IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2021.Doi Number

Intelligent Algorithmic Trading Strategy using Reinforcement Learning and Directional Change

Monira Aloud and Nora Alkhamees

Department of Management Information System, College of Business Administration, King Saud University, Saudi Arabia

Corresponding author: Monira Aloud (e-mail: mealoud@ksu.edu.sa).

Department of Management Information System, King Saud University

P. O. Box 75879, Riyadh 11588

Phone No: +966535704040

ORCID: https://orcid.org/0000-0002-0472-5151

The author thanks Researchers Supporting Project, number (RSP-2020/213), King Saud University, Riyadh, Saudi Arabia for supporting the research project.

ABSTRACT Designing a profitable trading strategy plays a critical role in algorithmic trading, where the algorithm can manage and execute automated trading decisions. Determining a specific trading rule for trading at a particular time is a critical research problem in financial market trading. However, an intelligent, and a dynamic algorithmic trading driven by the current patterns of price time-series data may help deal with this issue. Thus, Reinforcement Learning (RL) can achieve optimal dynamic algorithmic trading by considering the price time-series as its environment. A comprehensive representation of the environment states is indeed vital for proposing a dynamic algorithmic trading using RL. Therefore, we propose a representation of the environment states using the Directional Change (DC) event approach with a dynamic DC threshold. We refer to the proposed algorithmic trading approach as the DCRL trading strategy. In addition, the proposed DCRL trading strategy was trained using the Q-learning algorithm to find an optimal trading rule. We evaluated the DCRL trading strategy on real stock market data (S&P500, NASDAQ, and Dow Jones, for five years period from 2015-2020), and the results demonstrate that the DCRL state representation policies obtained more substantial trading returns and improved the Sharpe Ratios in a volatile stock market. A series of performance analyses demonstrate the robust performance and extensive applicability of the proposed DCRL trading strategy.

INDEX TERMS Machine learning; reinforcement learning; Q-learning; directional change event; algorithmic trading; stock market.

I. INTRODUCTION

Developing algorithmic trading strategies that can make timely stock trading decisions has always been a subject of interest for investors and financial analysts. The decision-making problem for financial trading remains particularly challenging given the variety of factors that can influence stock prices. The design challenge of algorithmic trading primarily emerges from the continuous evolution of price time-series and thus the dynamic cycle of making trading action decisions. Algorithmic trading is based on computer algorithms to produce automated trading decisions and place orders in the market and manage the portfolio. Recent advancements in information technologies and machine learning techniques have led to the creation of algorithmic trading, which is also referred to as quantitative trading [1]. Decision-making in financial trading requires the trading algorithm to explore the environment and make appropriate and timely decisions without supervised information.

Classic algorithmic trading strategy models include trend-following and mean reversion strategies [2]. Early works include the use of filter trading rules to control when to buy or sell a stock [3]. Several studies have investigated algorithmic trading, including algorithms based on fundamental and technical analysis indicators and algorithms based on machine learning techniques [1]. Machine learning algorithms learn from historical data and interacts with the environment to generate profitable trading rules. Machine learning techniques for algorithmic trading can be divided into supervised learning and RL algorithm-based methods [4]. Supervised learning methods examine and analyze training data (structured data) to predict stock prices or trends. The RL algorithm recognizes different environmental states, and it performs an action and receives feedback (i.e., a reward). Thus, RL methods learn to change actions to maximize future rewards [5]. In this study, we develop an algorithmic trading strategy using a machine learning technique, i.e., a hybrid of Reinforcement Learning (RL) and Q-learning algorithms.

RL is a machine learning method used for sequential decision-making problems [5]. It achieves policy improvement throughout continuous interaction with and ongoing evaluation of its environment. A RL agent performs a sequence of actions based on the environment states to receive a predefined reward. In contrast to supervised machine learning, which requires historical labeled data, the RL agent learns the environment's states and performs actions through continuous evaluation of the dynamic environment. The RL algorithm has several advantages, e.g., self-learning, ongoing behavior enhancements, and adaptivity to the environment states. RL has been applied effectively in different domains, e.g., job scheduling [6], pattern recognition [7], and algorithmic trading [8-11].

Despite the effectiveness and robustness of the RL algorithm, employing an algorithmic trading strategy remains a challenge in real-world trading for three reasons. First, using a physically fixed time interval (e.g., hourly data) to represent the environment states make the flow of price time-series irregularly spaced because prices are transacted at irregular times and at different magnitudes and directions [12]. Physical time employs a point-based system, where a single time unit for observing price changes in range from seconds to hours or even days; thus, time is homogeneous. Under intrinsic time, the Directional Change (DC) event approach emerges as an alternative approach for price time-series analysis that can capture periodic patterns in price time-series. Second, selecting appropriate features and data to represent the environment states can be difficult. For example, manual selection of features and data is challenging due to the large search space (e.g., fundamental, and technical indicator data) [9]. Finally, machine learning algorithms have a complex structure and a large number of different parameters [4]. Reducing the number of parameters results in simplifying the tracking and interruption of the trading performance results.

This study extends the Alkhamees and Aloud [7], where a DCRL model was introduced to detect directional price changes in price time-series. The proposed DCRL model is considered an alternative approach to the traditional time-series analytical approaches for environment state representation. Basically, these traditional approaches are based on fixed time interval analysis, in contrast, the DCRL model samples price time-series under intrinsic time. The DCRL model also learns the states of the price time-series to find the optimal dynamic threshold for DC event analysis. The dynamic DC threshold was introduced [13] to replace the fixed DC threshold, which is used to identify DC events (e.g., directional price changes).

This paper develops an intelligent and dynamic algorithmic trading strategy using the proposed DCRL model, specifically, we present two algorithmic trading strategies where the first is a direct RL approach and the second additionally incorporates a RL Q-learning algorithm. Essentially, the proposed DCRL algorithmic trading employs the DC event approach with the dynamic DC threshold to derive the state representation in RL. In addition, it uses the RL decision-making algorithm to make decisions and take the most appropriate trading action.

The DCRL algorithmic trading strategies were evaluated using real financial market data for stock trading. We conducted a series of systematic experiments to confirm the effectiveness and interpretability of the trading performance results. Therefore, we selected three common US stock indices to evaluate the performance of the DCRL algorithmic trading strategies (with and without Q-learning) and compare their performance against zero-intelligence (ZI) trading agents. The experimental results demonstrate that the DCRL algorithmic trading strategies are effective in different market situations and can potentially generate profits.

Our primary contributions are summarized as follows. First, we contribute to the financial market literature by designing and developing an algorithmic trading strategy that is suitable for stock markets by improving the RL environment state representation and action decision-making to ensure stable trading returns even in the case of volatile price time-series. Second, we contribute to the application of the DC event approach for the representation of the environmental states in RL. The proposed algorithmic trading considers sequential DC event recognition in the price time-series process using the dynamic DC threshold. This model can support decision-makers to determine optimal trading opportunities to maximize profits. Finally, we contribute to the literature by using the Q-learning algorithm to improve the learning process via the prior gained experience, and we capture long-term learning and continuous improvements via the Q-learning algorithm to achieve optimal policies under different market states.

The remainder of this paper organized as follows. Section 2 presents a brief discussion of literature related to the RL algorithm in financial trading. Section 3 provides a brief description of the DC event approach and the definition of the dynamic DC threshold. Section 4 describes DCRL algorithmic trading and the Q-learning algorithm. Section 5 presents the datasets, experiment settings, profitability results, and discusses the empirical results. Section 6 concludes the paper and presents suggestions for potential future work.

II. RELATED WORKS

Several works in financial and machine learning literature have exploited RL in different financial market studies, e.g., financial signal representation [4,7,14], building algorithmic trading [4,8-10, 15, 16], portfolio management [11, 17, 18], optimizing trade execution [19], Foreign Exchange (FX) asset allocations [20], changes in market regimes [11], and stock market modelling [21, 22]. Building algorithmic trading using RL has been the focus of many studies for a range of market settings. Some studies have used direct RL [23], while others have employed a value-based RL approach with a Q-Learning matrix to realize algorithmic trading [15, 23,24]. In addition, other studies have used Recurrent RL (RRL) approach [10, 11, 25] or applied a Q-learning algorithm to the design of trading strategies [9, 26, 27]. Furthermore, several recent studies have employed deep RL for financial portfolio management [17, 18].

Serving the literature on algorithmic trading using direct RL. Bertsimas and Lo [23] examined an application of the RL algorithm for trading a large block of equity over a fixed time horizon to minimize the expected cost of executing trades. They identify optimal trading rules (i.e., executed actions) as a strategy that evolves over a few days. Their experimental results demonstrated that the RL strategy saved between 25% and 40% of execution costs compared to the naïve strategy. However, this study's main drawback was the assumption that the quantity of each buy order is significantly high to increase the price of the traded security. The work by [22] designed a next-generation multi-agent systems (MAS) stock market simulator. Each agent learns price forecasting and stock trading autonomously via RL. The results demonstrate that agent learning allows accurate simulation of the market microstructure.

Several studies in the literature utilized a value-based RL approach with a Q-Learning matrix for algorithmic trading. Gao and Chan [24] and Pendharkar and Cusatis [15] employed a value-based RL approach with a Q-Learning matrix to develop algorithmic trading methods. Here, the core idea is to approximately calculate each state's value function (or state-action pair) and subsequently select the greedy trading action based on the value function. [24] used two performance functions, i.e., absolute profit and relative risk-adjusted profit, to train the algorithmic trading model. The authors in [15] proposed several RL agents for trading portfolio assets. They designed on-policy (SARSA (λ)) and off-policy (Q-learning) discrete state and discrete action agents. Here, the goal is to maximize one of the two values the portfolio returns or differential Sharpe ratios. They examined the impact of RL and trading frequencies. The results demonstrate that a continuous adaptive action RL trading strategy consistently performs the best in forecasting portfolio allocations in the following period. The learning frequency of RL algorithmic trading is essential in determining trading performance. The work by [9] and [20] demonstrated the effectiveness of the policy-based model over the value-based function model relative to performance and applicability.

With regard to the adoption of Q-learning, Neuneier [26] applied a Q-learning algorithm to optimize a trading portfolio. Neuneier constructed an Artificial Neural Network (ANN) to forecast price movement and then used the Q-learning algorithm to find an optimal policy. Another study [27] proposed a portfolio optimization technique using the RL Q-learning approach. This method improved the Q-learning algorithm for optimal asset allocation introduced [26]. This model simplifies the previous model [26] by using one value function for several assets, facilitating model-free policy iteration. Another study [9] used direct RL alteration and compared their algorithm to Q-learning and temporal difference algorithms using real data. Their results demonstrated that the deferential Sharpe ratio RRL system outperformed the Q-learning algorithm. Carapuço et al. [28] developed an RL trading system to trade in the foreign exchange market. They used ANNs with three hidden layers, where the neurons were trained as RL agents under the Q-learning algorithm using a simulated market environment framework. The framework was tested using EUR/USD market data from 2010 to 2017 with more than 10 tests with different initial conditions, and an average total profit of 114.0% ± 19.6% was achieved.

Other literature studies have used the Recurrent RL (RRL) approach. Moody et al. [10] proposed an application of the RRL approach. RRL is an unconstrained RL algorithm that solves the problem of dimensionality. Several studies have extended the RRL model. For example, Zhang and Maringer [25] used technical analysis indicators, fundamental analysis, and econometric study with RRL to improve trading decisions. The analytical indicators were filtered using the genetic algorithm evolutionary process. [8] combined RRL and a particle swarm with a Calmar ratio-based objective function for portfolio trading. They evaluated their method using S&P100 index stocks, and the results demonstrated that the proposed portfolio trading system outperformed benchmark trading strategies, particularly under high transaction cost conditions. In addition, the results demonstrated that the Calmar ratio was the best fitness function for particle swarm algorithms.

In recent years, RL research has clustered around deep learning RL. [17] used a financial-model-free RL framework to deliver a deep RL solution to the portfolio management problem. The central part of the deep RL framework is the Ensemble of Identical Independent Evaluators (EIIE) topology. An EIIE is a neural network designed to examine the historical data of an asset and evaluate its potential growth. In their work, the portfolio weights identify the action for the RL agent. The reward of the RL agent is the explicit average value of the recurring logarithmic returns. In a similar context, [18] offers a portfolio management approach using deep RL on markets with a dynamic number of assets. The neural network architecture is employed and trained using deep RL. Their design was tested on a historical dataset of one of the largest world cryptocurrency markets. The results outperform state-of-the-art methods in the literature, accomplishing average daily returns of over 24%.

The main advantage of the algorithmic trading strategies proposed in this paper is their continuous adaptability to new market conditions using a learning process resulting from dynamic DC events. In addition, existing RL algorithmic trading modules does not consider an event-based system, where an event is the basic unit for studying price time-series. Thus, the representation of environmental states (i.e., market states) in RL algorithmic trading must be improved to realize adaptability to market behaviours continuous changes.

III. Dynamic DC Event Approach

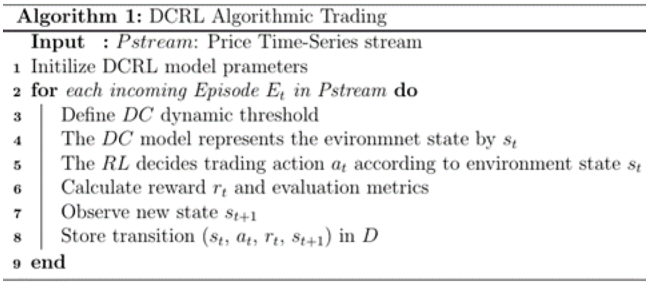
DC is an event approach for price time series analysis in financial markets [29]. The DC summarizes price movements using intrinsic time, which is an event-based timing system (irregularly spaced in time), rather than physical time, which is a point-based timing system that depends on fixed time intervals (e.g., hourly). DC identifies significant price movements (i.e., DC events) if the price change between two points satisfies a fixed given threshold value. DC has two types of events, i.e., upturn events, which are identified once the price change is greater than or equal to the fixed threshold value, and downturn events, which are identified once the price change is less than or equal to the fixed threshold value. A DC event detection algorithm with a comprehensive description of DC can be found in the literature [29]. The potential of DC event approaches has been proven in many studies for different financial markets, and they have been used for event detection [7, 13], forecasting models [30, 31], designing trading strategies [30, 32-40], profiling price time-series [31], and time-series analysis [41, 42].

A dynamic threshold definition method that replaces the DC fixed given threshold value has been proposed in [13]. The dynamic threshold is applicable to financial markets that operate over specific opening and closing times, e.g., stock markets. The dynamic threshold is a flexible value that can identify significant price movements (i.e., DC events) of different magnitudes in continuously changing and dynamic environments. Setting the dynamic threshold value depends on the previous day's price behaviors (i.e., the short-term price history) and additional data sources (e.g., news outlets) [13]. In general, the dynamic threshold is set depending on the previous day (between opening and closing price) price changes, and the overnight (between previous day closing price and current day opening price) price changes. In addition to the rate of change in price between the current day opening price and the reached trend price. The dynamic threshold definition equations can be found in [13].

In the work by [7], we were able to set the dynamic threshold value-based on only financial market data (without any additional data sources) using RL. In this study, to identify an actionable trading opportunity, we use the DC event approach and the dynamic threshold definition method [7].

IV. DCRL Algorithmic Trading

The proposed DCRL algorithmic trading comprises two main components. First, the DC event approach with the dynamic DC threshold is used to identify and represent the market's environmental states. Second, the RL decision-making algorithm is used to make decisions and take appropriate trading actions. In this section, the DCRL algorithmic trading strategy is described in detail. In addition, a rigorous formalization of this algorithmic trading approach and its associated characteristics are presented. Algorithm 1 depicts the core mechanism of the DCRL algorithmic trading.



Algorithm 1: DCRL Algorithmic Trading

```
Input : $Pstream$: Price Time-Series stream
Initilize DCRL model prameters
for each incoming Episode $E_t$ in $Pstream$ do
    Define $DC$ dynamic threshold
    The $DC$ model represents the evironmnet state by $s_t$
    The $RL$ decides trading action $a_t$ according to environment state $s_t$
    Calculate reward $r_t$ and evaluation metrics
    Observe new state $s_{t+1}$
    Store transition ($s_t$, $a_t$, $r_t$, $s_{t+1}$) in $D$
end
```

Algorithm 1. The core mechanism of the DCRL algorithmic Trading.

A. RL

RL comprises an agent, an environment represented by a set of states $s_t \epsilon S$, and a set of actions $a_t \epsilon A$. By performing an action at time t, the agent receives reward r_t and transforms from state s_t to state s_{t+1}. Here, the goal of the agent is to maximize its current reward. The RL algorithm creates models using the Markov decision process (MDP); thus, RL can be represented by a state, action, and reward sequence (s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+2},…). The RL algorithm designs policies (π) that associate an environment state *s* with action *a* to maximize the immediate reward r received over a specified time. In a trading algorithm, a trading rule can be considered a programmed policy π *(s, a)* that yields a trading action according to the available state data s_t at time *t*. As shown in Figure 1, the RL algorithm is based on the sequential interactions between the agent and its environment.

The RL algorithm includes two value functions, i.e., the function of states and the function of state-action pairs [5]. These functions estimate the effectiveness of an agent's action in a given state. The notion of "effectiveness" in RL is defined according to future rewards, i.e., the expected return in a financial trading context. Thus, these value functions are determined based on specified policies. The value of state *s* following policy π (denoted $v_\pi(s)$ is the expected return when starting in *s* and following π through the specified period. For the MDP, we can define the state-value function $v_\pi(s)$ for policy π as follows in Eq. 1:

$$v_\pi(s) = E_\pi[G_t|S_t = s] \tag{1}$$

where $E_\pi[.]$ is the expected value following policy π, and t is any time. Here, G_t is the cumulative discount rate for state *s* at time *t*, which is defined as follows in Eq. 2:

$$G_t(s) = [\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | \, S_t = s] \tag{2}$$

where gamma (γ) is a discount factor that takes a value between 0 and 1. Discount factor (γ) defines the importance of future rewards and weighs recent rewards more heavily. In algorithmic trading, a higher discount factor value implies that the agent will become more long-term investment oriented. For example, in the ultimate case of $\gamma = 1$, the agent considers each reward equally through the market run. In contrast, for γ=0, the agent is biased because it only reflects the current reward and discards future rewards.

Similarly, we define the function of a state-action pair *Q(s, a)*. The value of taking action *a* in state *s* following policy π (denoted $Q_\pi(s, a)$) is defined as the expected return starting from *s*, taking action *a*, and subsequently following policy π. The action value function for policy π is expressed as follows:

$$Q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a] \tag{3}$$

where G_t is the cumulative discount rate for all actions in state *s* at time *t*, which is defined as follows.

$$G_t(s) = [\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | \, S_t = s \, , A_t = a] \tag{4}$$

In RL, there are two main algorithms designed to find optimal action a_{t+1} to take given current state s_{t+1}. The first algorithm is the off-policy algorithm, where the *Q(s,a)* function does not depend on the agent's learning policy: thus, it learns from taking different actions (e.g., random actions). The second algorithm is an on-policy algorithm, where the *Q(s,a)* function is dependent on the agent's learning policy; thus, the agent learns from actions it has taken using the current policy $\pi(a|s)$.

B. DCRL States

The principle of RL is that an agent continuously interacts with the environment and learns the optimal trading rule to improve its trading strategy. For stock market trading, the environment comprises the current stock price data and historical price series, including a variety of fundamental data and technical analysis indicators. Therefore, selecting the set of data inputs is a prerequisite for trading agents to learn the stock market environment and discover trading rules. The underlying challenge of stock market trading is capturing market states at a specific time. For price time-series, commonly employed data in financial forecasting literature represent the price sequence at regular time intervals (e.g., daily data). In this study, we used the daily data of stock market indexes, i.e., the opening, closing, high, and low prices for each day.

The market state variable of each trading day is represented by a pair of the DC price trend direction (an upward or downward trend) and the type of detected event (overnight or previous day event). This gives six states for our research problem.-A lookup table (Table I) is established for state representation of the environment, where each state is signified with a single action associated with an expected reward.

The agent uses an RL algorithm to change from state s_t to s_{t+1}, which is based on learning the dynamics of the environment. Thus, if state s_t were Overnight or PreviousDay with an Upward trend, the action would be Sell because we think that the price increase that occurred due to an overnight or previous day price change was high. The same applies if s_t was Overnight or PreviousDay with a Downward trend, i.e., the action would be Buy because we think that prices have fallen sharply due to a sudden overnight or previous day change in price. The Overnight or Previous Day states are satisfied if the five-day moving average is greater than the overnight or previous day's price change. However, if the detected state is Neutral, which indicates no significant event was identified in the price time-series between time *t-1* and *t*, we use the optimal state-action value function to select the optimal policy for *t+1*.

C. DCRL Actions

At each time step *t*, the agent observes the environment's state s_t and executes a trading action following policy π *(s, a)*. Here, the agent actions are buy, sell, or hold, i.e., A= {Buy, Sell, Hold}. An agent receives a reward after it takes an action. An action a_t may have an impact on the agent's portfolio value, specifically, the cash and share values giving that a trading action executes at the current market closure price p_t.

Two experimental design constraints are assumed regarding the quantity of traded shares Q_t at time *t*. First, for the Buy action, the amount of shares to be bought by the trading agent is based on all available cash that agent has at time *t*. For the Sell action, the agent sells all of available shares at time *t*. In other words, the agent spends 100% of its cash when buying and 100% of its shares when selling. Second, there is no transaction cost in this simulation. By making these simplified assumptions, the complexity of the trading strategy is reduced to a level that can be explored and examined within the scope of this study. Simplicity is essential to understand an agents' trading behaviors and the trading rules generated by the agent because assigning variable quantities may result in a more complicated analysis. Note that relaxation of these assumptions does not affect generality or the accuracy of the obtained results. Nevertheless, we are aware of share quantity's critical role as a choice variable for the generated trading rules (especially with risk aversion).

D. Reward Function

An agent designed based on the RL algorithm learns the optimal policy to trade to achieve maximum profit; therefore, the reward function design is critical when designing trading strategies based on the RL algorithm. In stock market trading literature, several studies have used the Rate of Return (RoR) as a reward function [15].

In this study, we used two immediate reward criteria for the DCRL agent. The first criterion is Buy action, where the Relative Return (RR) is used (Eq. 5). Here, p_{Sell} and p_{Buy} are the selling and buying prices, respectively. The RR defined as the difference between the absolute price return at time *t* and the return reached by the target time. The second immediate reward criterion is for the Sell action, where the RoR is used (Eq. 6). The RoR is the net gain (or loss) of a single trade over a particular period based on the trade's initial cost.

$$RR = \ (p_t - p_{t-1})/p_{t-1} \tag{5}$$

$$RoR = \ (p_{Sell} - p_{Buy})/p_{Buy} \tag{6}$$

Here, p_t and p_{t-1} are the current price at time *t* and time $t-1$, respectively. To assess the action taken (i.e., the executed trading action), we employ two reward functions so that we can consider the different impact of both the Sell and Buy actions. The authors of [28] used two reward functions, i.e., the trade profit was used for closing a position (sell action), and the variation of unrealized profit was employed for opening (buy action) or holding a position.

E. Q-Learning Algorithm

Q-learning is an off-policy RL algorithm that seeks to maximize the total reward. Quality in the RL approach signifies how effective an executed action a_t at time *t* was relative to achieving a particular future reward. In the *Q*-learning algorithm, we create a *Q*-table or matrix that follows policy π*(s, a)* and randomly initialize the values in the matrix. Then, for each iteration of the market run, the *Q*-values are updated and stored in the matrix. Accordingly, the *Q*-matrix turn into a reference matrix for the agent to determine the optimal action based on the maximum *Q*-value. The *Q*-function uses the Bellman equation, which takes two inputs, i.e., state s_t and an action under policy π*(s, a)*. Given the current state s_t of the environment at time *t* and the taken action a_{t+1}, we can formulate the action value function following policy π as follows:

$$Q(s,a) = Q(a,s) + \ \alpha[R(s,a) + \gamma \max Q \quad (s',a') - Q(s,a)] \tag{7}$$

where *Q(s,a)* is the new *Q*-value for state s_t and action a_t, α is the learning rate satisfying $0 \leq \alpha \leq 1$, *R(s, a)* is the reward for taking action a_t at state s_t, γ is the discount factor (also referred to as the discount rate) satisfying $0 \leq \gamma \leq 1$, and max *Q' (s', a')* is the maximum expected reward for new state *s'* and all possible actions at state *s'*. Low alpha (α) values imply a slower learning rate, while higher alpha values indicate more rapidly learning of *Q* -value updates.

For simplicity, we refer to DCRL with Q-learning algorithm as QDCRL. A QDCRL agent learns an optimal state-action value function *Q** for the Neutral state, where an update process considers a quintuple $Q(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ of the environment. For the six states and three actions, we create a matrix $Q \in R^{6x3}$ initialized with random values. Therefore, *Q*(*s*,*a*) represents the *Q*-value for state *s* and action *a*. The *Q*(*s*,*a*) initial random values are subsequently updated in the simulation run by identifying new states and actions using the dataset, where reward *r*(*s*,*a*) is assigned for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3105259, IEEE Access

each selected action. The structure of QDCRL algorithmic trading is shown in Figure 2.

V. Experiment and Results

In this section, we discuss a series of experiments conducted with the proposed DCRL (with and without Q-learning) algorithmic trading strategies, including the datasets used, performance evaluation metrics, benchmarks, experimental settings, and trading performance results.

We evaluated three aspects of the proposed DCRL and QDCRL algorithmic trading strategies, i.e., trading performance profitability and effectiveness, as well as adaptability and efficiency of the dynamic threshold DC event approach for the RL environment state representation. Finally, we confirmed the efficacy of the Q-learning algorithm in RL for algorithmic trading.

A. Datasets

We performed several experiments to confirm the effectiveness and robustness of DCRL algorithmic trading using three different stock indices, i.e., S&P500, NASDAQ, and Dow Jones. These stocks were downloaded from Yahoo! Finance for the period July 2015 to July 2020 (five years, i.e., 1260 days of daily price data).

The movement of the three stock indices and their detailed price curve evolution is shown in Figure 2, and Table II shows a descriptive statistics analysis (mean (μ), standard deviations (σ), skewness, kurtosis, minimum, and maximum price values) of the investigated stock indices.

B. Experimental Settings

Here, we define the set of parameters used in our experiments. We examined different parameters settings for the QDCRL trading strategy systematically through several rounds of tuning. The parameter settings are summarized in Table III.

We executed several independent simulation runs using the same configuration values with different random seeds to confirm that the obtained results are consistent. This allowed us to verify the effectiveness and accuracy of the results. Therefore, the experimental results are averaged over 20 independent simulation runs.

For the learning rate (α) and discount factor (γ) parameters in QDCRL, we conducted a series of systematic experiments to explain the effect of these two parameters on the *Q*-learning algorithm and the profitability of QDCRL algorithmic trading. Here, we examined values between 0.85–0.99 for the discount factor and 0.0001, 0.001, 0.05, and 0.1 for the learning rate (these values are in line with previous studies [15, 28, 43]). The Return On Investment (ROI) and Sharpe Ratio (SR) results of the different γ and α values are given in Table IV. The objective is to find the optimal parameter set to be used for the QDCRL. Higher ROI and SR values indicate that a set of parameters achieved the best performance; thus, we selected these parameter sets in our experiments. We found that different discount factor (γ) values do not affect performance significantly. In contrast, small learning rate (α) values demonstrated slightly better performance than higher values, which implies the importance of a slower learning rate for the *Q*-value updates in the *Q*-learning algorithm.

C. Evaluation metrics

We used four metrics to evaluate the trading strategies' effectiveness and robustness: profit curve, ROI, SR, and the number of trading signals. In the literature, two commonly used performance criteria are portfolio returns and differential SR [15]. The four-evaluation metrics are defined as follows.

i. The profit curve is a qualitative metric to evaluate the profitability of trading strategies. The goal is to demonstrate the change in the agent's portfolio profit over time, which can signify the cumulative gain (profit or loss) of the trading strategies at each time point.

ii. The ROI is the ratio between the net trading gains (profit or loss) and trading cost. It is a quantitative evaluation indicator that measures portfolio profitability. Maximizing the ROI is simple, and it reflects a risk-neutral utility function [15].

iii. SR is a quantitative evaluation indicator to measure the risk-adjusted return of the agent's portfolio. The SR considers both the benefits and risks of an investment. Thus, it removes undesirable effects of risk factors on trading performance evaluation. As a result, the SR shows how to fit returns to the risk taken. Here, a higher SR value indicates a higher risk-adjusted RoR. Several studies into RL algorithmic trading have used SRs as a performance measure [8]. However, the SR penalizes price returns that are greater than a specific amount, weighs recent price returns higher than the past returns, and does not differentiate between the upside and downside possible growth of a portfolio [14]. The SR is expressed as follows:

$$SR = \frac{E(R)-R_f}{\sigma(R)}\sqrt{n} \tag{8}$$

where *E(R)* is the expected accumulated return of investments over trading period *T*, *σ(R)* is the standard deviation of the return, R_f is the risk-free factor, and n is the number of observations.

iv. The number of trading signals refers to the number of times a Buy or Sell trading action was executed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3105259, IEEE Access

during the trading period, which is measured to avoid exceedingly frequent trading resulting in extremely high risk.

D. Benchmark Trading Strategy

To further evaluate the performance the proposed DCRL and QDCRL trading strategies, we compare them to the ZI agent with a budget constraint for stock trading. The ZI is a benchmark widely used to evaluate an intelligent algorithmic trading model. It is a naïve approach that allows to assess the intelligence and learning effectiveness of the DCRL and QDCRL. In addition, we benchmark with the Direct RL designed by [9], which is a classical RL model for algorithmic trading. The reason behind choosing the Direct RL model as a baseline benchmark is to provide a rational comparison of the minimum level of supervised learning. Besides, this will allow us to evaluate the dynamic DC event approach's effectiveness in representing the environment's state. Furthermore, we compare the performance of DCRL and QDCRL with a classic DC event approach -fixed threshold- introduced by [12]. The DC approach provides a pattern detection for price time-series with no utilization of any machine learning techniques. We employed the DC approach using a variety of fixed thresholds ranging from [0.01,0.001]. The average performance of the different simulation runs was reported.

A ZI with a constraint trading agent was established [44] to trade in a continuous double auction market. Thus, the ZI agent has no intelligence and does not observe states in the market (i.e., it does not employ a learning process from historical data). As a result, the ZI agent is not informed of the current market conditions and does not have any beliefs about future price movements. The ZI agent places an order based on a random probability defined from a uniform distribution subject to budget constraints. Therefore, the ZI agent decides to either submit a buy or a sell order (or hold) with equal probability. Here, we examined the effect of the ZI trading strategy's randomness compared to the DCRL and QDCRL trading strategies.

E. Trading Performance Results

Our core focus is the profitability performance and effectiveness of the proposed DCRL and QDCRL algorithmic trading strategies; therefore, standard performance evaluations were conducted on three stock indices. Table V summarizes the results of the quantitative assessment using the ROI and SR. We identify three main findings from the results.

First, the QDCRL and DCRL trading strategies generate substantial profits for the three stock indices compared to the Direct RL, ZI and classic DC -fixed threshold- trading strategies. This observation confirms that the dynamic DC threshold effectively contributes to the algorithmic trading design. This is because the dynamic DC event approach summarizes patterns in the price time series. These patterns represent environmental states for the RL algorithm. Instead of employing discrete price values during training of the RL algorithm (e.g., Direct RL), a continuous environmental state signal is fed to the DCRL and QDCRL. The DC environmental states can provide more detailed information regarding the dynamic of price time series. This is also clear in the results of the classic DC trading strategy (as it comes in third place of the trading strategies performance), which also confirms the potential of employing an event based approach for summarizing price time series.

Second, generally, the QDCRL trading strategy outperformed the DCRL trading strategy without Q-learning. The QDCRL trading strategy generally outperformed DCRL on the S&P500 and Dow Jones stock indices in relation to ROI and SR, which indicates the Q-learning algorithm can potentially improve the trading performance resulting on good profits contained by an acceptable level of risk (for NASDAQ, the DCRL trading strategy was only slightly higher than QDCRL ROI).

Third, the QDCRL trading strategy SR values were significantly higher than those for the DCRL, ZI, Direct RL and the classic DC (except for NASDAQ the classic DC SR was slightly higher). Thereby confirming the validity of the trading actions taken by QDCRL, which avoids risk while securing profit especially when the price curve increases sharply. In addition, the high SR values for QDCRL implies that the risk of the QDCRL is more controllable, and that the trading performance results are more effective.

The number of trading signals generated by the QDCRL, DCRL, ZI, Direct RL and the classic DC trading agents for the different stock indices is plotted in Figure 3. For the three trading strategies designed based on the DC event approach (DCRL, QDCRL, and classic DC algorithmic trading), the number of trading signals indicates its sensitivity to price fluctuations in the market as a result of the learning process and adaptability to market changes. Figure 3 shows that the total number of trading signals generated by the QDCRL, DCRL and classic DC trading agents for the three stock indices was significantly less than that for the ZI and Direct RL agents. The average number of trading signals generated by the ZI trading agent for all the three stock indices represents approximately 42% of the total available trading time. As for the Direct RL trading agent, the average number of trading signals represents approximately 50% of the total available trading time. The higher the number of trading signals taking place the more likely its leading to negative investment results.

Figure 4 shows the daily portfolio return for the QDCRL, DCRL, and ZI trading agents during the target period (1260 days) for the S&P500, NASDAQ, and Dow Jones indices. We have excluded the portfolio return for the Direct RL given the massive negative returns during the vast of trading periods. For the S&P500, the DCRL and ZI initially outperformed the QDCRL. After that, we can clearly see how the learning is well reflected in the QDCRL performance, and hence, how the QDCRL has significantly outperformed both DCRL and ZI. The same applies to the Dow Jones (in the third chart), where, initially, ZI and DCRL outperformed QDCRL. However, as learning goes on, the QDCRL significantly outperformed both DCRL and ZI. The same also applies to NASDAQ, where learning has proven to be effective when used with RL and DC. Finally, learning had remarkably effect on QDCRL performance, and QDCRL generally outperformed both DCRL and ZI.

VI. CONCLUSION

In this paper, we have proposed two algorithmic trading strategies based on the DCRL model. Our main focus was to improve the environment state representations for the RL algorithm. The dynamic DC threshold event approach was able to precisely represent the environment states. In addition, it was able efficiently capture stable market states, which led to achieving profitable trading returns under acceptable risk levels in several stock indices. The effectiveness and robustness of the DCRL trading strategies were verified on real stock market data, and the experimental results demonstrate that the proposed DCRL algorithmic trading outperformed the ZI, Direct RL and classic DC trading strategies with higher total profits and SR, as well as more consistent profit curves.

Our primary contributions are summarized as follows. We defined the environment states in the RL algorithm using the dynamic DC threshold event approach, we developed a simple lookup table for RL algorithmic stock trading, and we employed the Q-learning algorithm to select the optimal policy under the Natural market state.

Given the dynamic nature of the price time-series, trained and adaptive algorithmic trading must be retrained when the environment states changes based on specified preconditions. The learning mechanism based on the dynamic DC threshold event approach is effective relative to improving the market's states' representation. The DCRL agents' trading performance (with and without Q-learning) were generally significant and turned a profit within an appropriate level of risk. These results indicate that, to generate proper trading rules and high-performance returns, learning the environment states is required (i.e., adaptive and non-static representations of the price time-series is needed).

We used two reward functions for the DCRL agents, where each reward is associated with a specific action (either a buy or sell action). The relative return reward function was sued for the buy action, and the rate of return reward function was used for sell action. We found that using these reward functions (rather than a single reward function) improved the Q-learning matrix's performance.

There are two reasons why the QDCRL trading algorithm outperformed DCRL. The first is the learning process for the optimal trading policy under specific market conditions. As stated previously, the performance of QDCRL agents depends on the selection of the optimal policy. The learning frequency of algorithmic trading plays a critical role in influencing the agent's trading analytical performance; however, we did not find that large learning rate (α) values are always effective. We consider the difference between loss and reward in the Neutral state was caused by the fact that Q-learning may effectively model the long-term discounted returns of a particular state. In addition, we restricted the agent to select from a finite action set based on the optimal policy, which may permit the agent to submit more trading signals. The results of this study suggest that adaptive QDCRL agents with Q-learning provide the best performance based on investment profitability and are more promising in practical applications.

This paper can be further extended in several research directions. For example, in the future, we can examine DCRL (with and without Q-learning) on high-frequency trading to explore and confirm the effectiveness of DCRL algorithmic trading, thus further improving and optimising DCRL to fit that trading context. In addition, we can evaluate applying DCRL algorithmic trading to different emerging markets, e.g., the Forex market and cryptocurrencies. Finally, DCRL algorithmic trading can only trade one asset at a time; thus, we can also extend our investigations to managing portfolios involving multiple assets.

Acknowledgments

We would like to thank the anonymous reviewers for their useful comments and suggestions. The author thanks Researchers Supporting Project, number (RSP-2020/212), King Saud University, Riyadh, Saudi Arabia for supporting the research project.

Declaration of interest

The authors report no conflicts of interest. The authors alone are responsible for the content and writing of the paper.

FIGURE CAPTIONS

Fig. 1. Interactive process of the RL Algorithm.

Fig. 2. Structure of QDCRL algorithmic trading.

Fig. 3. Price curve movement of S&P500, NASDAQ, and Dow Jones stock indices during target period.

Fig. 4. Comparison of number of trading signals of QDCRL, DCRL, ZI, Direct RL, and classic DC for S&P500, NASDAQ, and Dow Jones.

***Fig. 5.* Profit curves of QDCRL, DCRL, and ZI trading strategies for three stock indices.**

TABLES

TABLE I

LOOKUP TABLE FOR DCRL ALGORITHMIC TRADING

s	*a*	*s'*	*r(s, a, s')*
(Overnight, Upward trend)	Sell	Overnight$_{t+1}$, Previous$_{t+1}$, Neutral$_{t+1}$	Rate of return
(Overnight, Downward trend)	Buy	Overnight$_{t+1}$, Previous$_{t+1}$, Neutral$_{t+1}$	Relative return
(Previous Day, Upward trend)	Sell	Overnight$_{t+1}$, Previous$_{t+1}$, Neutral$_{t+1}$	Rate of return
(Previous Day, Downward trend)	Buy	Overnight$_{t+1}$, Previous$_{t+1}$, Neutral$_{t+1}$	Relative return
(Neutral, Upward trend)	Hold	Overnight$_{t+1}$, Previous$_{t+1}$, Neutral$_{t+1}$	0
(Neutral, Downward trend)	Hold	Overnight$_{t+1}$, Previous$_{t+1}$, Neutral$_{t+1}$	0

TABlE II

DESCRIPTIVE STATISTICS OF STOCK INDICES

Index	μ	σ	Skewness	Kurtosis	Min.	Max.
S&P500	2553.09	379.88	−0.0055	−1.06	1829.08	3386.15
NASDAQ	6775.45	1478.26	0.235	−0.766	4266.84	10767.09
Dow Jones	22608.29	3726.28	−0.227	−1.28	15660.18	29551.42

TABLE III

QDCRL PARAMETERS USED AT TIME *T*

Parameter	Value	Description
Cash	100,000	Initial account size (cash)
Shares	0	The number of units invested per trade (shares)
γ	0.99	Discount factor.
α	0.0001	*Q*-learning rate.

TABLE IV

ROI AND SHARPE RATIO (SR) FOR DIFFERENT PARAMETERS SETTINGS (DISCOUNT FACTOR (Γ) AND LEARNING RATE (A)). RESULTS FOR THE S&P500 STOCK INDEX ARE SHOWN.

γ	*α*	ROI	SR
0.85	0.0001	56.83	2.16
0.85	0.001	55.26	1.93
0.85	0. 05	47.55	1.62
0.85	0.10	53.89	2.16
0.90	0.0001	55.36	1.93
0.90	0.001	53.89	1.7
0.90	0. 05	44	1.59
0.90	0.10	54.9	2.12
0.95	0.0001	58.69	2.4
0.95	0.001	56.83	2.17
0.95	0. 05	38.25	2.13
0.95	0.10	48.81	1.86
0.99	0.0001	60.39	2.63
0.99	0.001	56.83	2.17
0.99	0. 05	50.36	1.6
0.99	0.10	54.29	1.79

TABLE V.

ROI AND SR OF QDCRL, DCRL, CLASSIC DC, ZI AND DIRECT RL TRADING STRATEGIES ON DIFFERENT STOCK INDICES.

	S&P500		**NASDAQ**		**Dow Jones**	
	ROI (%)	**SR**	**ROI (%)**	**SR**	**ROI (%)**	**SR**
ZI	15.32	−1.7	36.06	−1.28	14.89	−1.46
Classic DC	36.33	0.11	40.68	2.36	24.33	0.35
Direct RL	-22.29	-	-32.01	-	-8.77	-
DCRL	47.23	0.77	57.6	1.47	9.16	−0.023
QDCRL	60.36	2.63	53.82	1.96	37.34	2.48

REFERENCES

[1] P. Treleaven, M. Galas and V. Lalchand, Algorithmic trading review, Commun. ACM. 56 (2013) 76–85

[2] B. Bruce. Trading Algorithms, Student-Managed Investment Funds. 2nd ed. Cambridge: Academic Press; (2020):285–315

[3] E. Fama and M. Blume, Filter rules and stock market trading profits, J. Bus. 39 (1966) 226–41

[4] K. Lei, B. Zhang, Y. Li, M. Yang and Y. Shen, Time-driven feature-aware jointly deep reinforcement learning for financial signal representation and algorithmic trading, Expert Syst. Appl. 140 (2020)

[5] R. Sutton and A. Barto. Reinforcement Learning: an Introduction. Cambridge, MA: MIT Press; (1998)

[6] S. Chinchali, P. Hu, T. Chu, M. Sharma, M. Bansal, R. Misra, M. Pavone and S. Katti, Cellular network traffic scheduling with deep reinforcement learning. Thirty-Second AAAI Conference on Artificial Intelligence; (2018):766–74

[7] N. Alkhamees and M. Aloud, DCRL: approach to identify financial events from time series using directional change and reinforcement learning, Manuscript submitted for publication; (2020)

[8] F. Bertoluzzo and M. Corazza. Reinforcement Learning for Automatic Financial Trading: Introduction and Some Applications. Department of Economics, Ca' Foscari University of Venice, (2012) working Paper 2012:33.

[9] J. Moody and M. Saffell, Learning to trade via direct reinforcement, IEEE Trans. Neural Netw. 12 (2001) 875–89

[10] J. Moody, L. Wu, Y. Liao and M. Saffell, Performance functions and reinforcement learning for trading systems and portfolios, J. Forecast. 17 (1998) 441–70

[11] S. Almahdi and S.Y. Yang, An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown, Expert Syst. Appl. 87 (2017) 267–79

[12] J.B. Glattfelder, A. Dupuis and R.B. Olsen, Patterns in high-frequency FX data: discovery of 12 empirical scaling laws, Quant. Fin. 11 (2011) 599–614

[13] N. Alkhamees and M. Fasli, Event detection from time-series streams using directional change and dynamic thresholds. IEEE International Conference on Big Data (Big Data), Boston, MA; (2017):1882–91

[14] Y. Deng and F. Bao and Y. Kong and Z. Ren and Q. Dai, Deep direct reinforcement learning for financial signal representation and trading. IEEE transactions on neural networks and learning systems, (2017) 28(3), 653-664.

[15] P. Pendharkar and P. Cusatis, Trading financial indices with reinforcement learning agents, Expert Syst. Appl. 103 (2018) 1–13

[16] L. Weng and X. Sun and M. Xia and J. Liu and Y. Xu, Portfolio trading system of digital currencies: A deep reinforcement learning with multidimensional attention gating mechanism, Neurocomputing. 402 (2020) 171–82

[17] Z. Jiang, D. Xu and J. Liang, A deep reinforcement learning framework for the financial portfolio management problem. arXiv preprint arXiv: (2017)1706.10059.

[18] C. Betancourt and W. Chen, Deep reinforcement learning for portfolio management of markets with a dynamic number of assets. Expert Systems with Applications, (2021) 164, 114002.

[19] Y. Nevmyvaka, Y. Feng and M. Kearns, Reinforcement learning for optimized trade execution. Proceedings of the Twenty-third International Conference on Machine Learning, Pittsburgh; (2006):1–8

[20] M. Dempster and V. Leemans, An automated FX trading system using adaptive reinforcement learning, Expert Syst. Appl. 30 (2006) 543–52

[21] C. Kuo and C. Chen and S. Lin and S. Huang, Improving generalization in reinforcement learning–based trading by using a generative adversarial market model. IEEE Access, (2021) 9, 50738-50754.

[22] J. Lussange, I. Lazarevich and S. Bourgeois-Gironde and S. Palminteri and B. Gutkin, Modelling stock markets by multi-agent reinforcement learning. Computational Economics, (2021) 57(1), 113-147.

[23] D. Bertsimas and A.W. Lo, Optimal control of execution costs, J. Financ. Markets. 1 (1998) 1–50

[24] X. Gao and C. Laiwan, An algorithm for trading and portfolio management using Q-learning and Sharpe ratio maximization. Proceedings of the International Conference on Neural Information Processing; (2000):832–7

[25] J. Zhang and D. Maringer, Indicator selection for daily equity trading with recurrent reinforcement learning. Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation, Ac.M. (2013) (1757–8)

[26] R. Neuneier. Optimal Asset Allocation Using Adaptive Dynamic Programming. In Proceeding of Advances in Neural Information Processing Systems. Cambridge: MIT Press; (1996):952–8

[27] R. Neuneier, Enhancing Q-learning for optimal asset allocation, Adv. Neural Inf. Process. Syst. (1998) 936–42

[28] J. Carapuço, R. Neves and N. Horta, Reinforcement learning applied to Forex trading, Appl. Soft Comput. 73 (2018) 783–94

[29] M. Aloud, E. Tsang, R. Olsen and A. Dupuis, A directional-change events approach for studying financial time series, Econ. Open Access Open Assess. E-Journal. 6 (2012) 2012–36.

[30] A. Bakhach, E. Tsang and H. Jalalian, Forecasting directional changes in FX markets. IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (IEEE CIFEr'16), Athens, Greece; (2016)

[31] E. Tsang, R. Tao, A. Serguieva and S. Ma, Profiling high-frequency equity price movements in directional changes, Quant. Fin. 17 (2017) 217–25

[32] Ao and E. Tsang, Trading algorithms built with direction changes. IEEE Conference on Computational Intelligence for Financial Engineering and Economics (CIFEr) Conference, Shenzhen, China; (2019)

[33] A. Bakhach, E. Tsang and V.L. Raju Chinthalapati, TSFDC: A trading strategy based on forecasting directional change, Intell. Syst. Acc. Fin. Manag. 25 (2018b) 105–23

[34] N. Alkhamees and M. Fasli, An exploration of the Directional Change Based Trading strategy with dynamic thresholds on variable frequency data streams IEEE International Conference on the Frontiers and Advances in Data Science (FADS), Xi'an, 2017 2017; (2017a):108–13

[35] N. Alkhamees and M. Fasli, A Directional Change Based Trading strategy with dynamic thresholds. IEEE International Conference on Data Science and Advanced Analytics (DSAA), Tokyo, 2017; (2017):283–92

[36] M. Aloud, Directional-Change Event Trading strategy: profit-maximizing learning strategy, in the Seventh International Conference on Advanced Cognitive Technologies and Applications; (2015), F. Nice, ISBN: 978-1-61208-390-2:123–9

[37] M. Aloud, Profitability of Directional Change Based Trading strategies: the case of Saudi stock market, Int. J. Econ. Financ. Issues. 6 (2016) 87–95

[38] M. Aloud, Investment opportunities forecasting: GP-Based Dynamic Portfolio Trading system under directional-change framework, J. Comp. Fin. 22 (2017) 1–35

[39] M. Aloud and M. Fasli, Exploring trading strategies and their effects in the Foreign Exchange market, Comp. Intell. 33 (2017) 280–307

[40] M. Kampouridis and F.E.B. Otero, Evolving trading strategies using directional changes, Expert Syst. Appl. 73 (2017) 145–60

[41] M. Aloud, Time series analysis indicators under directional changes: the case of Saudi stock market, Int. J. Econ. Financ. Issues. 6 (2016) 55–64

[42] J. Ma, X. Xiong, F. He and W. Zhang, Volatility measurement with directional change in Chinese stock market: statistical property and investment strategy, Phys. A. 471 (2017) 169–80

[43] Jeong and H.Y. Kim, Improving financial trading decisions using deep Q-learning: predicting the number of shares, action strategies, and transfer learning, Expert Syst. Appl. 117 (2019) 125–38

[44] D.K. Gode and S. Sunder, Allocative efficiency of markets with zero intelligence traders: market as a partial substitute for individual rationality, J. Pol. Econ. 101 (1993) 119–37.