finance, 1992. **47**(5): p. 1731-1764.
11. Givoly, D. and J. Lakonishok, *The information content of financial analysts' forecasts of earnings: Some evidence on semi-strong inefficiency*. Journal of Accounting and Economics, 1979. **1**(3): p. 165-185.
12. Hudson, R., M. Dempsey, and K. Keasey, *A note on the weak form efficiency of capital markets: The application of simple technical trading rules to UK stock prices-1935 to 1994*. Journal of Banking & Finance, 1996. **20**(6): p. 1121-1132.
13. Wong, M., *Fund management performance, trend-chasing technical analysis and investment horizons: a case study*. Omega, 1997. **25**(1): p. 57-63.
14. Bauer, R.J., *Genetic algorithms and investment strategies*. 1994, New York: Wiley.
15. Neely, C.J., P. Weller, and R. Dittmar, *Is technical analysis in the foreign exchange market profitable? A genetic programming approach*. Journal of financial and Quantitative Analysis, 1997. **32**(04): p. 405-426.
16. Neely, C.J. and P.A. Weller, *Technical trading rules in the European monetary system*. Journal of International Money and Finance, 1999. **18**(3): p. 429-458.
17. Cheng, C.-H., T.-L. Chen, and L.-Y. Wei, *A hybrid model based on rough sets theory and genetic algorithms for stock price forecasting*. Information Sciences, 2010. **180**(9): p. 1610-1629.
18. Chien, Y.-W.C. and Y.-L. Chen, *Mining associative classification rules with stock trading data—A GA-based method*. Knowledge-Based Systems, 2010. **23**(6): p. 605-614.
19. Berutich, J.M., et al., *Robust technical trading strategies using GP for algorithmic portfolio selection*. Expert Systems with Applications, 2016. **46**: p. 307-315.
20. Esfahanipour, A. and S. Mousavi, *A genetic programming model to generate risk-adjusted technical trading rules in stock markets*. Expert Systems with Applications, 2011. **38**(7): p. 8438-8445.

21. Chang, P.-C., et al., *A dynamic threshold decision system for stock trading signal detection*. Applied Soft Computing, 2011. **11**(5): p. 3998-4010.
22. Dash, R. and P.K. Dash, *A hybrid stock trading framework integrating technical analysis with machine learning techniques*. The Journal of Finance and Data Science, 2016.
23. Izumi, Y., et al. *Trading rules on the stock markets using genetic network programming with candlestick chart*. in *2006 IEEE International Conference on Evolutionary Computation*. 2006. IEEE.
24. Chen, Y., et al., *A genetic network programming with learning approach for enhanced stock trading model*. Expert Systems with Applications, 2009. **36**(10): p. 12537-12546.
25. Mabu, S., et al., *Enhanced decision making mechanism of rule-based genetic network programming for creating stock trading signals*. Expert Systems with Applications, 2013. **40**(16): p. 6311-6320.
26. Mabu, S., M. Obayashi, and T. Kuremoto, *Ensemble learning of rule-based evolutionary algorithm using multi-layer perceptron for supporting decisions in stock trading problems*. Applied Soft Computing, 2015. **36**: p. 357-367.
27. Li, X. and K. Hirasawa, *Continuous probabilistic model building genetic network programming using reinforcement learning*. Applied Soft Computing, 2015. **27**: p. 457-467.
28. Li, X., S. Mabu, and K. Hirasawa, *A novel graph-based estimation of the distribution algorithm and its extension using reinforcement learning*. IEEE transactions on evolutionary computation, 2014. **18**(1): p. 98-113.
29. Husseinzadeh Kashan, A. *League Championship Algorithm: A New Algorithm for Numerical Function Optimization*. in *SoCPaR*. 2009.
30. Husseinzadeh Kashan, A., *An efficient algorithm for constrained global optimization and application to mechanical engineering design: League championship algorithm (LCA)*. Computer-Aided Design, 2011. **43**(12): p. 1769-1792.
31. Husseinzadeh Kashan, A., *League Championship Algorithm (LCA): An algorithm for global optimization inspired by sport championships*. Applied Soft Computing, 2014. **16**: p. 171-200.
32. Chen, Y. and X. Wang, *A hybrid stock trading system using genetic network programming and mean conditional value-at-risk*. European Journal of Operational Research, 2015. **240**(3): p. 861-871.
33. Watkins, C.J. and P. Dayan, *Q-learning*. Machine learning, 1992. **8**(3-4): p. 279-292.
34. Sutton, R.S., *Generalization in reinforcement learning: Successful examples using sparse coarse coding*. Advances in neural information processing systems, 1996: p. 1038-1044.
35. Sutton, R.S. and A.G. Barto, *Reinforcement learning : an introduction*. 1998.
36. Wiering, M.A. and H. Van Hasselt, *Ensemble algorithms in reinforcement learning*. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2008. **38**(4): p. 930-936.
37. Boyan, J. and A.W. Moore, *Generalization in reinforcement learning: Safely approximating the value function*. Advances in neural information processing systems, 1995: p. 369-376.
38. Gordon, G., *Stable function approximation in dynamic programming*. 1995, Pittsburgh, Pa.: School of Computer Science, Carnegie Mellon University.

39. Wang, Y.-H., T.-H.S. Li, and C.-J. Lin, *Backward Q-learning: The combination of Sarsa algorithm and Q-learning*. Engineering Applications of Artificial Intelligence, 2013. **26**(9): p. 2184-2193.
40. Husseinzadeh Kashan, A. and B. Karimi. *A new algorithm for constrained optimization inspired by the sport league championships*. in *IEEE Congress on Evolutionary Computation*. 2010. IEEE.
41. Wu, C., S. Chou, and H. Liaw, *A trend based investment decision approach using clustering and heuristic algorithm*. Science China Information Sciences, 2014. **57**(9): p. 1-14.
42. Wang, L., et al., *Genetic network programming with rule accumulation and its application to Tile-world problem*. J. Adv. Comput. Intell. Intelligent Informatics Journal of Advanced Computational Intelligence and Intelligent Informatics, 2009. **13**(5): p. 551-560.
43. Pring, M.J., *Technical analysis explained : the successful investor's guide to spotting investment trends and turning points*. 1991.
44. Patel, M., *Trading with Ichimoku clouds : the essential guide to Ichimoku Kinko Hyo technical analysis*. 2010.

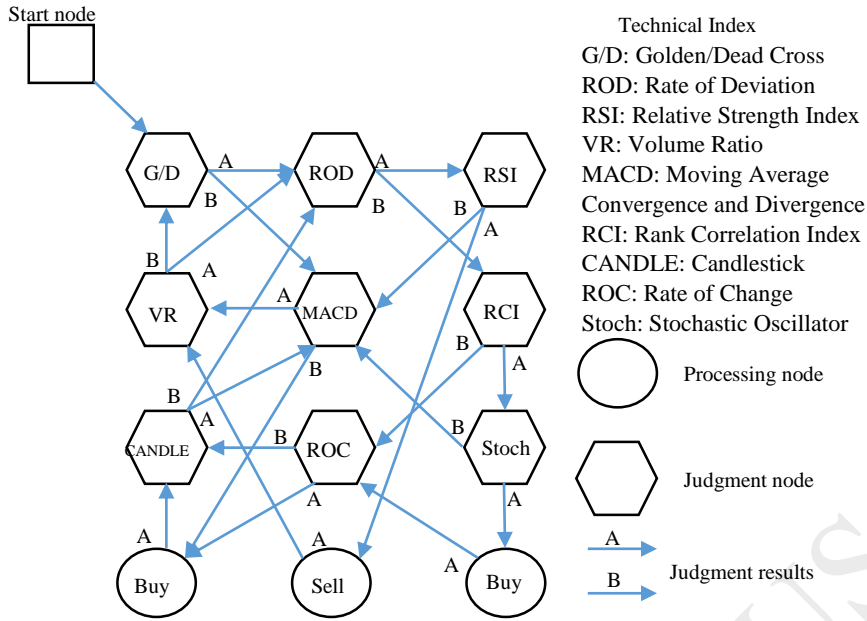


Fig. 1. Basic structure of a team in LCA-NBQL algorithm.

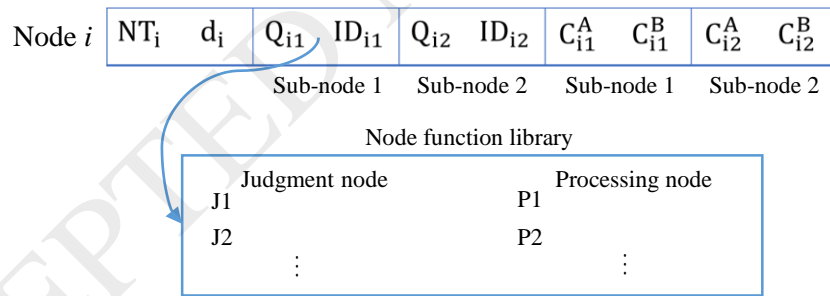


Fig. 2. Structure of each node.

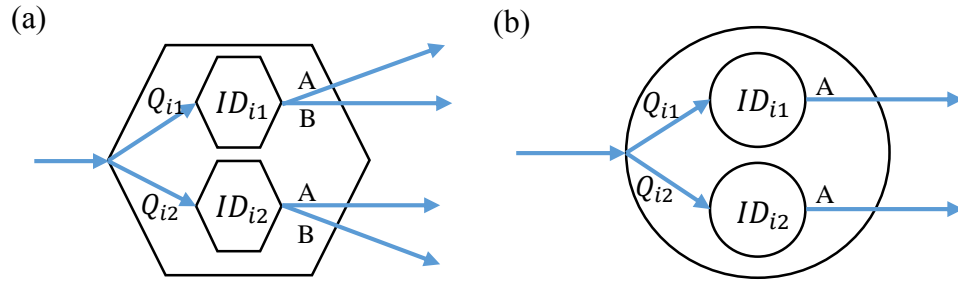


Fig. 3. Judgment node and processing node structures.

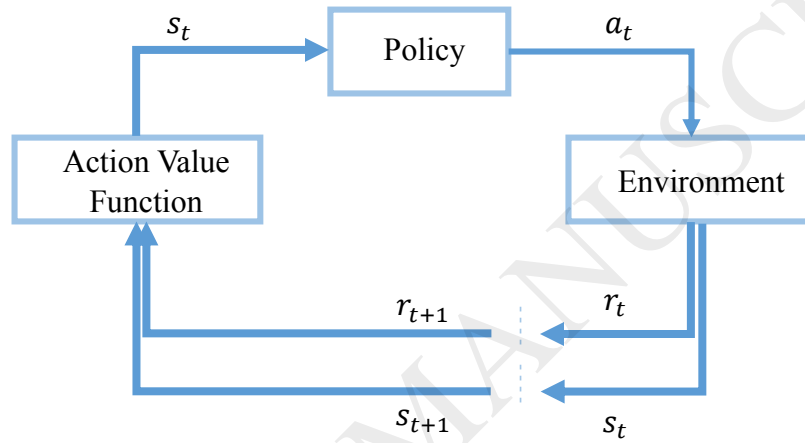


Fig. 4. The standard architecture of the reinforcement learning.

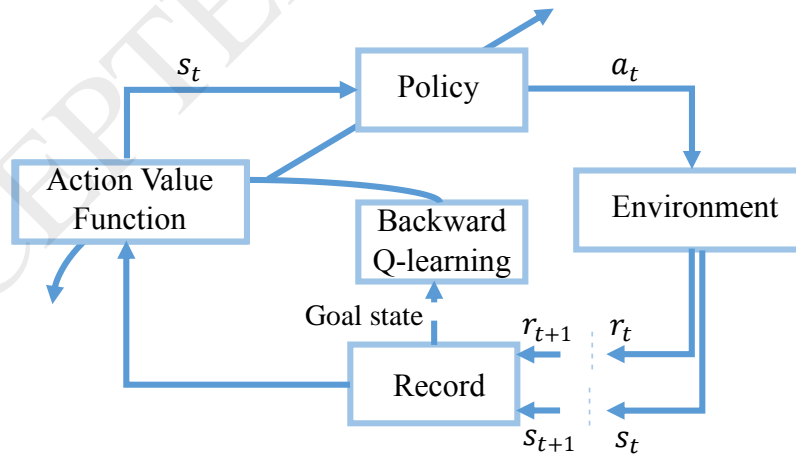


Fig. 5. The architecture of the reinforcement learning algorithm integrated with backward Q-learning.

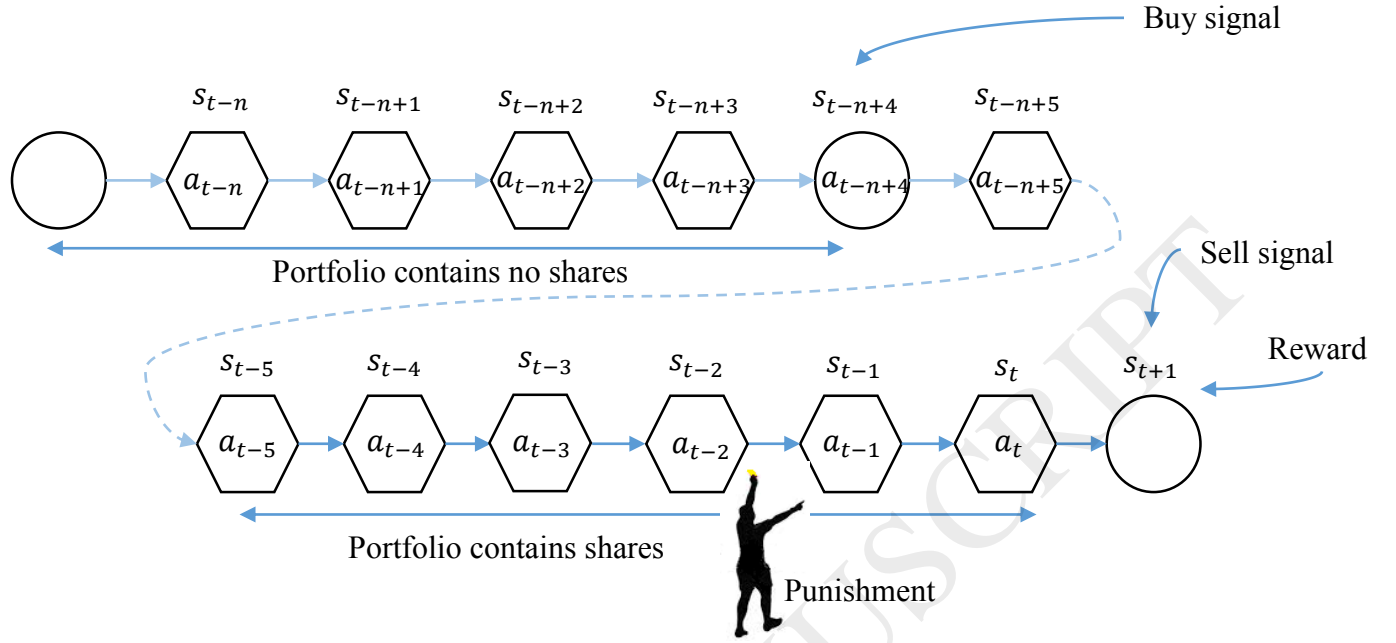


Fig. 6. An example of backward Q-learning.

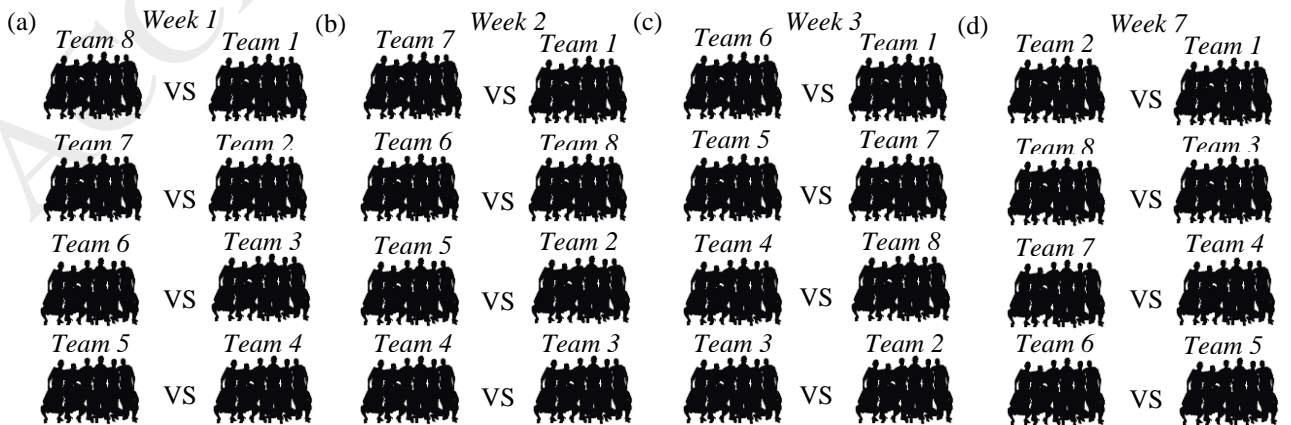


Fig. 7. An illustrative example of the league scheduling algorithm.

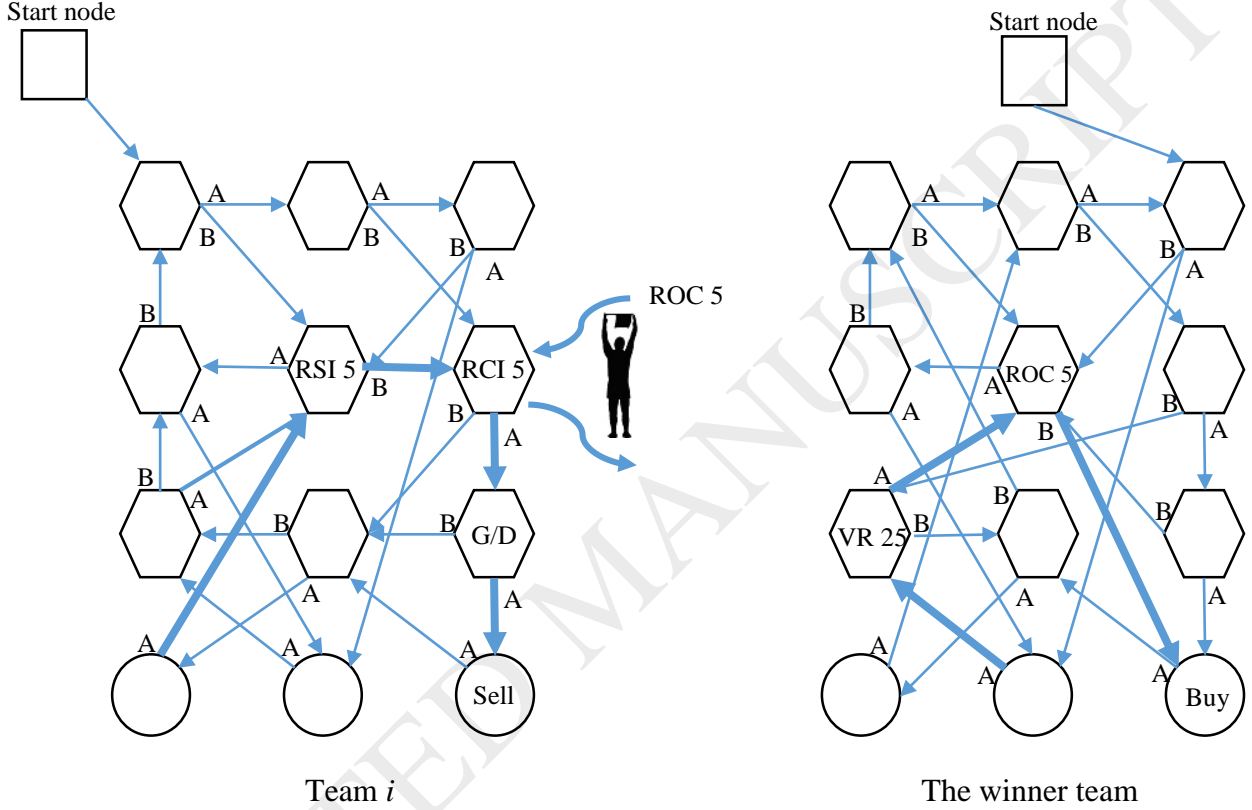


Fig. 8. An example of learning from fortes of a winner team.

$m :=$ the number of steps in each week

Integer n is selected randomly such that $n \in \{1, \dots, m\}$;

Choose randomly n -step among m -step of team i . Let the selected steps be $\{step_{m_1}, \dots, step_{m_n}\}$;

For each step m_p do

Integer v_{m_p} is selected randomly such that $0 \leq v_{m_p} \leq \min(\text{length}_{m_p}^i, \text{length}_{m_p}^{\text{winner}})$;

Choose randomly v_{m_p} nodes among the nodes in step m_p of team i ;

For each selected node do

If the function of selected node belongs to functions of step m_p in winner team

The selected node in team i learns the connection from the node in step m_p of the winner team which has the same function;

Else if the function doesn't belong to functions of step m_p in winner team

Change the function of the selected node in step m_p in team i by the function of the selected node among nodes in step m_p in the winner team (repetition is not allowed);

The value of Q is updated using Eq. 7;
 End if
 End for
 End for

Fig. 9. Procedure of learning from a winner team.

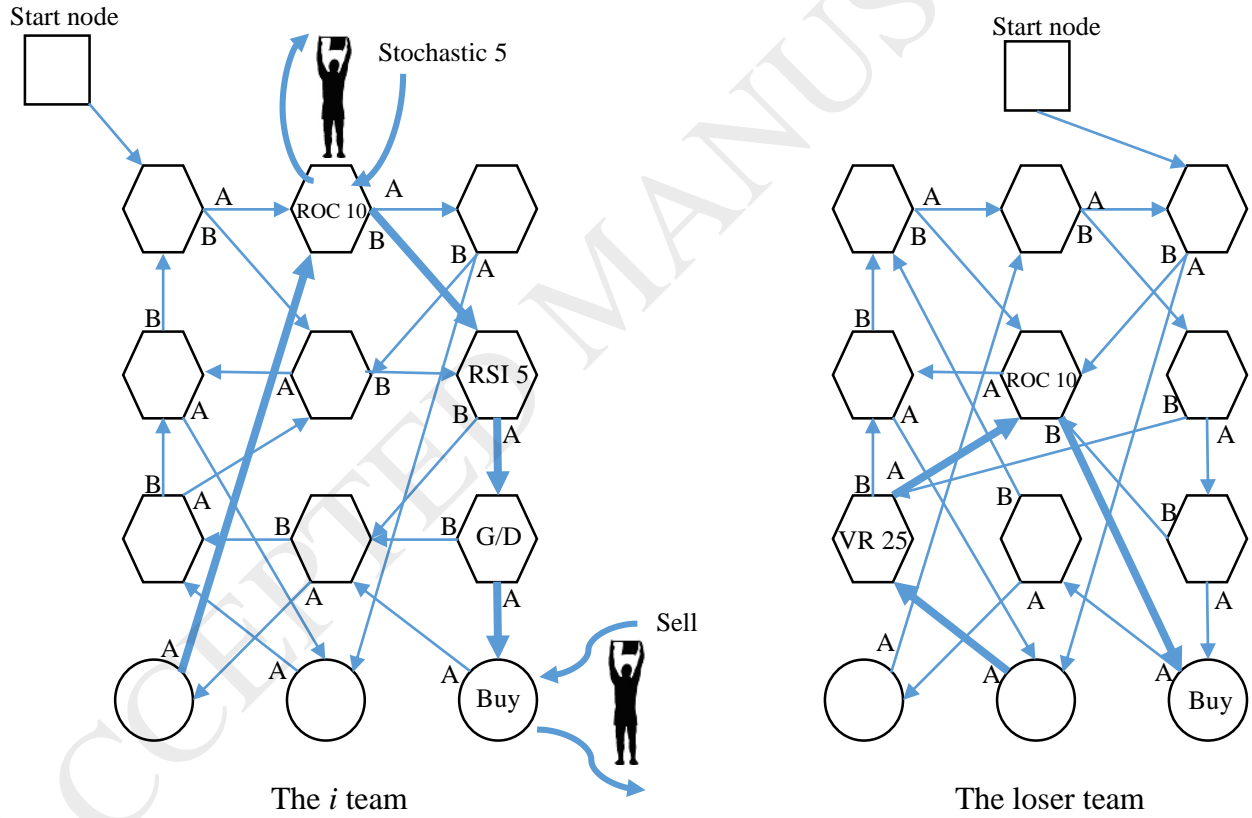


Fig 10. An example of learning from weaknesses of a loser team.

$m :=$ the number of steps in each week

Integer n is selected randomly such that $n \in \{1, \dots, m\}$;

Choose randomly n -step among m -step of team i . Let the selected steps be $\{step_{m_1}, \dots, step_{m_n}\}$;

For each step m_p do

Integer V_{m_p} is selected randomly such that $0 \leq V_{m_p} \leq \min(length_{m_p}^i, length_{m_p}^{loser})$;

Choose randomly V_{m_p} nodes among the nodes in step m_p of team i ;

For each selected node do

If the function of selected node belongs to functions of step m_p in the loser team

The function of the selected node in team i is changed by another function from function library

Else if the function doesn't belong to functions of step m_p in the loser team

The connections of the selected node changed with probability of p_q ;

End if

End for

End for

Fig 11. Procedure of learning from a loser team.

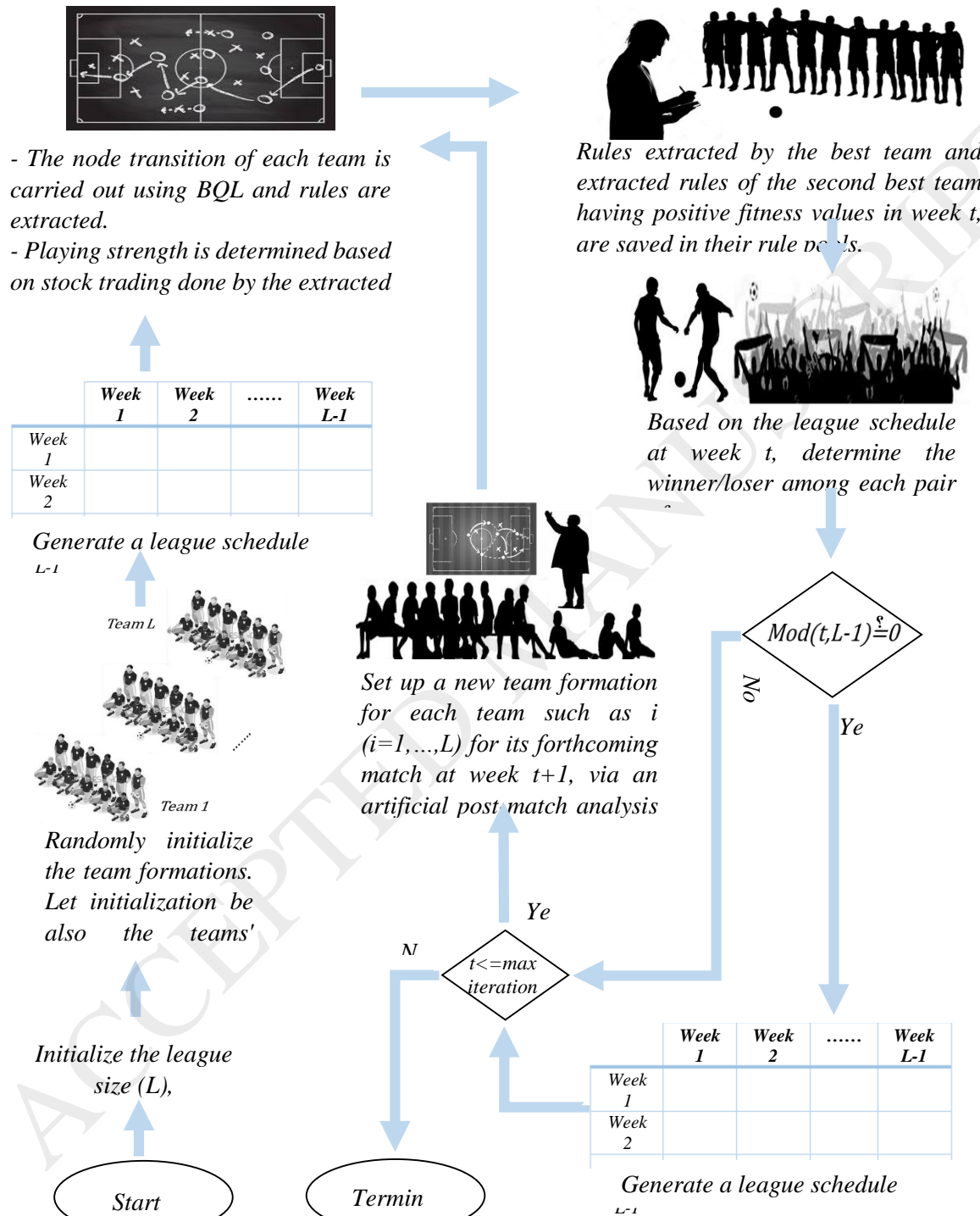


Fig. 12. Flowchart of LCA-NBQL algorithm for stock trading rule extraction.

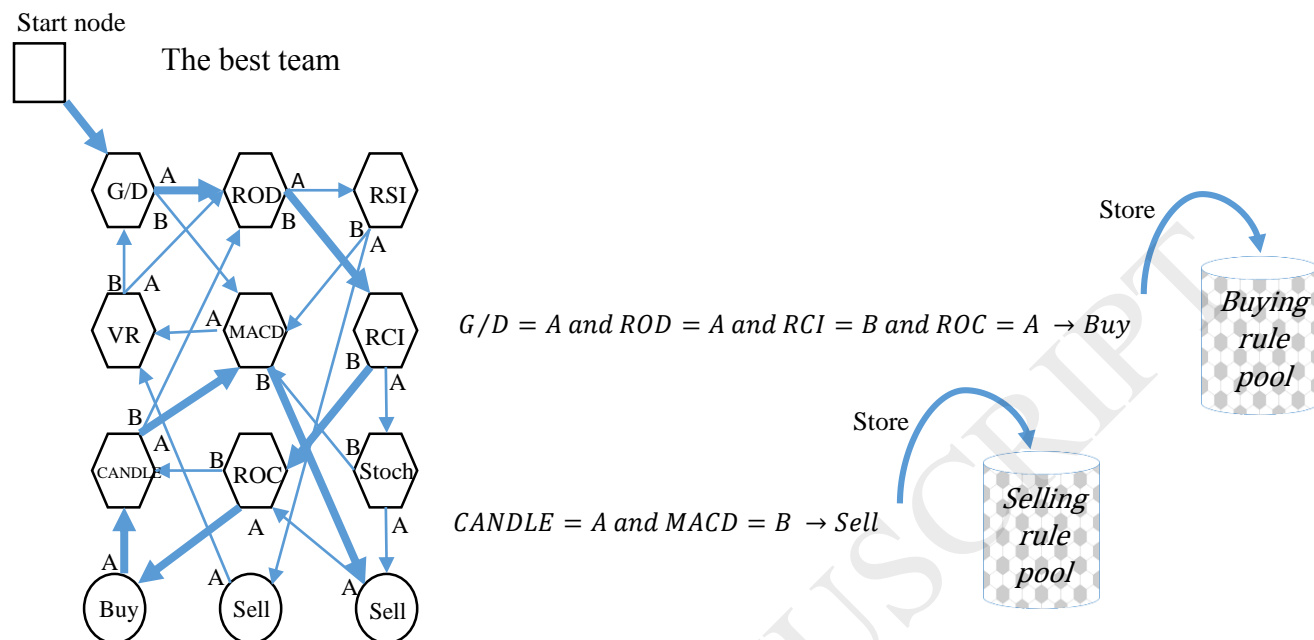


Fig. 13. Rule representation.

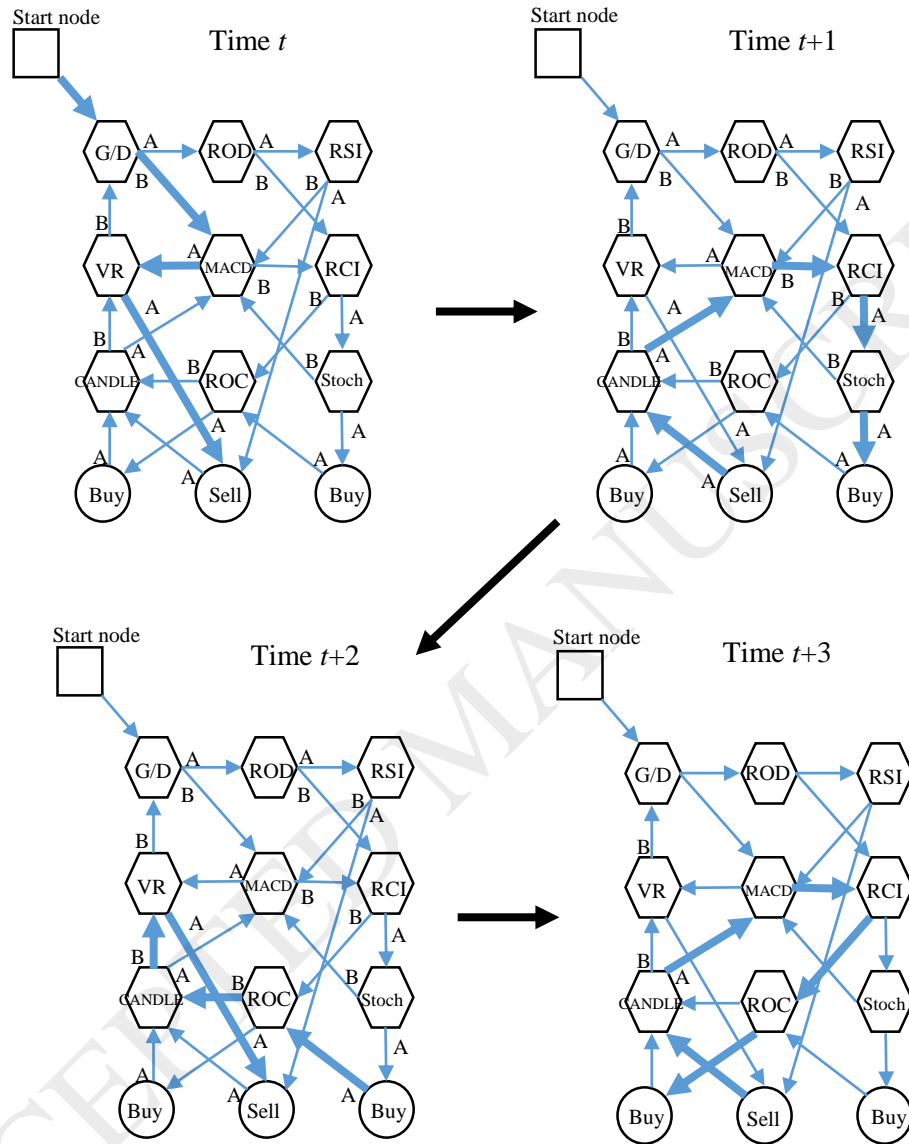


Fig. 14. An example of rule extraction from graph structure.

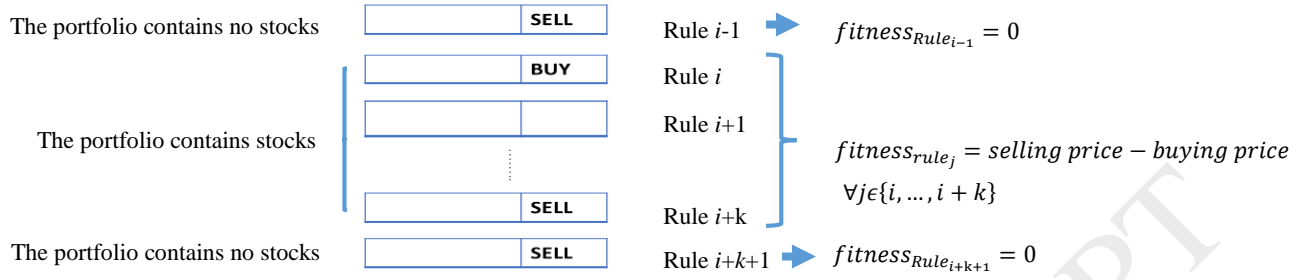


Fig. 15. An example of determining the fitness value of extracted rules.

Target classes selection

If up trend

$k_1 = buy_{up}$

$k_2 = sell_{up}$

else if down trend

$k_1 = buy_{down}$

$k_2 = sell_{down}$

end if

Fig. 16. Procedure for selection of target classes.

If stocks are held

If $m_{k_1}(d) \geq Mean_{k_1} + \eta_1 Std_{k_1}$

No action

Else if $m_{k_2}(d) \geq Mean_{k_2} + \eta_2 Std_{k_2}$

Sell all stocks

Else

No action

End if

Else if no stocks are held

If $m_{k_2}(d) \geq Mean_{k_2} + \eta_3 Std_{k_2}$

No action

Else if $m_{k_1}(d) \geq Mean_{k_1} + \eta_4 Std_{k_1}$

Buy stocks using all the funds

Else

No action

End if

End if

Fig. 17. Procedure of buying and selling decision.

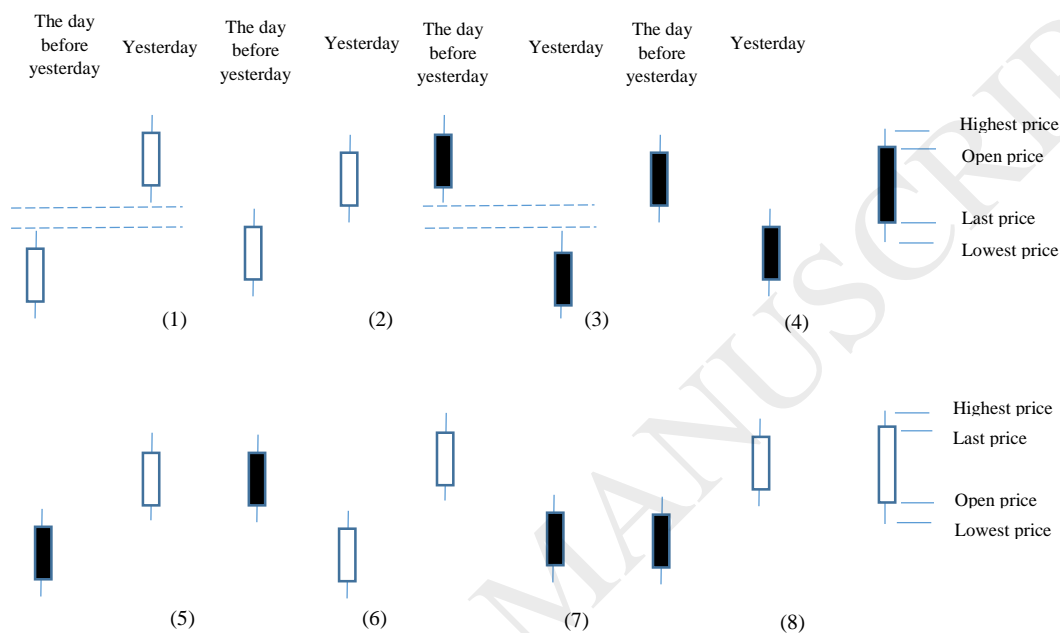


Fig. 18. Candlestick chart patterns.

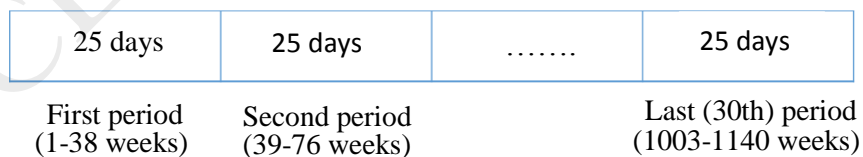


Fig. 19. An example of training period division.

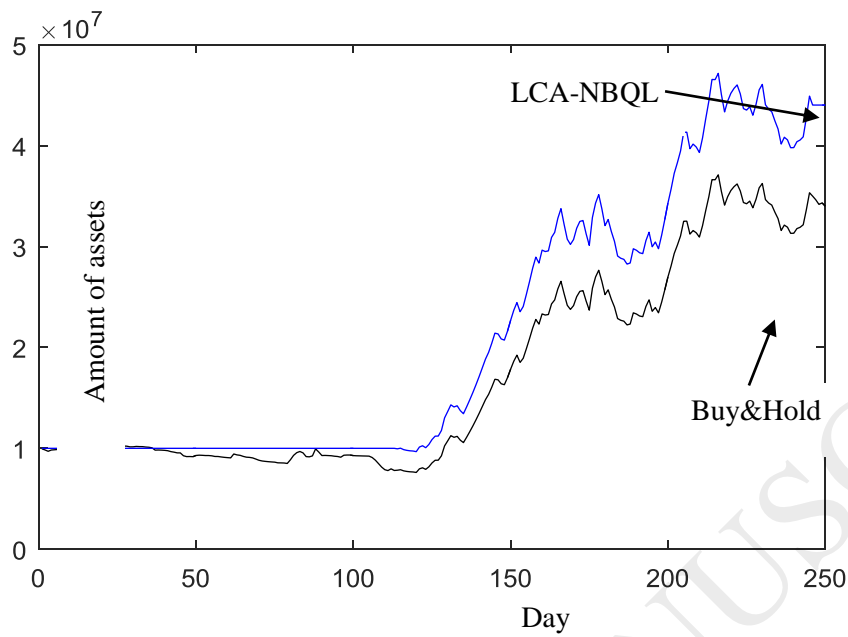


Fig. 20. Changes in the amount of assets in the trade of MARK in the testing phase.

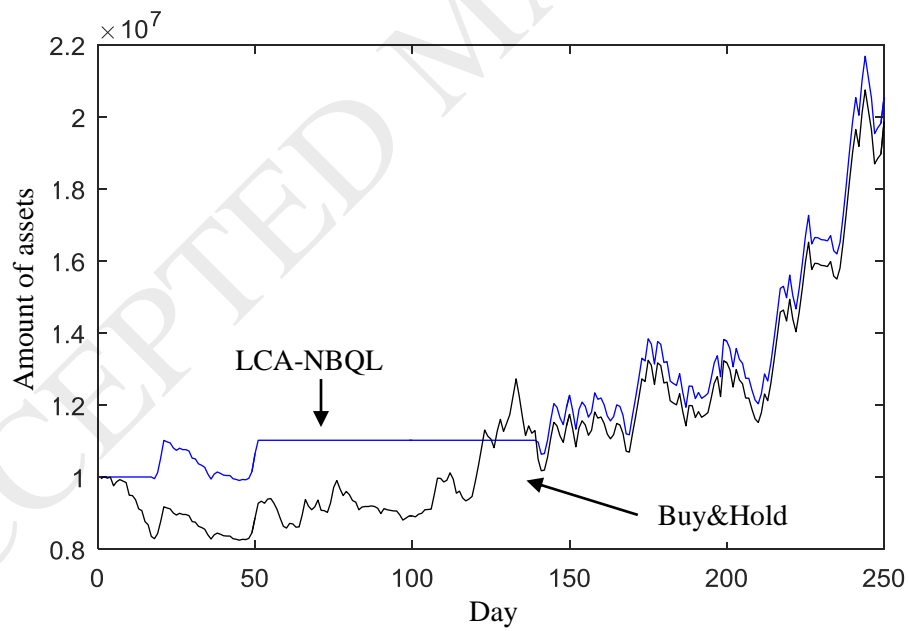


Fig. 21. Changes in the amount of assets in the trade of ROOI in the testing phase.

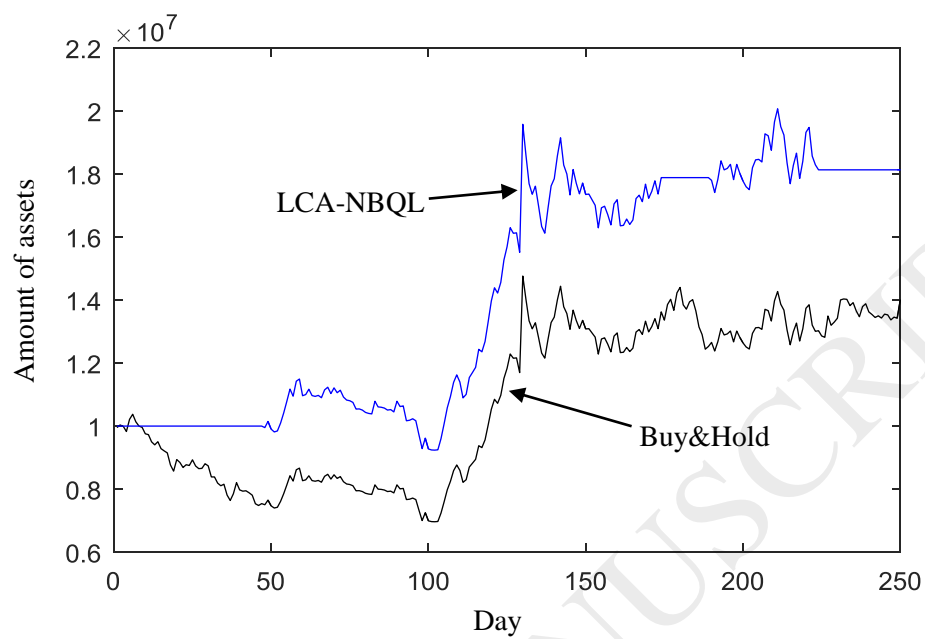


Fig. 22. Changes in the amount of assets in the trade of NAFTA in the testing phase.

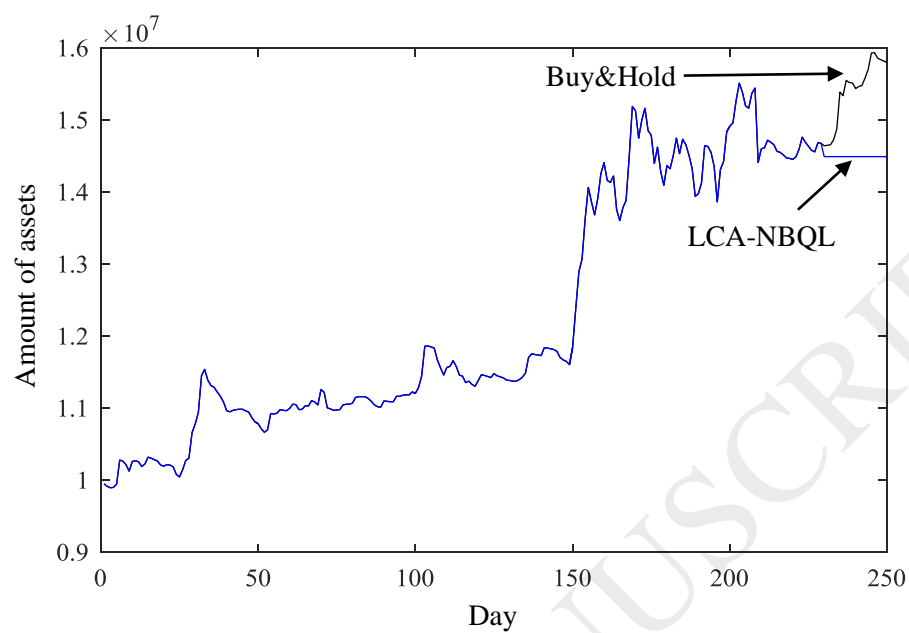


Fig. 23. Changes in the amount of assets in the trade of MKBT in the testing phase.

Table 1. Hypothetical SWOT matrix.


	Adopted <i>S/T</i> strategy	Adopted strategy	<i>S/O</i>	Adopted strategy	<i>W/T</i>	Adopted strategy	<i>W/O</i>
	<i>i</i> had won & <i>l</i> had won	<i>i</i> had won & <i>l</i> had won		<i>i</i> had won & <i>l</i> had won		<i>i</i> had won & <i>l</i> had won	
	Focusing on	Focusing on		Focusing on		Focusing on	
<i>S</i>	Weaknesses of <i>j</i>	Weaknesses of <i>j</i>					
<i>W</i>				Strengths of <i>j</i>		Strengths of <i>j</i>	
<i>O</i>		Strengths of <i>k</i>				Strengths of <i>k</i>	
<i>T</i>	Weaknesses of <i>k</i>			Weaknesses of <i>k</i>			

Table 2. An example of attitude data.

Time (day)	Technical Index								
	ROD	RSI	ROC	VR	Stochastic	RCI	G/D	MACD	CANDLE
<i>t</i>	-	-	-	<i>A</i>	-	-	<i>B</i>	<i>A</i>	-
<i>t+1</i>	-	-	-	-	<i>A</i>	<i>A</i>	-	<i>B</i>	<i>A</i>
<i>t+2</i>	-	-	<i>B</i>	<i>A</i>	-	-	-	-	<i>B</i>
<i>t+3</i>	-	-	<i>A</i>	-	-	<i>B</i>	-	<i>B</i>	<i>A</i>

Table 3. Transaction cost.

Deal type	Transaction cost (IRR)
buy	0.486 %
sell	1.029 %

Table 4. Simulation conditions.

LCA-NBQL
The number of team = 20
The number of nodes: 61 (judgment node: 40, processing node: 20, start node: 1)
The maximum number of weeks (iterations) is =38
$\alpha_b=0.9$, $\gamma_b=0.95$, $\alpha=0.1$, $\gamma=0.4$, $\varepsilon=0.1$
GNP-RL
The number of individuals = 100
(mutation: 38, crossover: 60, elite: 1)
The number of nodes: 61 (judgment node: 40, processing node: 20, start node: 1)
The number of generations=38
Crossover rate $P_c=0.5$, mutation rate $P_m=0.02$
$\alpha=0.1$, $\gamma=0.4$, $\varepsilon=0.1$
Genetic Algorithm
The number of individuals = 100

(mutation: 30, crossover: 80, elite: 1)
 Maximum number of iteration=100
 Crossover rate $P_c=0.8$, mutation rate $P_m=0.3$

Table 5. Calculation period of the technical indexes.

Technical Index	Period 1	Period 2	Period 3
Rate of Deviation	5	10	25
Relative Strength Index	5	10	25
Rate of Change	5	10	25
Volume Ratio	5	10	25
Stochastics	5	10	25
Ranked Correlation Index	5	10	25
Golden/Dead Cross	5 (short term)	25 (long term)	
Moving Average Convergence Divergence	5 (short term)	25 (long term)	9 (signal)

Table 6. Judgment results for each technical index and Candle pattern.

Technical index	Judgment result			
	A	B	C	D
ROD	$(-\infty, -0.05]$	$(-0.05, 0)$	$[0, .05)$	$[0.05, \infty)$
RSI	$[0, 0.2]$	$(0.2, 0.5)$	$[0.5, 0.8)$	$[0.8, 1]$
ROC	$[0, 0.9)$	$[0.9, 1)$	$[1, 1.1)$	$[1.1, \infty)$
Volume Ratio	$(-\infty, 0.3)$	$[0.3, 0.5)$	$[0.5, 0.7)$	$[0.7, \infty)$
Stochastics	$[0, 0.3]$	$(0.3, 0.5)$	$[0.5, 0.7)$	$[0.7, 1]$
RCI	$(-\infty, -.6]$	$(-0.6, 0)$	$[0, 0.6)$	$[0.6, \infty)$
Golden/Dead Cross	Golden cross	Dead cross	other	-
MACD	Golden cross	Dead cross	other	-
Candlestick chart	Each pattern in Fig. 18 corresponds to judgment result			

Table 7. Profits and their rates in the test simulation.

Company	Stock	LCA-NBQL		GNP-RA		Buy & Hold	
	Symbol	Profit [IRR]	Profit rate [%]	Profit [IRR]	Profit rate [%]	Profit [IRR]	Profit rate [%]
Telecommunication company of Iran	MKBT	4493340	44.93	-7255353	-72.55	5641355	56.41
Dana Insurance	BDAN	6464142	64.64	-6802202	-68.02	3831309	38.31
Dr. Abidi Pharmaceutical Laboratory	ABDI	46004409	460.04	-3989000	-39.89	44127844	441.28
Calcimin	KSIM	4335220	43.35	-5941916	-59.42	3083981	30.84
Machine Sazi Arak	MARK	34018252	340.18	-6505068	-65.05	23648035	236.48
Ardakan Industrial Ceramics Co	ARDK	-889863	-8.90	-6556333	-65.56	-1889673	-18.90
IRISL Group Company	KSHJ	-703653	-7.04	-8091688	-80.92	-771340	-7.71
Iran Zinc Mines Development Company	ROOI	10544997	105.45	-5622048	-56.22	9659206	96.59
Barez Industrial Group	BARZ	2673888	26.74	-7641398	-76.41	2038698	20.39
Iran Khodro Co	IKCO	10307758	103.08	-7339414	-73.39	1542288	15.42
Informatics Services Corporation	INFO	2523214	25.23	-6861617	-68.62	3302263	33.02
MAPNA Group	MAPN	1970312	19.70	-7285451	-72.85	3052488	30.52
Tehran Construction Co	NSTH	-928367	-9.28	-7957376	-79.57	-2730320	-27.30
Thran Cement Company	STEH	-664566	-6.65	-7549597	-75.50	-1373324	-13.73
Oil Industry Investment	NAFT	8130751	81.31	-7811984	-78.12	3866046	38.66
Shazand Petrochemical Company	PARK	-139783	-1.40	-7539771	-75.40	-792563	-7.93
Iranian Investment Incorporation Co	IRNZ	4364899	43.65	-6647576	-66.48	3429538	34.30
Omid Investment Group Corporation	OIMC	2881061	28.81	-6750402	-67.50	1719112	17.19
Pars Tousheh Investment Company	TSHE	5998048	59.98	-6655771	-66.56	5943603	59.44
Bank Mellat	BTEJ	1741904	17.42	-7255410	-72.55	1211560	12.12