Expert Systems With Applications 282 (2025) 127540

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Review

A Survey on recent advances in reinforcement learning for intelligent investment decision-making optimization

Feng Wang [a],*, Shicheng Li [a], Shanshui Niu [b], Haoran Yang [c], Xiaodong Li [d], Xiaotie Deng [e]

[a] School of Computer Science, Wuhan University, Wuhan 430072, China
[b] Department of Statistics & Data Science, Carnegie Mellon University, Pittsburgh, 15213, USA
[c] Department of Finance and Risk Engineering, New York University, NY, 11201, USA
[d] College of Computer and Information, Hohai University, Nanjing, 211100, China
[e] Center on Frontiers of Computing Studies, Peking University, Beijing, 100871, China

ARTICLE INFO

Keywords:
Reinforcement learning
Deep learning
Trading strategy
Strategy optimization
Intelligent decision-making

ABSTRACT

Reinforcement learning (RL) has emerged as a powerful tool for optimizing intelligent investment decision-making. With the rapid evolution of financial markets, traditional approaches often struggle to effectively analyze the vast and complex datasets involved. RL-based methods address these challenges by leveraging neural networks to process large-scale financial data, dynamically interacting with market environments to refine strategies, and designing tailored reward functions to achieve diverse investment objectives. This paper provides a comprehensive review of recent advancements in RL for investment decision-making, with a focus on four key areas, i.e., portfolio selection, trade execution, options hedging, and market making. These four problems represent highly challenging instances of multi-stage , multi-objective decision optimization in investment, highlighting the strengths of RL-based methods in effectively balancing trade-offs among different objectives over time. Detailed comparison work of state-of-the-art RL-based methods is presented, analyzing the action spaces, state representations, reward structures, and neural network architectures. Finally, the paper discusses some new challenges and point out some directions for future research in the field.

Contents

* Corresponding author.
E-mail addresses: fengwang@whu.edu.cn (F. Wang), lishicheng@whu.edu.cn (S. Li), shanshun@andrew.cmu.edu (S. Niu), hy2782@nyu.edu (H. Yang), xiaodong.li@hhu.edu.cn (X. Li), xiaotie@pku.edu.cn (X. Deng).

https://doi.org/10.1016/j.eswa.2025.127540
Received 31 December 2024; Received in revised form 25 February 2025; Accepted 1 April 2025
Available online 23 April 2025

1. Introduction

Intelligent investment decision-making optimization has become increasingly vital in today's complex and fast-paced financial markets. As global economies expand and financial instruments proliferate, the ability to make informed and strategic investment choices is important for both individual investors and institutional entities. Traditional investment approaches, often grounded in static models and historical data analysis (Abarbanell & Bushee, 1997; Huang, Capretz, & Ho, 2019), frequently fall short in capturing the dynamic and multifaceted nature of financial environments. These methods struggle to adapt to real-time market fluctuations, manage vast and intricate datasets, and incorporate diverse investment objectives, thereby limiting their effectiveness and scalability.

In recent years, reinforcement learning (RL) has emerged as a powerful tool in decision-making, which enabling agents to learn optimal strategies for complex problems (Chen, Lu et al., 2021). As a kind of machine learning paradigm, the agent of RL iteratively interacts with its environment to maximize cumulative rewards. This approach has proven highly effective in managing uncertain and dynamic settings, showcasing its versatility across a range of applications (Gu et al., 2024). Unlike traditional decision-making methods, RL does not rely on prior knowledge of a fully defined environment model. Instead, it incrementally learns and refines strategies through continuous interaction with the environment, making it particularly well-suited to complex real-world problems (Kiran et al., 2021). Furthermore, a key advantage of RL lies in its ability to balance short-term and long-term objectives. By simultaneously considering the immediate rewards of current actions and the potential future benefits of subsequent decisions, RL enables agents to optimize for sustained success rather than short-term gains. The capability to balance trade-offs between immediate and future rewards helps RL avoid local optima and focus on achieving long-term goals. Thus, these unique attributes have driven RL to significant advancements across various domains, i.e., resource allocation (Ye, Li, & Juang, 2019) and energy management (Li, He, Peng, & Wang, 2019), etc.

The rapid advancements in deep learning over recent years have greatly expanded the application of RL, leading to the development of deep reinforcement learning (DRL). DRL integrates deep learning techniques to employ deep neural networks (DNN) for approximating complex policy and value functions, thereby enhancing decision-making capabilities. Through the reward function of RL, DNN is trained to obtain optimal strategies, allowing DRL to excel in high-dimensional and complex problem domains and significantly improving the practical applicability of RL (Zhou, Tian, Buyya, Xue, & Song, 2024). Real-world decision-making often involves vast and intricate state and action spaces, which present significant challenges for traditional methods. DRL effectively addresses these challenges by utilizing approximation techniques for value or policy functions, enabling it to efficiently search for and learn optimal strategies. This makes DRL particularly advantageous in domains such as game control (Koyamada et al., 2023) and robot control (Sekkat et al., 2024). Furthermore, multi-agent RL extends these capabilities to scenarios where multiple agents interact and learn collaboratively or competitively, providing robust solutions to complex decision-making problems involving cooperation and competition (Vinyals et al., 2019).

Inspired by the promising achievement, numerous RL-based optimization methods have been proposed for solving investment decision-making, offering several distinct advantages. First, financial markets are characterized by complex and nonlinear dynamics that traditional linear models often fail to capture effectively. RL can efficiently deal with the dynamics by leveraging techniques such as deep neural networks for nonlinear function approximation. The capability enables a deep understanding of market behavior, leading to more accurate and smart investment decisions. Secondly, RL demonstrates exceptional adaptability by enabling automatic learning and adjustment of decision strategies in response to evolving market conditions. Through real-time interactions with the market, RL continually refines and optimizes investment strategies, ensuring they remain robust and effective in the face of changing environments. Third, RL provides the flexibility to design customized reward functions tailored to specific decision objectives, i.e., balancing risk and return. This adaptability allows RL-based methods to address diverse decision goals, making them highly versatile for various investment scenarios. Leveraging these advantages, RL-based methods have demonstrated innovative and effective solutions in addressing investment decision-making.

The complexity and diversity of financial markets necessitate the development of tailored RL-based methods for specific application scenarios. To advance the application of RL in investment decision-making, it is critical to systematically analyze and summarize existing research. However, current reviews of RL in investment optimization are limited and lack comprehensive coverage. For instance, Liang, Yang, Wang, and Han (2019) reviewed RL-based financial trading systems but focused exclusively on trading strategies. Millea (2021) summarized numerous works in the field of RL-based methods for investment decision optimization but lacked in-depth analysis of specific financial applications.

Table 1
Summary of RL-based methods for typical investment decision-Making problem.

Problem	Decision frequency	Optimization objective	Environment model	Policy optimization	Data format
Portfolio selection	Daily or longer intervals	Annualized return, sharpe ratio, maximum drawdown	Model-free	Policy-based and value-based	OHLCV (Open, High, Low, Close, Volume) and technical indicators
Optimal execution	Hourly to minute-level data	Implementation shortfall, customized objectives	Model-free	Policy-based and value-based	Order book with price levels
Options hedging	Simulated daily data	Mean–variance Model, Profit and Loss (PnL), Variance of PnL, Conditional Value at Risk (CVaR) of PnL	Model-free	Policy-based and value-based	Simulated pricing data
Market making	Tick-level to intraday data	Profit and Loss (PnL), inventory risk, quote efficiency	Model-based and Model-free	Policy-based and value-based	Order book snapshots, trade-by-trade data, order placement and cancellation Logs

This paper offers a systematic review of recent advances in RL-based investment decision-making optimization, focusing on four representative problems: portfolio selection, trade execution, options hedging, and market making, which are summarized in Table 1. These problems are chosen for their multi-stage and multi-objective nature, requiring the optimization of conflicting goals over time. RL methods are particularly well-suited to address these challenges due to their ability to continuously optimize strategies and balance trade-offs between short-term and long-term objectives. By interacting with dynamic market environments, RL can adapt to real-time changes, improving decision-making across these complex and interrelated investment tasks. By exploring a wide range of applications, the review aims to provide a general perspective on the current state-of-the-art RL-based methods. It examines not only the methodologies and algorithms employed but also the unique challenges and opportunities present in applying RL to diverse financial scenarios. RL-based decision-making methods are typically classified into two categories, i.e., value-based and policy-based methods, based on their optimization mechanisms. Furthermore, they can be categorized as model-based or model-free methods, depending on whether an environmental model is employed for strategy optimization.

The main contributions of this paper are as follows: (1) comprehensive coverage of four core investment problems. We provide a thorough analysis of RL applications across four representative investment decision-making problems: portfolio selection, trade execution, options hedging, and market making, which are represent fundamental aspects of intelligent investment decision-making and (2) categorization and comparative analysis of RL-based methods. We classify RL-based methods into value-based and policy-based approaches, based on how the agent optimizes strategies. We also differentiate between model-based and model-free methods depending on whether an environmental model is used. (3) identification of challenges and opportunities in future research. We highlights the unique challenges faced when applying RL to investment decision-making, and propose actionable recommendations for addressing these challenges, offering insights into potential improvements and future research directions.

The remainder of this article is structured as follows. Section 2 provides an overview of the research background from the perspectives of investment decision problems and RL. Section 3 categorizes and discusses current RL-based methods in the field of intelligent investment decision-making in various research problems. And Section 4 discusses some future research directions in this rapidly developing field. This survey aims to offer a comprehensive review of the domain, potentially inspiring innovative approaches to address existing challenges.

2. Reinforcement learning and investment decision-making problems

In this section, we provide a brief introduction to the foundational concepts and research background, offering a concise overview of RL and prevalent investment decision-making problems.

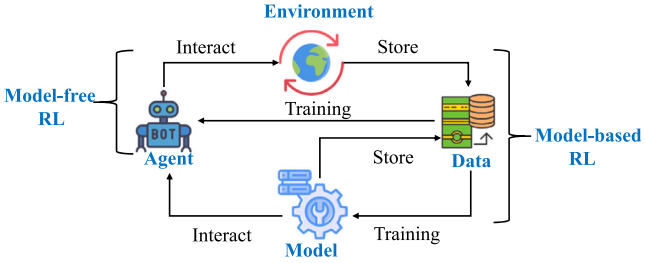

Fig. 1. Framework of model-based RL and model-free RL.

2.1. Reinforcement learning basics

RL is a branch of machine learning used to enable agents to learn optimal strategies through interaction with the environment to achieve predefined goals. The environment and the agent are the two fundamental elements in RL. The agent makes current action selections based on the explored environment states, while the environment transitions states based on the agent's actions and returns corresponding exploration rewards. State space, action space, and reward function are three core concepts.

State Space: The state space, denoted by S, represents the set of all possible states in the environment. A state reflects the environment's current situation, such as $S_t \in S$, which represents the state of the agent at time t. States can represent in various forms, including images, continuous variables, or discrete variables, depending on the specific application. The complexity and observability of the environment are directly influenced by the dimensionality of the state space. At each time step, the agent observes the current state and makes decisions by selecting actions.

Action Space: The action space, denoted by A, comprises all possible actions that the agent can take in the environment. Through these actions, the agent interacts with the environment, triggering state transitions and receiving reward feedback. For example, $A_t \in A$ represents the action taken by the agent at time t. Depending on the specific problem, the action space can be continuous (e.g., controlling the velocity of a vehicle) or discrete (e.g., moving a chess piece). The agent's objective is to choose actions that optimize cumulative future rewards, guiding its decision-making process toward long-term success.

Reward Function: The reward function, denoted by R, defines the agent's immediate feedback from the environment. It assigns a numerical value, either positive (reward) or negative (penalty), based on the agent's actions and their resulting outcomes. For instance, $R_t \in R$ represents the reward received at time t, reflecting how favorable the agent's action was in a given state. The reward function evaluates the agent's performance in achieving its objectives and directly influences its learning process. By associating specific actions and outcomes with corresponding rewards, the function encourages behavior that maximizes long-term cumulative rewards. The effectiveness of the reward

function is vital, as it shapes the agent's strategy and determines its ability to adapt to and optimize complex environments.

RL algorithms can be broadly categorized into two types based on the agent's reliance on an environment model: model-based and model-free algorithms. Model-based RL algorithms leverage an environment model to facilitate planning and decision-making by simulating state transitions and rewards. In contrast, model-free RL algorithms bypass the need for an explicit model and instead learn the optimal policy directly through continuous interaction with the environment.

Furthermore, model-free algorithms can be further divided based on their approach to policy optimization. Value-based algorithms focus on estimating value functions to derive the optimal policy, whereas policy-based algorithms directly optimize the policy itself to maximize cumulative rewards. This distinction highlights the diversity of algorithms within RL and their applicability to different problem domains.

2.1.1. Model-based and model-free reinforcement learning

Model-based RL algorithms focus on constructing an environment model capable of predicting future states and corresponding rewards based on a given state and action. This model serves as a foundation for planning and decision-making, enabling the agent to obtain optimal strategies. To build the model, the agent interacts with the environment, collecting data to learn state transitions and reward structures. Once constructed, the model is used to simulate various state–action scenarios, predicting outcomes and guiding the agent's decision-making process, as shown in Fig. 1. Planning techniques such as value iteration or policy iteration are then applied to derive optimal strategies.

Model-based approaches are particularly advantageous in scenarios where interaction with the real environment is costly or limited, as they allow for extensive simulations to refine strategies, such as Dyna-Q (Peng et al., 2018) and Model-Based Policy Optimization (MBPO) (Pineda, Amos, Zhang, Lambert, & Calandra, 2021). A key advantage of model-based RL is its ability to simulate the performance of different strategies within a virtual environment before deploying them in the real world. This feature enhances learning efficiency by minimizing the need for real-world interactions. However, the performance of these algorithms is highly dependent on the accuracy of the environment model. When the model's predictions deviate from reality, the learned strategies may fail to perform effectively in real-world applications. Moreover, accurately modeling complex environments is particularly challenging due to the complexities and uncertainties inherent in real-world systems. These limitations constrain the applicability of model-based RL, particularly in scenarios that demand highly precise environmental modeling.

Model-free RL offers an different approach in which agents directly learn policies through iterative interactions with the environment, without explicit environmental models. By leveraging trial-and-error experiences, model-free algorithms optimize policies to maximize cumulative rewards, learning independently of prior knowledge about the environment's dynamics. This approach is particularly beneficial in scenarios where constructing an explicit model is impractical or infeasible due to the complexity or unpredictability of the environment.

2.1.2. Policy-based and value-based reinforcement learning

Policy-based RL directly learns a policy function that maps states to action probability distributions over actions. By optimizing the parameters of the policy function, the agent aims to maximize the expected cumulative reward. A parameterized policy is commonly denoted as $\pi(a|s,\theta)$, where s is the state, a is the action, and θ is the parameter of the policy function. The objective of policy-based RL is to learn a policy that maximizes the expected cumulative reward by selecting optimal actions. Classical policy-based RL algorithms include REINFORCE (Williams, 1992), actor-critic (AC) (Bhatnagar, Sutton, Ghavamzadeh, & Lee, 2009), asynchronous advantage actor-critic (A3C) (Mnih et al., 2016), deep deterministic policy gradient (DDPG) (Ye et al., 2020), and proximal policy optimization (PPO) (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017).

Policy-based methods offer robust convergence and generalization capabilities, making them suitable for problems with large or continuous action spaces. However, these methods often require extensive training time, substantial historical experience data, and significant computational resources.

Value-based RL focuses on estimating value functions to solve decision-making problems. The value function represents the expected cumulative reward starting from a given state or state–action pair under a specific policy. Two widely used types of value functions are state value function and state–action value function. State value function $V^{\pi}(s)$ represents the expected cumulative reward starting from a state s under policy π. State–action value function $Q^{\pi}(s,a)$ represents the expected cumulative reward starting from a state s and taking action a under policy π.

Value-based algorithms update the value functions using the bellman equation. For the state value function $V^{\pi}(s)$, its update rule can be represented as:

$$V^{\pi}(s) \leftarrow \mathbb{E}_{\pi}[r + \gamma V^{\pi}(s')] \tag{1}$$

where r is the immediate reward, γ is the discount factor, and s' is the next state.

For the state–action value function $Q^{\pi}(s,a)$, its update rule can be represented as:

$$Q^{\pi}(s,a) \leftarrow \mathbb{E}_{\pi}[r + \gamma Q^{\pi}(s',a')] \tag{2}$$

where a' is the action at the next state s'.

Q-learning (Watkins & Dayan, 1992) is a classical value-based RL algorithm, which iteratively updates Q-values to learn the optimal policy. Extensions of Q-learning, such as Deep Q-Networks (DQN) (Mnih et al., 2015), Double DQN (DDQN) (Van Hasselt, Guez, & Silver, 2016), and Dueling DQN (Wang et al., 2016), have further improved its effectiveness.

Value-based methods typically offer faster convergence and improved data efficiency, leveraging techniques such as experience replay buffers. However, they struggle to evaluate all possible actions efficiently, making them less suitable for the environments with high-dimensional or continuous action spaces. Moreover, these methods rely heavily on accurate value function approximations. Errors in estimating the value function can propagate over time, leading to suboptimal policies.

2.2. Investment decision-making problems

Investment decision-making encompasses the diverse and complex choices made by investors, traders, and financial institutions in financial markets, involving tasks such as selecting investment targets, devising strategies, managing risks, and optimizing returns. These decisions are inherently multi-faceted, requiring careful analysis and judgment to navigate dynamic and uncertain market environments.

The importance of investment decision-making has grown in financial markets, which function as critical platforms for capital allocation, risk management, and price discovery across a wide range of financial instruments, including stocks, bonds, futures, and options. Market fluctuations directly impact key decisions such as portfolio allocation and operational strategies, highlighting the need for accurate, adaptive, and high-quality decision-making to achieve investment objectives and mitigate risks effectively.

As mentioned above, four representative and challenging investment decision-making problems: portfolio selection, trade execution, options hedging, and market making, are involved for discussion in this survey. These four problems represent fundamental aspects of intelligent investment decision-making. Each one addresses a critical task in the investment process, ranging from asset allocation (portfolio

selection) to managing market risk (trade execution, options hedging) and ensuring liquidity (market making). Together, they encompass the primary challenges that investors and financial institutions face in real-world financial markets.

These problems are prototypical examples of multi-stage, multi-objective decision optimization. Each involves sequential decisions where actions taken at one stage influence future outcomes and requires balancing conflicting objectives, such as maximizing returns, minimizing risks, and reducing costs. These characteristics make them highly representative of the complexities faced in investment scenarios and ideal benchmarks for evaluating and advancing RL-based methodologies.

These problems are highly interrelated in practice. These correlations can be understood in terms of how they interact within a broader investment. For example, portfolio selection involves determining the optimal allocation of capital across various assets to achieve a desired risk-return profile. However, the effectiveness of these decisions can be undermined by real-world execution challenges, such as slippage and the impact of large portfolio rebalances on asset prices. Furthermore, market makers play a critical role in providing liquidity, especially for options contracts, which directly links market-making activities to options hedging strategies. The hedging of options positions often requires continuous adjustments, which are influenced by market-making activities and, in turn, impact the broader investment strategy.

These four problems have been the focus of substantial recent research, making them appropriate case studies for discussing the latest advancements in RL for financial applications. By focusing on these well-studied problems, the paper provides a comprehensive view of RL's impact in the financial domain.

2.2.1. Portfolio selection

Portfolio selection constitutes a multi-stage and multi-objective optimization problem that coordinates asset allocation under dynamic market conditions. Approaches typically operate at daily or longer intervals, balancing trade execution costs with market responsiveness. The optimization objectives encompass three principal metrics: annualized return measuring compounded gains, sharpe ratio quantifying risk-adjusted performance, maximum drawdown assessing downside risk, and etc. Input data formats predominantly utilize OHLCV (Open, High, Low, Close, Volume) time series augmented by technical indicators (e.g., moving averages, relative strength index), which collectively capture price momentum and liquidity patterns.

Traditional theories of portfolio selection are broadly classified into single-period models and sequential allocation models. The single-period portfolio selection model, introduced by Markowitz and Markowitz (1967) through the mean–variance framework, employs mathematical optimization to address portfolio allocation. The Black–Litterman model (Black & Litterman, 1992) later extended this approach by incorporating prior beliefs to address parameter estimation challenges.

Conversely, sequential allocation models are rooted in the capital growth theory (Hakansson, 1971), which emphasizes long-term returns and risk management while accounting for rebalancing costs. Algorithms in this category include follow-the-winner, follow-the-loser, pattern recognition, and meta-learning approaches (Li & Hoi, 2014). However, these algorithms often rely on assumptions about asset time series patterns, which may not align with real-world markets, reducing their adaptability and effectiveness.

2.2.2. Optimal execution

Optimal execution represents a temporal coordination challenge in financial markets where large orders must be fragmented into suborders to mitigate market impact. This process operates at hourly to minute-level frequencies to balance price slippage risks with opportunity costs. The optimization objectives primarily focus on minimizing implementation shortfall which is defined as the difference between decision price and executed price, while accommodating institution-specific constraints through customized penalty functions (Almgren & Chriss, 2001; Donnelly, 2022). Input data formats require granular order book records containing price levels, market depth, and hidden liquidity indicators, which collectively enable microstructure-aware execution scheduling.

Traditional trade execution strategies include Volume Weighted Average Price (VWAP) (Madhavan, 2002) and Time Weighted Average Price (TWAP) (Noh & Weston, 2022). These methods aim to execute orders at the average market price over a specific period, which are simple and practical.

Advanced models, such as the Almgren–Chriss model (AC model) (Almgren & Chriss, 2001) and the Adaptive Arrival Price model (Almgren & Lorenz, 2007), extend these strategies. The AC model uses mathematical methods to minimize trading costs under certain market assumptions, while the Adaptive Arrival Price model dynamically adjusts trading volume based on liquidity. However, traditional models often fail to consider dynamic market conditions beyond liquidity, performing bad in real-world scenarios.

2.2.3. Option hedging

Option hedging constitutes a dynamic risk management process that calibrates positions between derivatives and underlying assets to neutralize portfolio sensitivities (Hauser & Eales, 1987). Contemporary strategies predominantly utilize simulated daily data to account for market incompleteness and transaction cost uncertainties. The optimization objectives include: Mean–variance efficiency balancing expected Profit and Loss (PnL) with its variance, tail risk minimization via Conditional Value at Risk (CVaR) at the setting confidence level, path-dependent PnL stabilization under volatility regimes and etc. Input data formats employ synthetic pricing paths generated through stochastic differential equations (e.g., geometric Brownian motion, Heston model), which enable stress testing across extreme yet plausible market scenarios.

Traditional approaches rely on the Black–Scholes model (Black & Scholes, 1973) for option pricing and employ the Greeks (e.g., Delta, Gamma, Theta, Vega) to determine asset proportions for hedging. Delta hedging (Hull & White, 2017) is one of the most common strategies. To achieve Delta Hedging, the hedge ratio is calculated based on changes in option and asset prices, ensuring that the price movement of one short option corresponds to the movement of δ units of the underlying asset. This creates a risk-neutral portfolio, also known as Delta neutrality. As the prices of the underlying asset and the options change, the Delta value also changes, affecting the number of options required for hedging.

However, the assumptions underlying these strategies, such as constant volatility and negligible transaction costs, often do not hold in real-world markets. Financial markets exhibit characteristics such as heavy-tailed return distributions and varying transaction costs, challenging the effectiveness of traditional hedging strategies.

2.2.4. Market making

Market making constitutes a latency-sensitive optimization process that strategically places bid–ask quotes to capture spread profits while managing inventory exposure. This high-frequency operation requires tick-level to intraday data processing to respond to millisecond-scale market microstructure changes. The optimization objectives triad comprises: PnL maximization through spread capture, inventory risk control via mean-reverting position thresholds, quote efficiency optimization balancing fill rates with adverse selection costs, and etc. Input data formats integrate three critical components: order book snapshots recording depth at price levels, trade-by-trade sequences with execution metadata, and order placement or cancellation logs revealing latent liquidity dynamics.

From an optimization perspective, market-making aims to maximize expected returns while managing inventory and adverse selection risks.

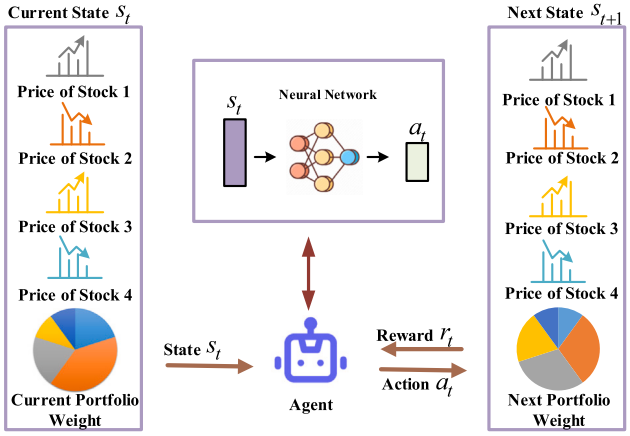

Fig. 2. Framework of RL-based Portfolio Selection Method.

Traditional market-making models employ geometric Brownian motion for price changes, Poisson or Hawkes processes for order flow, and jump processes for spreads. Optimal strategies are often derived by formulating the problem as a stochastic optimization problem and solving it using the Hamilton–Jacobi-Bellman (HJB) equation.

The Avellaneda–Stoikov (AS) model (Avellaneda & Stoikov, 2008) exemplifies this approach, representing optimal bid and ask prices as functions of market parameters. However, these models exhibit notable limitations. Simplistic assumptions, such as static market conditions, lead to rigid frameworks that struggle to adapt to dynamic environments. Furthermore, as financial instruments become more complex, market-making evolves into a high-dimensional optimization problem, which undermines the effectiveness of traditional control methods.

3. Reinforcement learning for intelligent investment decision-making optimization

This section primarily reviews and analyzes the current research on the application of RL-based methods to solve the aforementioned four typical financial market decision-making problems.

3.1. Reinforcement learning in portfolio selection

Compared to traditional methods that rely heavily on market assumptions for sequential decision-making in portfolio selection, RL-based methods provide an end-to-end approach to asset weight generation. These methods also allow for flexible decision-making objectives, demonstrating significant potential. As illustrated in Fig. 2, RL-based approaches reformulate portfolio selection as a task of reallocating asset weights over a multi-asset time series. At the beginning of each trading period, the RL agent evaluates the potential performance of assets, increasing weights for those expected to perform well and reducing weights for weaker ones. In recent years, various RL-based techniques have been proposed. These approaches consider factors such as cumulative returns, transaction costs, and risk preferences by designing reward functions and leveraging neural networks to extract meaningful market information.

Developing accurate environmental models for portfolio selection is challenging due to the complex behavior of asset prices in financial markets. As a result, most existing approaches employ model-free RL methods. This section analyzes and summarizes current research efforts, as detailed in Table 2. These studies primarily use daily stock data from Chinese and U.S. markets and minute-level data from the Bitcoin market. The methods can be broadly classified into policy-based and value-based RL approaches. Early research focused on value-based methods, which utilize neural networks to process asset price series and evaluate discrete investment actions. However, discrete actions lack the precision required for fine-grained control over investment weights. To address this limitation, later studies introduced more sophisticated action spaces that allow for precise investment weight adjustments.

Additionally, policy-based RL methods have been adopted to overcome the constraints of value-based approaches. These methods enable agents to directly output continuous investment weights, allowing for more flexible and dynamic adjustments to portfolio allocations. In policy-based RL, researchers have enhanced performance through two key strategies. First, they employ more efficient neural network architectures to automatically extract essential market features, enabling agents to better understand environmental states and make informed decisions. Second, they design improved reward functions to ensure that the learned strategies are aligned with the requirements of real-world financial markets.

3.1.1. Value-based RL methods

Value-based RL (RL) methods estimate expected returns for each state–action pair and select the action that maximizes these returns to form strategies. These methods are limited to discrete action spaces and require discretization when addressing continuous problems. In portfolio selection, this involves converting portfolio weights into discrete decisions, such as actions to buy, sell, or hold each asset. Common approaches, like Deep Q-Networks (DQN), are used to optimize strategies under these conditions. However, the need for discretization introduces challenges, particularly as the action space complexity grows.

In 1997, Neuneier et al. applied Q-learning to portfolio selection (Neuneier, 1997). Their method used discrete actions for each asset and defined the reward function as the portfolio's net returns after transaction costs. While accounting for transaction costs, risk preferences, and portfolio constraints, this approach often failed to ensure feasible solutions due to the lack of budget constraints.

To address this, researchers proposed improved action space designs. Park et al. introduced a mapping function that converted original actions into feasible solutions (Park, Sim, & Choi, 2020), while Gao et al. divided each asset into N segments, using these as the smallest units for decision-making. Techniques like Double DQN (DDQN) and Dueling DQN further improved action selection precision and reduced action space size (Gao et al., 2020).

Despite these advancements, value-based methods still face scalability issues. As the number of assets increases, the action space grows exponentially, making it increasingly difficult for DQN-based models to find optimal strategies efficiently.

To address the challenges posed by large action spaces in portfolio selection, some researchers have adopted divide-and-conquer strategies and introduced multi-agent RL methods. For example, Lucarelli et al. proposed a multi-agent DQN framework that employs both local and global agents (Lucarelli & Borrotti, 2020). In their approach, local DQN agents extract features from the price series of individual assets and determine corresponding actions. At the end of each time step, these agents calculate the price changes of their assigned assets and the overall portfolio, generating local and global rewards that jointly guide policy updates. Similarly, Gao et al. implemented a hierarchical DQN structure where lower-layer agents manage the weights of two individual assets, while higher-layer agents allocate weights among these lower-layer agents. This hierarchical structure reduces the size of the action space for each agent, improving computational efficiency (Gao et al., 2021).

As financial markets evolve, the diversification of investment portfolios has become an important focus for researchers and financial institutions. Lee et al. introduced a cooperative multi-agent RL method to address the problem of capital concentration in a small number of assets (Lee et al., 2020). In their approach, each agent generates bullish, neutral, or bearish actions, optimizing decisions based on local and global losses. The local loss function follows conventional DQN principles, with the objective of maximizing the Q value. Additionally,

Table 2
Comparison of RL-based methods for portfolio selection.

Category	Reference	Neural network	RL algorithm	Dataset	State space	Action space
Value-based RL	Gao, Gao, Hu, Jiang, and Su (2020)	CNN	DQN	CAH, CAT, CCE, CCL, DIS & Daily data	Four basic price features (Open, Close, High, Low)	Asset weights at minimum unit
	Gao et al. (2021)	CNN	H-DQN	Four stocks in A-share market & Daily data	Four basic price features (Open, Close, High, Low)	Weights by lower agents
	Lee, Kim, Yi, and Kang (2020)	MLP	DQN,PG	Russell 3000 index & Daily data	Closing prices	Sell, buy, or hold
Policy-based RL	Jiang, Xu, and Liang (2017)	CNN, RNN, LSTM	DPG	Top-10 cryptocurrencies on Poloniex & 30-min data	Four basic price features (Open, Close, High, Low)	Asset weights
	Ye et al. (2020)	HAN, LSTM	DPG	Top-10 cryptocurrencies on Poloniex & 30-min data and HighTech & Daily data	Four basic price features (Open, Close, High, Low), news	Asset weights
	Wang, Zhang, Tang, Wu, and Xiong (2019)	LSTM-HA	PG	NYSE, NASDAQ, NYSE Arca, A-shares & Daily data	Trading and fundamental features	Asset weights (long and short)
	Wang, Huang, Tu, Zhang and Xu (2021)	TCN, GCN, LSTM	PG	DJIA, HSI, CSI 100 & Daily data	Price and technical, macro indicators	Asset weights (long and short)
	Xu, Zhang, Ye, Zhao, and Tan (2021)	RAT	PG	Top-10 cryptocurrencies on Poloniex & 30-min data and Kaggle S&P500 & 30-min data	Four price features	Initial, short, invest weights
	Niu, Li, and Li (2022)	TCN, SA	DQN, PG	Dow Jones, HSI, CSI 100 & Daily data	Four basic price features (Open, Close, High, Low) and technical, macro indicators	Asset weights (long and short)
	Sun, Wei, and Yang (2024)	GraphSAGE	PPO	Stocks, bonds, market indexes & Daily data	Financial graph features, SHAP selected	Asset weights (long-term)

the second-to-last network layer outputs a positional confidence score for asset rankings. A global loss function is then applied to these scores to promote decision diversity among agents, ensuring a broad range of asset rankings and improving risk resilience.

These studies demonstrate that multi-agent RL approaches effectively enhance portfolio diversification and strategy efficiency. By increasing the number of agents, these methods achieve greater strategy diversity, leading to improved excess returns and Sharpe ratios.

However, in portfolio selection issues, value-based RL methods require the discretization of the action space, thus they cannot achieve precise control over asset weights. To address this drawback, Huang et al. combined a DQN-based scoring module with a PPO-based portfolio selection module to construct decisions (Huang & Tanaka, 2022). The DQN scoring module outputs single asset trading signals, including three discrete actions: buy, sell, or skip. The PPO module then uses the market information of multiple assets along with the actions output by the DQN scoring module to generate precise asset weight vectors. This method combines the PPO algorithm with the DQN algorithm, effectively compensating for the shortcomings of the DQN algorithm in handling continuous action spaces.

3.1.2. Policy-based RL methods

Policy-based RL methods optimize the agent's strategy directly through gradient information to maximize rewards. This approach is particularly suited to tasks involving continuous and uncertain action spaces, as it allows decision variables to be represented directly as asset weight vectors. These methods are well-suited for portfolio selection problems, where agents determine asset weights for the next period, encompassing both long and short positions. Policy updates are guided by reward functions such as average returns, the Sharpe ratio, and maximum drawdown.

In 2017, Jiang et al. introduced a generic policy-based RL framework for portfolio management called the Ensemble of Identical Independent Evaluators (EIIE) architecture (Jiang et al., 2017). This approach utilizes a standardized price tensor as input and employs Long Short Term Memory (LSTM) networks along with temporal convolution to extract temporal features from asset price data. At the start of each holding period, the agent selects asset weight vectors to maintain until the period ends. By comparing asset price vectors before and after the holding period, the portfolio value and updated weights are calculated. Temporal convolution networks are used to extract time-series features from each asset's price matrix, generating scores for a Softmax layer that computes the final asset weights.

The EIIE architecture demonstrated superior performance compared to traditional online portfolio selection algorithms in practical applications, such as the Bitcoin market. While this approach enables dynamic adjustment of the number of assets by evaluating each asset independently using the same network structure, it has limitations. Specifically, the architecture does not account for correlations between assets, which results in incomplete market information being observed by the agents. Furthermore, the reward functions employed in EIIE focus solely on cumulative returns, neglecting important risk measures relevant to real-world investment decisions.

In recent years, researchers have extended the EIIE framework by introducing enhancements in key areas such as state space and reward functions. Improvements to the state space have been achieved through two main approaches: incorporating additional original features and refining existing feature extraction techniques.

State Space: In terms of representing original features, some works have used methods like Bert to extract textual features from social media and news, enriching the existing state space representation (Koratamaddi, Wadhwani, Gupta, & Sanjeevi, 2021; Ye et al., 2020). Ye et al. proposed a method using a Hierarchical Attention Network (HAN) to predict future stock trends, with news feature information input in the middle layers of the network (Ye et al., 2020). Besides, Nawathe, Panguluri, Zhang, and Venkatesh (2024) enhances the state space representation by integrating financial sentiment data from SEC filings and news headlines and refining the reward function to better align with portfolio performance metrics. By enriching the original features, this method better perceives market information, leading to superior investment decisions.

The main improvements in feature extraction include extracting correlations between assets, different periodic time series features, and long-term correlations. Considering that the EIIE framework does not take into account the correlation between assets, Wang et al. proposed using a Long Short-Term Memory network with a history state attention mechanism (LSTM-HA) to process stock features (Wang et al., 2019). LSTM-HA first uses LSTM to extract features from the historical data of each stock, then further enriches the temporal features through a historical attention mechanism, HA, which better captures features over long intervals. Moreover, this method uses a Cross Asset Attention Network (CAAN) to analyze the correlations of all invested stocks and scores each stock based on this, effectively addressing previous shortcomings in focusing on asset correlations, resulting in better investment strategies than existing benchmarks. It is noteworthy that this approach incorporates a buying-winners-and-selling-losers mechanism by setting fixed ratios for long and short positions, carrying out long and short operations on the highest and lowest-scored assets, respectively. Additionally, it conducts sensitivity analysis to provide rational explanations for the investment decisions made, with agents more likely to choose stocks that are high-growth, low-volatility, high implied value, and undervalued in the recent period.

Building on previous studies, Wang et al. have improved the network structure of the asset scoring module to better extract the temporal relationships of individual assets and the interrelationships between assets (Wang, Huang et al., 2021). They utilized Temporal Convolutional Networks (TCN) to extract the temporal feature information of single assets, and both Spatial Attention (SA) mechanisms and Graph Convolutional Networks (GCN) to extract information about the correlations between assets. SA is used to capture short-term correlation features between assets, while GCN extracts long-term correlation features. With this network structure, agents can extract more asset feature information, achieving higher investment returns. It is important to note that this method also incorporates a market sentiment scoring module, using LSTM-HA to extract features from macroeconomic and sentiment indicators, thereby allowing agents to flexibly adjust the ratios of long and short positions under different market conditions, effectively avoiding substantial losses from sudden financial crises (Millea, 2021).

Stock time series data exhibits both long-term and short-term characteristics. Long-term price data often shows regression tendencies, while short-term price data is influenced by factors such as market sentiment, resulting in localized changes. Traditional models, such as recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) networks, are not fully capable of capturing all these features. To address these limitations, Xu et al. introduced the Relation-aware Transformer (RAT), an improvement to the attention mechanism (Xu et al., 2021). RAT replaces individual moment features with aggregated features within a time window to compute the Query and Key matrices in the attention mechanism. This approach effectively captures both long-term and short-term temporal patterns. Backtesting results, including metrics such as annual returns and Sharpe ratio, demonstrate that RAT outperforms traditional methods, offering higher returns and greater robustness.

Portfolio selection problems, characterized by multi-asset and multi-time series data, pose significant challenges for feature extraction. Recent advancements in deep learning have introduced various techniques to address these challenges. Methods such as Two-stream Convolution, Dilated Convolution (Zhang et al., 2020), Wavenet (Marzban, Delage, Li, Desgagne-Bouchard, & Dussault, 2021), Deep Residual Shrinkage Network (Sun, Jiang, & Su, 2021), and structures based on Temporal Attention and Graph Neural Networks (Li, Cui, Cao, Du, & Zhang, 2022; Shi et al., 2022; Soleymani & Paquet, 2021) have been employed to extract asset features. These approaches leverage temporal dynamics and inter-asset dependencies more effectively. Backtesting results across diverse financial market environments confirm their superior performance, demonstrating that advanced neural network architectures can significantly enhance feature extraction for portfolio selection, enabling more robust and informed investment strategies.

In addition to improving the feature extraction ability of neural network, some scholars combine modern portfolio theory with RL. Jang and Seong (2023) propose a novel approach that uses Tucker decomposition to integrate technical analysis and stock return covariates into a multimodal model. By employing a 3D CNN and Tucker decomposition, the method effectively extracts features from the input data, which are then used to optimize the portfolio through a DRL framework. This integration allows for a more sophisticated handling of multimodal data, leading to improved performance in portfolio optimization.

Reward Function: Agents in portfolio selection must learn strategies that effectively balance maximizing returns while minimizing risks. To address this, researchers have designed various reward functions to integrate risk management into the learning process. Zhang et al. introduced a volatility aversion coefficient into the path reward, enabling direct control over risk exposure during strategy optimization (Zhang et al., 2020). Similarly, Tina et al. incorporated three risk-oriented reward functions into their portfolio selection model: the Sharpe ratio and information ratio, maximum drawdown, and turnover rate (Wang, Pradeep, & Chen, 2022). These reward functions provide a structured framework for balancing return and risk, enhancing the robustness of the resulting investment strategies. Specifically, when directly incorporating a volatility aversion coefficient, the reward function is designed as $R = \frac{1}{T}\sum_{t=1}^{T}\left(\hat{r}_t - \lambda\sigma^2(\hat{r}_t)\right)$. $\hat{r}_t$ is the total return of an asset over a period. When using the Sharpe ratio and information ratio to indirectly control volatility, the reward function is updated as $R = \frac{\frac{1}{T}\sum_{t=1}^{T}\hat{r}_t}{\sqrt{\sigma^2(\hat{r}_t)}}$ or $R = \frac{\frac{1}{T}\sum_{t=1}^{T}(\hat{r}_t - r_b)}{\sqrt{\sigma^2(\hat{r}_t - r_b)}}$ The ratio of the mean excess return to its volatility is used as the reward to guide the agent's updates, with $r_b$serving as the benchmark for comparison. These reward functions guide agents to update strategies that balance the mean excess returns against volatility. Liu et al. incorporated Moody's differential sharpe ratio as part of the reward function, representing the rate of change of the sharpe ratio as a function of the first and second moments of returns (Liu, Liu, Zhao, Pan and Liu, 2020). Additionally, Wang et al. used current period returns as rewards in the asset scoring module and maximum drawdown in the market scoring module, updating different network modules with different rewards (Wang, Wei, An, Feng and Yao, 2021). The study compared the effectiveness of maximum drawdown, sharpe ratio, and information ratio as reward functions in risk control, finding that maximum drawdown is a more reasonable reward function that helps agents better balance returns and risks.

Most recently, hierarchical RL methods have also been applied to solve portfolio selection problems. Considering that most studies have overlooked the issue of slippage in portfolio selection, Wang et al. used hierarchical RL to address this problem (Wang, Wei et al., 2021). The first agent is responsible for constructing the asset weight vector at the macro level of the investment portfolio, while the second agent handles the micro-level transaction execution after asset weight adjustments by reading data from the short-term micro order book. The multi-agent architecture significantly reduces micro-level transaction risks and aligns more closely with real market operations. In practical portfolio selection problems, which often involve a large number of different assets, agents struggle to learn effective strategies due to the vast dimensions of the action space. Zha et al. used hierarchical RL to alleviate this issue (Zha, Dai, Xu, & Wu, 2022), where high-level agents evaluate a large number of assets based on macro asset data to select a subset of assets with higher future returns, and lower-level agents generate portfolio selection weight based on historical microdata of the selected assets. To address the issue of distribution shift between current and future environments, Jiang et al. (2024) introduces a novel Representation Transfer Reinforcement Learning (RTRL) method for portfolio management by addressing the issue of distribution shift between current and future environments. The key innovation lies in extracting a shared high-level representation through pre-training,

which aligns the dynamics of different environments. This approach enhances the robustness of the learned policy by disentangling action representations and fine-tuning for specific tasks. The method theoretically bridges the gap between pre-trained representations and RL, offering a more adaptable and robust framework for portfolio management.

In addition, recent advancements in portfolio optimization have incorporated advanced feature extraction techniques and context-aware RL to achieve superior results. Sun et al. proposed a method that combines GraphSAGE with RL to capture non-Euclidean relationships between assets (Sun et al., 2024). This approach enhances feature extraction and employs SHAP for efficient feature selection, resulting in improved performance across key metrics, including the sharpe ratio and maximum drawdown. Similarly, Yang et al. introduced the Task-Context Mutual Actor–Critic (TC-MAC) algorithm, which uses a Task-Context learning framework to encode both asset features and global context (Yang, 2023). By maximizing mutual information between asset-specific features and contextual data, this method generates optimal portfolio embeddings and improves portfolio management policies. These approaches demonstrate the potential of integrating advanced learning frameworks with sophisticated feature extraction techniques in portfolio optimization.

Despite the successes of RL methods in portfolio selection, significant challenges remain, particularly in reducing noise during the training process. The low signal-to-noise ratio of financial market data makes it difficult for agents to achieve stable convergence toward optimal strategies. To address this issue, researchers have explored the integration of imitation learning to enhance the robustness of RL-based approaches. Liu et al. proposed a behavior cloning module that enables agents to imitate a post-hoc "optimal strategy" for generating asset weights (Liu, Liu et al., 2020). This method introduces a loss function that combines the RL gradient loss and the action imitation loss as a weighted sum, significantly improving the stability and convergence of the learning process. Similarly, Niu et al. incorporated multiple expert strategies as imitation models for agents (Niu et al., 2022). By maximizing RL cumulative rewards and minimizing imitation losses from these expert strategies, agents can acquire diverse sub-strategies. A DQN-based meta-learning module then dynamically selects appropriate sub-strategies based on real-time market information. By integrating expert knowledge into the learning process, these methods not only improve training stability but also enable agents to develop a range of strategy styles, enhancing their adaptability to dynamic and rapidly changing market conditions.

3.2. Reinforcement learning in optimal execution

As mentioned above, RL-based methods can overcome the limitations of traditional ones in trade execution problems. As illustrated in Fig. 3, trade execution is a process of interaction between a large-scale trader and the market. Upon receiving information such as the limit order book (LOB), the agent decides to place an order. Once the order is executed, the environment provides feedback to the agent. Unlike common investment decisions, trade execution strategies are not about profiting from trades but about reducing the costs of trading. Therefore, trade execution requires constant adjustment of strategies based on a series of market factors such as market liquidity. Due to its strong capability for interaction with the environment, RL can play a significant role in solving this problem.

For optimal execution problems, current research methods primarily employ model-free RL methods, which have the following two main advantages: (1) The dynamics of actual financial markets are extremely complex and it is difficult to establish accurate models to describe these changes. Model-free methods do not require modeling of the market; instead, they learn decision-making strategies through direct interaction with the environment, improving the market's uncertainty and variability. (2) In trade execution, the timeliness of decisions is

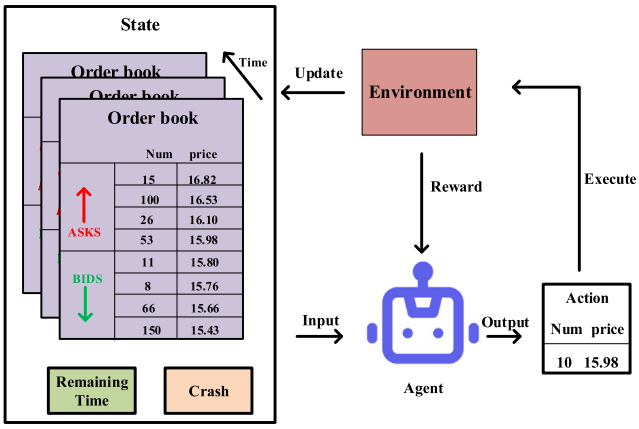

Fig. 3. Framework of RL-based Optimal Execution Method.

crucial. Model-free methods can make decisions more rapidly because they do not use on pre-built models for inference; rather, decisions are made directly based on current market conditions and interaction experience.

This section mainly analyzes and compares RL-based methods currently applied in trade execution from the perspectives of value-based and policy-based approaches, as detailed in Table 3. Compared to portfolio selection problems, RL-based methods for trade execution usually utilize higher frequency datasets, typically at minute or second intervals. In value-based RL methods, the agent evaluates different discrete actions (such as order volumes or order prices) and chooses the action with the highest score as the current investment strategy after receiving state information such as remaining time and outstanding orders. Research in this area primarily focuses on enriching the agent's state space and designing a more rational action space to enhance decision-making performance. Some methods listed do not use neural networks primarily due to the employing of the traditional RL algorithms. Many early RL-based methods, were developed before the widespread use of deep learning. These methods use simpler function approximation techniques, such as tabular Q-values or linear function approximators, rather than neural networks. These methods remain relevant in cases with smaller state or action spaces where deep learning is not necessary due to the simplicity of the problem.

In policy-based RL methods, because a continuous action space is used, the agent can directly output the order price, which aligns more closely with real investment decisions. Besides improving the agent's perception of the state space and using reasonable reward functions to enhance method robustness, researchers have also proposed different RL frameworks based on issues found in the actual trade execution market, such as the smallest quote unit, significant data noise, and information asymmetry, and the games among multiple investors, thus improving the practicality of the methods.

3.2.1. Value-based RL methods

Similar to value-based RL methods used in asset allocation, in trade execution problems, the agent evaluates actions such as order volume or order price at each moment based on the input information, which includes remaining time, remaining assets, asset prices, and trading volumes. Value-based RL methods, such as Q-learning and DQN (Deep Q-Network), can be used to derive optimal strategies.

In 2006, Nevmyvaka and colleagues were the first to use RL to solve trade execution problems (Nevmyvaka et al., 2006), setting the state space inputs as $(t, i, o_1, o_2, \ldots, o_n)$. Here, the private state is related to the remaining time t and remaining assets i, and the public state relates to market variables $o_1, o_2, \ldots, o_n$ that are not influenced by a single trader. The action space was set by the price difference between the ask and bid prices. The reward function was defined as the Implementation

Table 3
Comparison of RL-based methods for optimal execution.

Category	Reference	Neural network	RL algorithm	Dataset	State space	Action space
Value-based RL	Nevmyvaka, Feng, and Kearns (2006)	–	Q-Learning	AMZN, NVDA, QCOM & second level data	Remaining time, remaining assets, market variables	Distance to bid price
	Shen, Huang, Yan, and Obermayer (2014)	–	Risk averse Q-Learning	AMZN & second level data	Remaining time, remaining assets	Distance to bid price
	Lin and Beling (2021a)	DNN	Double DQN, Dueling DQN	NYSE & second level data	Remaining time, remaining assets, LOB, VWAP	Price between 0-2TWAP
	Jeong and Kim (2019)	DNN	DQN	S&P500, HSI, Eurostoxx50, KOSPI & daily Price	Price change	Buy, hold, sell
	Hendricks and Wilcox (2014)	–	Q-Learning	SBK, AGL, SAB & second level data	Remaining time, remaining assets, spread, volume	Price
	Macrìand Lillo (2024)	DNN	Double DQN	Simulated Data	Remaining time, remaining assets, mid-price	Number of share
	Huang, Zhou, Cui, and Lu (2024)	CNN	Double DQN	HSI, FCHI, IXIC, S&P500, DJI	Prices, volume	Sell, hold, or buy
Policy-based RL	Ye et al. (2019)	CNN, DNN	DDPG	3 stocks & second Level Data	Output of feature engineering	Distance to bid price
	Lin and Beling (2021b)	LSTM	PPO	NYSE & second level data	Remaining time, remaining assets, LOB	Price between 0-2TWAP
	Pan, Zhang, Luo, He, and Liu (2022)	CNN	PPO	Simulations from CSI300	Remaining time, remaining assets, LOB, VWAP	Distance to bid price
	Fang et al. (2021)	RNN	Distilled PPO	CSI800 & Minute level data	Remaining time, remaining assets, LOB	Number of share

Shortfall (IS), where a higher selling price of assets leads to greater rewards. This method builds a Q-table using the Q-learning algorithm and selects the action with the highest Q-value to form a decision strategy. However, this method assumes that all investors are risk-neutral, which does not align with actual market behavior. Therefore, Shen et al. considered investors with different risk preferences and designed a new utility function (Shen et al., 2014).

$$u(x) = \begin{cases} \frac{1}{\lambda}[(x+1)^{\lambda} - 1], & \text{if } x \geq 0 \\ x, & \text{if } x < 0 \end{cases}$$

The risk aversion coefficient λ is used in the utility function, where a higher λ indicates a higher preference for risk among investors (Föllmer & Schied, 2011). By incorporating this utility function into the Q-value updates, the agent can exhibit different risk preferences, thus able to simulate investors with varying risk attitudes and perform better when market conditions suddenly worsen.

Given that earlier methods were simplistic in their setup of state spaces, action spaces, and reward function designs, some researchers have made improvements to enhance decision-making performance.

State Space: To enrich the representation of the state space and allow the agent to access more comprehensive information, Lin et al. have included additional indicators such as multiple order book prices, which can be expressed as $ask_{p1}, \ldots, ask_{p5}, bid_{p1}, \ldots, bid_{p5}$ and the volatility of volume-weighted average prices (Lin & Beling, 2021a). Micro-market information can be analyzed through order book prices, providing the agent with detailed information on asset trading at a given moment. Macro-market information can be characterized using representative indicators like trading volume and volatility, aiding the agent in understanding the overall trading trends. Both types of information complement each other and help the agent make better decisions. To address the issue of high dimensionality in the state space, Jeong et al. have used DQN to tackle the curse of dimensionality (Jeong & Kim, 2019), while Ning et al. used Double DQN (DDQN) to avoid the overestimation problems of DQN (Ning, Lin, & Jaimungal, 2021), and Lin employed variants like Dueling DQN for faster convergence (Lin & Beling, 2021a).

Action Space: Unlike asset allocation, optimal execution decisions involve two actions: setting the order price and determining the order quantity. Since a single agent might struggle to output both actions simultaneously, this has been a significant challenge in applying RL to trade execution. Some researchers have addressed this by pre-setting the order quantity, either by planning it in advance (Lin & Beling, 2021a) or by ordering the remaining total assets (Nevmyvaka et al., 2006). However, these methods are limited as they cannot adjust trading strategies dynamically with market changes. To overcome this, more sophisticated strategies have been developed, such as using Deep Neural Networks (DNNs) by Jeong et al. to dynamically determine order quantities (Jeong & Kim, 2019), aiming for optimal trade execution strategies.

Additionally, some scholars have tried combining traditional models with RL to solve trade execution problems. For instance, Hendricks and colleagues criticized the traditional AC model for not accurately reflecting market price-time trends and failing to adjust dynamically as market directions change. By adopting a dynamic strategy concept and using RL to output weights at each moment, they improved on the standard execution volumes and dynamically updated weights, allowing trading strategies to adjust flexibly according to market trends (Hendricks & Wilcox, 2014). This flexibility is essential because the value-based RL methods need to discretize order prices, and the efficiency of algorithms can be significantly impacted by the granularity of discretization and the dimensionality of actions. In practice, accurately controlling the order prices for multiple assets poses a substantial challenge to value-based RL methods. Based on AC model, Andrea et al. introduced time-varying liquidity when using Double DQN considering it is hard and latent to measure in real time (Macrì& Lillo, 2024).

Besides, Huang et al. proposed a novel approach called Multi-Agent Double Deep Q-Network (MADDQN) (Huang et al., 2024), which balances risks and revenues using two different agents based on Double DQN. The findings indicate that MADDQN can perform better with a generalized robustness.

3.2.2. Policy-based RL methods

As discussed earlier, policy-based RL methods typically perform better in solving problems with continuous action spaces compared to value-based methods. For trade execution issues, constructing a continuous action space allows for precise control over order prices.

Researchers like Ye et al. have employed the DDPG algorithm (Ye et al., 2019), Lin et al. used the PPO algorithm (Lin & Beling, 2021b), Pan et al. utilized a hybrid actor-critic architecture with PPO (Pan et al., 2022), and Fang et al. replaced DQN with distilled PPO (Fang et al., 2021), enabling RL to effectively address continuous action space challenges in trade execution.

To improve the performance of policy-based methods, some researchers have further refined the state space and reward functions. In addition, certain studies have perfected the frameworks of RL in trade execution problems to better suit real trading scenarios.

State Space: To enable agents to more effectively extract features from micro price information and market environment data, Ye et al. constructed feature extraction networks using fully connected layers and convolutional layers (Ye et al., 2019). Lin et al. designed two different PPO architectures: PPO Stack and PPO LSTM (Lin & Beling, 2021b). PPO LSTM inputs order book information for each period, while PPO Stack inputs past order book data at each moment. Through the organic integration of these two PPO architectures, the agent can gather market information over a period, rather than just from the previous time segment, allowing for more informed decision-making.

Reward Function: To address the potential instability of using Implementation Shortfall (IS) as a reward function, Lin et al. revised the reward function (Lin & Beling, 2021b). Specifically, the reward is based on the difference in IS at time t between using the PPO algorithm and the TWAP algorithm for executing large orders. If the IS derived from the PPO is less than that from TWAP, the agent receives a reward of -1; if the IS from PPO exceeds 1.1 times the IS from TWAP, the reward is 0; and if the IS is significantly better (more than 1.1 times better), the reward is 1. This method of calculating rewards only at the final trading moment and comparing it with the classic TWAP method makes training more stable.

Frameworks: Pan et al. used a two-stage RL method to address issues related to the minimum quote unit in financial markets (Pan et al., 2022). By selecting a range for the order price using continuous actions and then choosing the final order price within this range using discrete actions, they were able to determine the theoretically optimal order price based on public state information and then refine this price using private state information in the second stage. This "global-local" search strategy enhances solution quality and makes model training more stable. As shown in Fig. 4, a "teacher" model with historical data is trained to establish the best historical trading execution strategies. Then, a "student" model learns from the "teacher" model to overcome the inaccuracies caused by market data noise.

In addition to these advancements, Fang et al. introduced Oracle Policy Distillation (OPD) (Fang et al., 2021), a "teacher-student" learning framework that narrows the gap between existing trading strategies and optimal order execution strategies. As shown in Fig. 4, a "teacher" model with historical data is trained to establish the best historical trading execution strategies. Then, a "student" model learns from the "teacher" model to overcome the inaccuracies caused by market data noise. In practical financial markets, where multiple investors execute trades simultaneously, creating a "trading game", Bao et al. developed a multi-agent DDPG method (Bao & Liu, 2019), MADDPG, which uses a centralized training with decentralized execution (CTDE) approach (Lowe et al., 2017). This method allows multiple agents to train stably and develop diverse strategies by adjusting each agent's reward function to both cooperate and compete, overcoming the limitations of single-agent training instability and policy uniformity.

These studies show that policy-based RL methods, which do not rely on value functions to directly estimate optimal strategies, are more suited for solving trade execution problems, especially in high-dimensional continuous action spaces, compared to value-based methods.

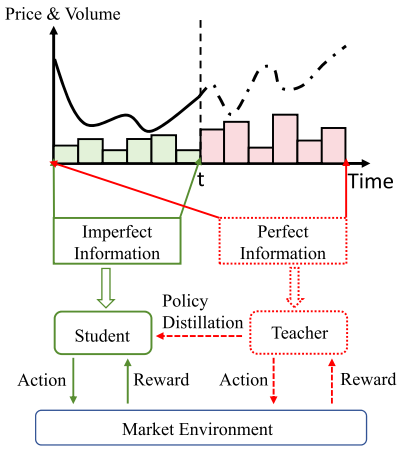

Fig. 4. The framework of oracle policy distillation.

3.3. Reinforcement learning in option hedging

For options hedging problem, traditional methods often fail to effectively balance reducing trading costs and maintaining hedge neutrality. RL, with its flexible reward function design, enables agents to learn strategies that not only reduce transaction costs but also maximize hedge balance. Here are some of the main advantages of using RL in options hedging: (1) dynamic decision making: One of the main features of RL is its ability to adapt to a dynamical environment. In options hedging, market conditions frequently change, such as the fluctuations in asset prices and market volatility. RL can deal with this information in real time and adjust hedging strategies accordingly. It continually learns from market feedback to optimize strategies to fit the current market environment, which is crucial for hedging. (2) risk control: in options trading, risk control is a critical consideration. RL incorporates risk management into its learning process through the design of appropriate reward functions. This means the algorithm is not only focused on maximizing returns but also considers reducing potential risks. As a result, RL-based methods are gaining increasing attention from researchers in the field of options hedging.

Due to the difficulty of building a stable and accurate hedging environment model, current RL-based methods used in options hedging are model-free. These methods can be categorized into policy-based and value-based RL, depending on the learning strategy: policy-based methods focus on directly learning the policy that dictates the agent's actions, typically suitable for environments where the action space is continuous. Value-based methods involve learning to estimate the value of each action, which is more common in discrete action spaces. Researchers have been exploring ways to improve these methods by enlarging the state space, designing more rational action spaces, and creating better-tailored reward functions.

As shown in Table 4, recent studies are listed and analyzed in terms of their neural network structures, RL algorithms, and the designs of their state and action spaces. It is worth noting that due to the limited amount of historical data available for actual options hedging problems, there is a risk of overfitting with these methods. To counter this, current RL-based methods often use simulators that mimic the key distribution characteristics of real samples. This allows the generation of large datasets for training and evaluation, enhancing the generalizability of the methods. Overall, the combination of dynamic adaptation and integrated risk management makes RL a compelling approach for developing sophisticated and effective hedging strategies in the financial sector.

Table 4
Comparison of RL-based methods for option hedging.

Category	Reference	Neural network	RL algorithm	Dataset	State space	Action space
Value-based RL	Halperin (2017)	–	Q-Learning	–	Time, underlying	Holding at next time
	Kolm and Ritter (2019)	–	Q-Learning	Simulations	Time, underlying, holding, option price	Holding at next time
	Cao et al. (2023)	DNN	DQN	Simulations	Time, underlying, holding	Percentage of the position at next time
	Buehler et al. (2019)	LSTM	DQN	Simulations	Underlying, holding	Holding at next time
Policy-based RL	Cao et al. (2023)	DNN	DDPG	Simulations	Time, underlying, gamma, vega	Holding at next time
	Giurca and Borovkova (2021)	DNN	DDPG	Simulations	Underlying and volatility, holding, vega, Delta	Holding at next time
	Du et al. (2020)	DNN	PPO	Simulations	Time, underlying, holding, maturity	−100L, …, 100L, L is the Holding of the option
	Gu (2022)	LSTM	Off-the-shelf PPO	Simulations	Underlying, holding	Change of the underlying percentage
	Sharma, Chen, Noh, DeJesus, and Schlener (2024)	DNN	D4PG	Simulations	Price, Gamma, maturity	Fraction of the max holding

3.3.1. Value-based RL methods

In the problem of option hedging, value-based RL methods involve the agent receiving financial market information such as the price of the underlying asset and the Greeks of the option. Using algorithms like Q-learning and deep Q-learning, the agent scores actions at the current moment, such as holding positions of the underlying asset, using evaluation methods like the mean–variance model, and selects the action with the highest score (usually the holding amount for the next period), thereby achieving a strategy that is low-risk and has stable returns. In 2017, Halperin et al. pioneered a discrete-time option hedging model based on Q-learning (Halperin, 2017). This model assumes that the price of the underlying asset follows a geometric Brownian motion, calculates the option price using the BSM formula (Hauser & Eales, 1987), and constructs the state space based on the random variables of remaining time and price. The action space is constructed based on the amount of the underlying asset to be held in the next period. The reward function is based on the mean–variance model of asset changes, and the Q-learning algorithm is used to update the Q-table to optimize the strategy. Since Q-learning is a model-free method, it can learn hedging strategies from market data with minimal assumptions. Compared to traditional methods, this approach can achieve better hedging strategies to solve problems in option portfolios.

In recent years, with the further activation of the options trading market, many researchers have begun to focus on how to improve the performance of solving option hedging problems with RL and have attempted to make improvements in aspects such as state space, action space, and reward functions.

State Space: Some researchers have expanded the representation of the state space to include richer dimensional information, helping agents make better decisions. For example, Petter et al. have incorporated information such as the price of options into the state space (Kolm & Ritter, 2019), while Cao et al. have added indicators like Vega and Gamma to the state space representation (Cao et al., 2023). Vega and Gamma are metrics that effectively depict changes in asset prices and their volatility. An increase in these indicators means an increase in hedging risks. By incorporating these two metrics into the state space representation and through model training, the final strategy obtained aims to minimize such risks. As the expansion of the state space leads to increased model complexity, training becomes more time-consuming. To address this issue, Buehler et al. (2019) and Cao et al. (2023), have used a combination of deep learning and RL algorithms like DQN to improve the training speed of the model.

Action Space: Unlike the problem of trade execution, in option hedging, the funds required to maintain the hedging strategy may exceed the available funds. Therefore, Cao et al. did not design the action to be the quantity of the underlying asset held in the next period, but rather as the percentage of the required funds relative to total capital (Cao et al., 2023). This design of the action space helps avoid situations where the investor's available funds are insufficient to maintain the hedge.

Reward Function: In the context of option hedging, the reward function setup by Halperin et al. was found to be impractical because it ignored the transaction costs associated with option hedging. To refine the reward function, Petter et al. introduced a dynamic transaction cost model (Kolm & Ritter, 2019), which is defined as:

$$cost(n) = multiplier \times TickSize \times (|n| + 0.01n^2) \tag{3}$$

where 'multiplier' is the transaction fee rate, 'TickSize' is the minimum quote unit, and $0.01n^2$ represents the market impact of the transaction. Incorporating such a transaction cost function into the reward function can reduce the significant costs associated with frequent rebalancing by the agent. Buehler et al. set up a more comprehensive transaction cost function (Buehler et al., 2019), which includes proportional transaction costs, fixed transaction costs, and other transaction costs, better reflecting the costs needed to change hedging positions. Even more, to avoid CVA (Credit Valuation Adjustment) risks in other derivatives, Daluiso et al. address these specific risks by including a risk-averse reward function that balances returns and controls volatility (Daluiso, Pinciroli, Trapletti, & Vittori, 2023), making it more suitable for CVA, which has jump-to-default and market risk components.

3.3.2. Policy-based RL methods

Due to the ever-expanding state space and the continuous nature required for action selection, value-based reinforcement methods no longer meet investors' needs. As a result, some researchers have shifted their focus to policy-based RL methods. For instance, Cao et al. (2023), as well as Giurca and Borovkova (2021), have employed the DDPG algorithm to avoid significant biases in variance estimation directly within the mean–variance model. Du et al. (2020), as well as Gu with others (Gu, 2022), used PPO instead of DQN to address the option hedging problem. Malekzadeh, Poulos, Chen, Wang, and Plataniotis (2024) and Sharma et al. (2024) using distributional RL to handle more complex and more extreme option hedging scenarios.

Among the various policy-based RL algorithms, the PPO algorithm has garnered widespread attention due to its efficient policy optimization capabilities and strong robustness. In recent years, some scholars have attempted to apply the PPO algorithm to solve investment decision optimization problems, designing more reasonable action spaces and reward functions to more effectively address real-world option hedging issues.

Action Space: Du et al. have replaced DQN with the PPO algorithm to address the option hedging problem (Du et al., 2020). They restricted

the action space based on the number of options held, controlling the funds required for hedging within a reasonable range, which allows the algorithm to converge more quickly to the optimal solution.

Reward Function: Gu et al. have made appropriate improvements to the reward function based on the use of the PPO algorithm (Gu, 2022). By introducing a bias term in the reward function, they constrain the agent's rewards. For instance, by adding a suitable indicator function to the reward function, they can correct discrepancies between the option price and its practical meaning, effectively simulate different hedging environments, and enhance the agent's decision-making ability in the actual financial market. The above studies, through an analysis of the characteristics of the option hedging problem and suitable modifications to existing RL frameworks, help agents learn better hedging strategies in real market environments, thereby enhancing decision-making performance.

Some researchers attempted to hedge risks and gain profits not only between stocks and options but also in between other financial derivatives and options, which is much more complex and the traditional RL approaches tend to fail in this context due to the underlying complexity of these products. To address this issue, Sharma et al. utilized Distributional RL, more specifically, Distributed Distributional DDPG (D4PG) (Sharma et al., 2024). In this situation, D4PG outperformed traditional RL methods by estimating return distributions with DQN.

3.4. Reinforcement learning in market making

RL-based methods provide significant advantages for market-making strategies by flexibly extracting market state features and dynamically adjusting decisions in response to evolving conditions. Compared to traditional approaches, RL enables more adaptive and data-driven decision-making. Unlike previous sections which classify strategies as value-based or policy-based, market making presents unique challenges that require a distinct categorization. The primary difficulty is that accurately backtesting market-making strategies using historical data is extremely challenging. This stems from the complexities involved in simulating order book dynamics, liquidity effects, and the intricate interplay between an agent's actions and subsequent market responses.

In classical financial theory, market-making strategies are often derived from theoretical models that prescribe optimal actions under controlled conditions. Model-based RL methods build on these models by constructing mathematical simulations of the market environment. These simulations capture critical microstructural features, such as bid–ask spread variations, order flow dynamics, and inventory risks, allowing agents to use the model to guide their actions and achieve efficient decision-making. However, such methods face limitations due to inherent model inaccuracies and discrepancies between simulated environments and the real world.

To mitigate these issues, researchers have developed model-free RL methods that learn directly from real market data without relying on an explicit environment model. In these approaches, historical data that encompasses detailed order book and order flow information forms the basis for constructing the trading environment. Agents are trained using reward components such as trading profit (PnL), volatility, order execution success, and inventory penalties, with their actions defined as order decisions aimed at transacting at optimal prices. The difficulty of accurately backtesting market-making strategies using historical data, owing to factors like liquidity constraints and the dynamic market impact of orders, necessitates this dual perspective. Consequently, while many other trading strategies can be effectively modeled using purely model-free methods, market making benefits from a combined approach that leverages both model-based and model-free techniques to address its unique challenges in simulating realistic market conditions.

As shown in Table 5, this section analyzes and summarizes RL-based methods in market-making strategies, categorized into model-based and model-free types. In model-based reinforcement methods, since traditional financial models are used and the datasets are simulated, efforts mainly focus on designing different reward functions and employing multi-agent RL frameworks to enhance the diversity and robustness of the learned strategies. In model-free RL methods, historical market data is needed to construct the agent interaction environment, with scholars primarily improving decision-making capabilities from perspectives such as state space, action space, and reward functions.

3.4.1. Model-based RL methods

As previously mentioned, RL-based methods provide increased flexibility in optimizing market-making strategies compared to traditional stochastic analysis frameworks. Model-based RL approaches require financial models of market conditions, such as price dynamics, to construct environments for agent interaction. Research in this area has largely focused on extending traditional financial models and enhancing solution techniques, with relatively little change in the state space (e.g., spreads, price dynamics, and inventory) or action space. To better reflect real financial market conditions, Gašperov et al. incorporated the Hawkes process to simulate the dynamics of order flows, capturing the self-exciting nature of market events (Gašperov, Begušić, Posedel Šimović, & Kostanjčar, 2021). Similarly, Gu'eant et al. extended the Avellaneda–Stoikov (AS) model from single-asset to multi-asset settings, enabling broader applicability in diversified market scenarios (Guéant & Manziuk, 2019). In addition, RL has been used to replace traditional numerical methods, such as finite difference algorithms, to address problems that these methods cannot solve, such as high-dimensional optimal market-making strategies. By integrating RL techniques with extended financial models, researchers have developed solutions that better adapt to the complexities of dynamic market environments, overcoming the limitations of traditional approaches.

In terms of design of RL algorithms, researchers have focused on improving reward functions to achieve diversified strategies and provide greater flexibility in solving optimal market-making problems. By tailoring reward functions, agents can balance competing objectives and adapt to various market conditions. Lim et al. proposed a reward function based on the Constant Absolute Risk Aversion (CARA) utility function, which enables agents to balance returns and risks effectively (Lim & Gorse, 2018). Their approach demonstrated that the strategy learned by the agent outperformed the optimal market-making strategy derived from the Avellaneda–Stoikov (AS) model, showcasing RL's superior flexibility and ability to leverage market information more comprehensively than traditional methods. Similarly, Mani et al. incorporated the Glosten–Milgrom information model from financial theory to simulate the behavior of other market participants (Mani, Phelps, & Parsons, 2019). They defined the reward function as a weighted sum of market-making profits, inventory holdings, and the bid–ask spread, and employed a risk-averse SARSA algorithm to optimize the strategy. This algorithm adjusts the agent's state–action value estimates by overestimating low-reward transitions and underestimating high-reward ones, leading to a robust strategy that achieves consistent profits compared to traditional and risk-neutral approaches. These advancements in reward function design demonstrate the potential of RL methods to produce flexible and robust market-making strategies that outperform conventional techniques.

Considering that there might be multiple market makers with different strategies in actual markets, some researchers have used multi-agent RL to study the competitive and equilibrium issues among different market makers. Ganesh et al. have used multi-agent RL to address the competition and equilibrium problems of market makers with different strategies in the market (Ganesh et al., 2019). Initially, Initially, the methods exposed agents to a simulated environment with different decision-making styles: random, consistent, and adaptive. Furthermore, the action space for the agents was defined as placing orders on both sides of the mid-price and hedging-held positions, allowing them to compete and learn from the aforementioned agents. The study showed that agents could dynamically adjust their market-making strategies

Table 5
Comparison of RL-based methods for market making.

Category	Reference	Neural network	RL algorithm	Dataset	State space	Action space
Model-based RL	Guéant and Manziuk (2019)	2-layer MLP	Actor-Critic	Multi-asset mid-price simulation data	Inventory per asset, order distance from mid-price	Order placement on each asset
	Ganesh et al. (2019)	2-layer MLP	PPO	Mid-price simulation data	Transaction flow, inventory, mid-price, spread, total volume	Orders' distance from mid-price; hedging ratio
Model-free RL	Spooner and Savani (2020)	–	Q-learning, SARSA, R-learning	Simulated order books from 4 industries, 10 stocks	Agent status: inventory, current orders; market status: bid–ask spread, mid-price changes, volume, order imbalance, volatility, RSI	10 predefined order actions
	Sadighian (2019)	MLP	A2C, PPO	Level II data from Bitmex	Order quantity, order flow, order imbalance, other indicators	17 predefined order actions
	Zhong, Bergstrom, and Ward (2021)	–	Q-learning	Globex futures contracts 2019, Level 2 data to seconds	Order book, inventory, cumulative PnL	Constrained discrete actions, posting on buy and sell sides
	Gašperov and Kostanjčar (2021)	2-layer MLP, LSTM	Non-gradient RL via genetic algorithms	Bitstamp 30 days BTC/USD data	Inventory, price range, trend prediction signals	Optimal order distance from the order book
	Chung, Chung, Lee, and Kim (2022)	MLP	Rainbow DQN, PPO	61 days KOSPI200 index futures order book data	Order imbalance, queue information	Four actions regarding buy and sell prices

based on the order placement strategies of other agents and could generate behaviors such as skewed order placements, similar to human market makers, learning effective and reasonable strategies in complex markets with diverse participants.

Spooner et al. proposed an adversarial RL framework to develop more robust market-making strategies (Spooner & Savani, 2020). This approach models the discrete-time optimal market-making strategy selection as a sequential zero-sum game between two agents: one representing the market maker and the other representing the gaming environment. At each stage, the environment agent seeks to minimize the market maker's profits by manipulating the parameters of the market model, while the market maker agent uses RL to maximize its profits by selecting optimal market-making strategies, such as order placement positions. This adversarial setup is particularly effective under conditions of high market volatility, where significant differences often exist between the training and testing datasets. By challenging the market maker agent with adverse scenarios, the method enables it to learn more robust investment strategies, improving decision-making performance in dynamic and unpredictable market environments.

3.4.2. Model-free RL methods

Model-free RL methods have been proposed to address the limitations of model-based approaches, which rely on assumptions about market dynamics and order flows. These assumptions can result in inaccurate representations of the market, leading to suboptimal decisions. In contrast, model-free RL methods bypass the need for explicit financial models by using historical market data to construct an interactive environment for agents. The state space in these methods typically includes both external market conditions and the agent's internal states, while the action space encompasses decisions such as setting order prices and quantities.

Without relying on traditional financial models, model-free RL methods are more aligned with real market dynamics and offer greater flexibility in designing state spaces and reward functions. The state space plays a critical role in ensuring that agents have access to sufficient information to make informed decisions. Similarly, the reward function is pivotal in guiding agents to learn effective market-making strategies. By leveraging the diversity and complexity of real market data, model-free RL methods provide a robust framework for developing adaptive and high-performing market-making strategies.

State Space: The state space for the optimal market-making strategy problem in RL can be composed of multiple features. Due to the complex structure of order books, factors such as order type, size, and position must be considered, often leading to a large dimensionality of the state space that makes it difficult for agents to extract effective information. In practical research, features are typically manually set to represent the state of the order book and the market-making agent. Patel et al. defined a state space that, in addition to basic features such as spreads, also included price, trading volume, and volatility (Patel, 2018). Beysolow et al. divided the state space into personal account status and market state (Beysolow II & Beysolow II, 2019). Personal account status includes current inventory, quotes, and the distance to the order book, while market state includes the spread, middle price changes, order book imbalance measures, trading volume, volatility, and relative strength index (RSI) technical indicators, also incorporating tile coding to expand the representation of features. Sadighian et al. manually constructed a series of features based on order flow imbalances and order book imbalances, effectively capturing the microstructure of the order book and aiding in better decision-making (Sadighian, 2019, 2020). Zhong et al. defined an Aggregation Function to simplify the representation of the state space, constructing a series of indicators such as market order penetration through limit orders, inventory control ratios, middle price drift, and cumulative PnL (Zhong et al., 2021).

Moreover, Gašperov et al. also introduced LSTM to generate market trend prediction signals as part of the state space information, which was more effective in using short-term market trends to profit in simulation backtesting (Gašperov & Kostanjčar, 2021). Compared to directly using all order book features as the state space, these methods effectively reduce feature dimensionality by manually defining the state space, thus lowering the difficulty of model training and enhancing training efficiency.

To meet the trading needs of actual markets, recent studies have used high-frequency data to construct features. Chung et al. (2022) and Zhao and Linetsky (2021) utilized the most detailed tick-by-tick order data (including placing, withdrawing, and execution of orders), constructing a series of features related to order queuing. Zhao and Linetsky (2021) mainly focused on the adverse selection problems in market-making strategies, introducing order queuing features to improve the state space and using policy-based RL for update strategy. By incorporating high-frequency data to expand features, the methods

can make intelligent investment decisions even in real high-frequency trading markets.

Action Space: In market-making strategies, the action space is typically defined to include decisions on the distance between quoted prices and the mid-price, along with custom discrete actions. Since factors such as price, position, and quantity must be considered simultaneously, the action space can become quite large, which poses challenges for efficiency. To address this, simplifying the action space in advance is often necessary. For instance, Zhong et al. reduced the action space to four scenarios: (0,0), (0,1), (1,0), and (1,1), corresponding to whether to place orders on the buy and sell sides at optimal positions. They further constrained feasible actions based on inventory ratios to ensure inventory risk remained within acceptable limits (Zhong et al., 2021).

Building on this, Sadighian (2020) and Spooner and Savani (2020) predefined 10 and 17 types of order placement behaviors, respectively, to represent the agent's action space, restricting actions to positions near the best prices. These predefined action spaces reduce the complexity of training, speeding up convergence while maintaining sufficient flexibility for agents to choose diverse actions. This approach not only simplifies algorithm training but also preserves strategy diversity, enabling agents to adapt effectively to varying market conditions.

Reward Function: Designing reward functions for market-making strategy agents involves balancing multiple objectives: maximizing order execution and market-making profits while controlling inventory risk. To achieve this, RL-based market-making strategies often structure reward functions into several main components: realized gain components (i.e., rewards already obtained through trading), inventory profit and loss penalties (i.e., unrealized profits or losses in the market), and other custom reward components (including incentives for transactions).

Beyond traditional realized and inventory values, Spooner et al. innovatively proposed an asymmetric trimming of the reward function, represented as $R_t = \min(0, \eta \mathrm{UPnL}_t) + \min(\mathrm{RPnL}_t, \kappa)$ (Spooner & Savani, 2020). UPnL represents the profit and loss due to accumulated inventory, and RPnL is the profit and loss from transactions. This trimming effectively prevents agents from pursuing speculative behaviors based on holding gains or trends. Ganesh et al. expanded the penalty components for inventory profits with absolute and quadratic inventory penalties (Ganesh et al., 2019). Moreover, Sadighian et al. proposed three major types of reward functions based on trading profit (PnL), objectives, and risk (Sadighian, 2019, 2020). Beyond position profit and loss, the measure of transaction completion more accurately reflects the quality of single-step actions. This design makes the market-making strategy more inclined to gain trading profits through multiple transactions rather than by following major trends, thus aligning more closely with the actual objectives of market-making strategies.

4. Further perspectives

While existing studies have highlighted the success of RL-based methods in intelligent investment decision-making optimization, this section outlines several promising future research directions. Key challenges are identified, and potential solutions are discussed in detail.

4.1. Benchmark datasets

The advancement of RL-based methods for investment decision-making optimization has principally relied on individually collected market data, leading to a significant lack of standardized benchmark datasets. This lack of standardization creates obstacles for reproducibility and fair performance comparisons across different methods. Developing high-quality, standardized datasets is essential for driving further research in this domain. FinRL (Liu et al., 2022; Liu, Yang et al., 2020) has made valuable contributions by offering frameworks for financial data preprocessing and backtesting environments. Building on these foundations, future research should focus on creating comprehensive, problem-specific datasets to improve the reliability and comparability of various methods in the field.

4.1.1. Data quality

Financial market data is characterized by unique challenges, including high noise, non-stationarity, and non-linearity, which complicate the development of stable backtesting environments. Existing approaches often rely on raw financial data, where the quality of the data plays a critical role in determining the performance of RL-based methods. Issues such as errors, missing values, and outliers in the data can adversely affect the accuracy and stability of methods. To address these challenges, future research should focus on advancing data cleaning and preprocessing techniques to improve data quality, thereby enhancing the reliability and effectiveness of RL-based methods.

4.1.2. Feature selection and multimodal data

Feature selection plays a crucial role in identifying the most relevant and informative attributes from raw data to enhance the performance and generalization of RL-based methods. In investment decision-making, features such as technical indicators, fundamental data, and market sentiment are commonly used. However, many existing approaches rely on empirically chosen features, which may not always optimize model performance. Classical methods (Markowitz & Todd, 2000) and advanced machine learning techniques (Sun et al., 2024) can be useful to address feature correlations, noise, and redundancy, facilitating the automatic selection of representative features.

In addition to feature selection, leveraging multimodal data can further enhance RL-based methods. The growing availability of diverse financial data sources, such as social media, news articles, and economic reports, can provide multimodal data for RL-based methods. For instance, Ye et al. (2020) employed neural networks to extract actionable insights from news data, enabling agents to perceive a more comprehensive state space and improve strategy profitability.

4.2. Multi-agent simulation

Financial market simulation using multi-agent RL is a key research area in investment decision-making. Compared to single-agent systems, multi-agent RL more effectively models the complex dynamics of real-world financial markets. Existing multi-agent RL methods (Bao & Liu, 2019) for solving trade execution have demonstrated their potential. However, more complex environments or real-world settings often face challenges, such as agent homogeneity and limited consideration of individual investor preferences. Some important research topics are briefly introduced below.

4.2.1. Heterogeneity of multi-agents

In Multi-Agents RL for financial market simulations, the assumption of agent homogeneity often limits the model's ability to capture the true diversity of real-world market participants (Wang, Hu, Wu, & Zhao, 2023). Financial markets are inherently heterogeneous, with investors exhibiting distinct characteristics, such as varying risk tolerance, investment strategies, and time horizons. For instance, some agents may focus on long-term value investing, while others may employ short-term speculative strategies. Ignoring these differences reduces the model's realism and can lead to less effective simulations. The assumption of agent homogeneity in existing models simplifies the complexity of real-world investor behavior, making it harder to model the diverse and dynamic interactions seen in financial markets.

Future research should explore the incorporation of agent heterogeneity, where each agent's behavior is characterized by unique attributes and preferences. This could involve designing agents that differ in their risk aversion, investment goals, and decision-making processes. Additionally, the interactions between heterogeneous agents could be studied to better understand competitive and cooperative dynamics, providing a more comprehensive view of how diverse investor behavior impacts market outcomes.

4.2.2. Investor preferences

Investor preferences, including risk aversion, investment horizons, and profit objectives, are essential for realistic financial market simulations. In existing research, these preferences are either assumed to be uniform (Bao & Liu, 2019) across agents or not fully integrated into the decision-making process (Niu et al., 2022). However, real-world investors have distinct preferences that influence their actions in the market. Accurately modeling these preferences is crucial for developing realistic investment decision-making strategies. Many current multi-agent simulations assume homogeneous agent preferences, which limits the ability to simulate the full spectrum of investor behaviors in financial markets. Furthermore, some models fail to incorporate investor preferences directly into agents' reward functions, leading to simplified representations of real-world decision-making.

Future work should embed investor preferences (e.g., risk tolerance, investment goals, and time horizons) into agents' reward functions. By modeling these preferences, researchers can ensure that agents make decisions that align with individual goals, such as maximizing long-term wealth, minimizing risk, or achieving short-term profits. Moreover, exploring how agents with different preferences interact (e.g., risk-averse vs. risk-seeking agents) can offer insights into how diverse investor behaviors influence market outcomes, such as price volatility and liquidity.

4.3. Efficiency of reinforcement learning algorithms

Real-time adaptability and robustness are two critical challenges faced by existing methods. Addressing these challenges is essential to enhancing the applicability and effectiveness of RL in dynamic market environments.

4.3.1. Real-time performance

Online learning involves systems that continuously receive data and update model parameters in real time, gradually refining their performance. For RL-based methods in financial markets, it is crucial to adapt decision-making strategies dynamically in response to evolving market conditions. However, the frequent parameter updates required in high-frequency trading scenarios place substantial computational demands on systems.

A practical method is to limit parameter updates during online solving, and applying updates after collecting enough new data. This strategy aligns with offline learning paradigms (Prudencio, Maximo, & Colombini, 2023) and batch-based RL approaches (Chen & Jiang, 2019; Liu, 2023). Although these methods have been explored theoretically, their practical application within the context of investment decision-making remains underdeveloped. Future research could focus on the implementation and adaptation of these paradigms to financial markets, aiming to enhance the real-time performance of RL algorithms and better support the high-frequency trading environment.

4.3.2. Robustness of algorithms

Current research in RL for intelligent investment decision-making primarily focuses on backtesting results in single-agent scenarios. However, real-world financial environments involve competition from other investors and institutions, which introduces complexities that are not captured in existing backtesting. Consequently, the results observed in simulated environments may diverge significantly from those in actual market conditions. Chen, Chen, Sang, Yang and Huang (2021) demonstrated that RL-based methods lack robustness against deliberately designed attacks, which lead to instability in the strategy.

Future research should concentrate on improving the stability and robustness of RL algorithms, particularly by enhancing training strategies that account for the dynamic nature of financial markets and the potential for adversarial attacks. Strengthening these aspects will help agents better navigate market fluctuations and respond effectively to competitive pressures.

5. Conclusion

In this paper, we provided a comprehensive review of recent advances in RL applied to intelligent investment decision-making optimization, focusing on key areas, i.e., portfolio selection, trade execution, options hedging, and market-making strategies. By comparing action spaces, state spaces, and neural network structures, we discussed and compared various RL-based methods. While this review highlights the most prominent research topics, it is important to acknowledge that the field of investment decision-making encompasses a broader range of applications, many of which were not covered in this paper. Notably, RL has demonstrated significant potential in other financial applications, including market simulation, fraud detection, and credit risk assessment. As research continues to evolve, RL-based methods are expected to enhance their efficacy, driving innovation in investment decision-making and related domains.

CRediT authorship contribution statement

Feng Wang: Conceptualization, Writing – review & editing. **Shicheng Li:** Writing – original draft, Writing – review & editing. **Shanshui Niu:** Literature Collection, Writing – original draft. **Haoran Yang:** Literature Collection, Writing – original draft. **Xiaodong Li:** Writing – review & editing. **Xiaotie Deng:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 62173258, T2495254).

Data availability

No data was used for the research described in the article.

References

Abarbanell, J. S., & Bushee, B. J. (1997). Fundamental analysis, future earnings, and stock prices. *Journal of Accounting Research*, *35*(1), 1–24.

Almgren, R., & Chriss, N. (2001). Optimal execution of portfolio transactions. *Journal of Risk*, *3*, 5–40.

Almgren, R., & Lorenz, J. (2007). Adaptive arrival price. *Algorithmic Trading III: Precision, Control, Execution*, *2007*, 59–66.

Avellaneda, M., & Stoikov, S. (2008). High-frequency trading in a limit order book. *Quantitative Finance*, *8*(3), 217–224.

Bao, W., & Liu, X.-y. (2019). Multi-agent deep reinforcement learning for liquidation strategy analysis. arXiv preprint arXiv:1906.11046.

Beysolow II, T., & Beysolow II, T. (2019). Market making via reinforcement learning. *Applied Reinforcement Learning with Python: With OpenAI Gym, Tensorflow, and Keras*, 77–94.

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., & Lee, M. (2009). Natural actor–critic algorithms. *Automatica*, *45*(11), 2471–2482.

Black, F., & Litterman, R. (1992). Global portfolio optimization. *Financial Analysts Journal*, *48*(5), 28–43.

Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of Political Economy*, *81*(3), 637–654.

Buehler, H., Gonon, L., Teichmann, J., Wood, B., Mohan, B., & Kochems, J. (2019). *Deep hedging: hedging derivatives under generic market frictions using reinforcement learning: Swiss finance institute research paper, 19–80*.

Cao, J., Chen, J., Farghadani, S., Hull, J., Poulos, Z., Wang, Z., et al. (2023). Gamma and vega hedging using deep distributional reinforcement learning. *Frontiers in Artificial Intelligence*, *6*, Article 1129370.

Chen, Y.-Y., Chen, C.-T., Sang, C.-Y., Yang, Y.-C., & Huang, S.-H. (2021). Adversarial attacks against reinforcement learning-based portfolio management strategy. *IEEE Access*, *9*, 50667–50685.

Chen, J., & Jiang, N. (2019). Information-theoretic considerations in batch reinforcement learning. In *International conference on machine learning* (pp. 1042–1051).

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., et al. (2021). Decision transformer: Reinforcement learning via sequence modeling. *Advances in Neural Information Processing Systems*, *34*, 15084–15097.

Chung, G., Chung, M., Lee, Y., & Kim, W. C. (2022). Market making under order stacking framework: A deep reinforcement learning approach. In *Proceedings of the third ACM international conference on AI in finance* (pp. 223–231).

Daluiso, R., Pinciroli, M., Trapletti, M., & Vittori, E. (2023). Cva hedging with reinforcement learning. In *Proceedings of the fourth ACM international conference on AI in finance* (pp. 261–269).

Donnelly, R. (2022). Optimal execution: A review. *Applied Mathematical Finance*, *29*(3), 181–212.

Du, J., Jin, M., Kolm, P. N., Ritter, G., Wang, Y., & Zhang, B. (2020). Deep reinforcement learning for option replication and hedging. *The Journal of Financial Data Science*, *2*(4), 44–57.

Fang, Y., Ren, K., Liu, W., Zhou, D., Zhang, W., Bian, J., et al. (2021). Universal trading for order execution with oracle policy distillation. *vol. 35*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 107–115).

Föllmer, H., & Schied, A. (2011). *Stochastic finance: an introduction in discrete time*.

Ganesh, S., Vadori, N., Xu, M., Zheng, H., Reddy, P., & Veloso, M. (2019). Reinforcement learning for market making in a multi-agent dealer market. arXiv preprint arXiv:1911.05892.

Gao, Z., Gao, Y., Hu, Y., Jiang, Z., & Su, J. (2020). Application of deep q-network in portfolio management. In *2020 5th IEEE international conference on big data analytics* (pp. 268–275).

Gao, Y., Gao, Z., Hu, Y., Song, S., Jiang, Z., & Su, J. (2021). A framework of hierarchical deep Q-network for portfolio management.. In *ICAART (2)* (pp. 132–140).

Gašperov, B., Begušić, S., Posedel Šimović, P., & Kostanjčar, Z. (2021). Reinforcement learning approaches to optimal market making. *Mathematics*, *9*(21), 2689.

Gašperov, B., & Kostanjčar, Z. (2021). Market making with signals through deep reinforcement learning. *IEEE Access*, *9*, 61611–61622.

Giurca, A., & Borovkova, S. (2021). Delta hedging of derivatives using deep reinforcement learning. Available At SSRN 3847272.

Gu, S. (2022). Deep reinforcement learning with function properties in mean reversion strategies. *The Journal of Financial Data Science*, *4*(3), 54–65.

Gu, S., Yang, L., Du, Y., Chen, G., Walter, F., Wang, J., et al. (2024). A review of safe reinforcement learning: Methods, theories, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *46*(12), 11216–11235.

Guéant, O., & Manziuk, I. (2019). Deep reinforcement learning for market making in corporate bonds: beating the curse of dimensionality. *Applied Mathematical Finance*, *26*(5), 387–452.

Hakansson, N. H. (1971). Capital growth and the mean-variance approach to portfolio selection. *Journal of Financial and Quantitative Analysis*, *6*(1), 517–557.

Halperin, I. (2017). Qlbs: Q-learner in the black-scholes (-merton) worlds. arXiv preprint arXiv:1712.04609.

Hauser, R. J., & Eales, J. S. (1987). Option hedging strategies. *North Central Journal of Agricultural Economics*, 123–134.

Hendricks, D., & Wilcox, D. (2014). A reinforcement learning extension to the Almgren-Chriss framework for optimal trade execution. In *2014 IEEE conference on computational intelligence for financial engineering & economics (cIFEr)* (pp. 457–464).

Huang, Y., Capretz, L. F., & Ho, D. (2019). Neural network models for stock selection based on fundamental analysis. In *2019 IEEE Canadian conference of electrical and computer engineering* (pp. 1–4).

Huang, Z., & Tanaka, F. (2022). MSPM: A modularized and scalable multi-agent reinforcement learning-based system for financial portfolio management. *Plos One*, *17*(2), Article e0263689.

Huang, Y., Zhou, C., Cui, K., & Lu, X. (2024). A multi-agent reinforcement learning framework for optimizing financial trading strategies based on timesnet. *Expert Systems with Applications*, *237*, Article 121502.

Hull, J., & White, A. (2017). Optimal delta hedging for options. *Journal of Banking & Finance*, *82*, 180–190.

Jang, J., & Seong, N. (2023). Deep reinforcement learning for stock portfolio optimization by connecting with modern portfolio theory. *Expert Systems with Applications*, [ISSN: 0957-4174] *218*, Article 119556.

Jeong, G., & Kim, H. Y. (2019). Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications*, *117*, 125–138.

Jiang, W., Liu, M., Xu, M., Chen, S., Shi, K., Liu, P., et al. (2024). New reinforcement learning based on representation transfer for portfolio management. *Knowledge-Based Systems*, [ISSN: 0950-7051] *293*, Article 111697.

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

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., et al. (2021). Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, *23*(6), 4909–4926.

Kolm, P. N., & Ritter, G. (2019). Dynamic replication and hedging: A reinforcement learning approach. *The Journal of Financial Data Science*, *1*(1), 159–171.

Koratamaddi, P., Wadhwani, K., Gupta, M., & Sanjeevi, S. G. (2021). Market sentiment-aware deep reinforcement learning approach for stock portfolio allocation. *Engineering Science and Technology, an International Journal*, *24*(4), 848–859.

Koyamada, S., Okano, S., Nishimori, S., Murata, Y., Habara, K., Kita, H., et al. (2023). Pgx: Hardware-accelerated parallel game simulators for reinforcement learning. *vol. 36*, In *Advances in neural information processing systems* (pp. 45716–45743).

Lee, J., Kim, R., Yi, S.-W., & Kang, J. (2020). Maps: Multi-agent reinforcement learning-based portfolio management system. arXiv preprint arXiv:2007.05402.

Li, X., Cui, C., Cao, D., Du, J., & Zhang, C. (2022). Hypergraph-based reinforcement learning for stock portfolio selection. In *ICASSP 2022-2022 IEEE international conference on acoustics, speech and signal processing* (pp. 4028–4032).

Li, Y., He, H., Peng, J., & Wang, H. (2019). Deep reinforcement learning-based energy management for a series hybrid electric vehicle enabled by history cumulative trip information. *IEEE Transactions on Vehicular Technology*, *68*(8), 7416–7430.

Li, B., & Hoi, S. C. (2014). Online portfolio selection: A survey. *ACM Computing Surveys*, *46*(3), 1–36.

Liang, T., Yang, X., Wang, L., & Han, Z. (2019). Review on financial trading system based on reinforcement learning. *Journal of Software*, *30*(3), 845–864.

Lim, Y.-S., & Gorse, D. (2018). Reinforcement learning for high-frequency market making. In *ESANN 2018-proceedings, European symposium on artificial neural networks, computational intelligence and machine learning* (pp. 521–526).

Lin, S., & Beling, P. A. (2021a). A deep reinforcement learning framework for optimal trade execution. In *Machine learning and knowledge discovery in databases. applied data science and demo track: European conference, ECML pKDD 2020, ghent, Belgium, September 14–18, 2020, proceedings, part v* (pp. 223–240).

Lin, S., & Beling, P. A. (2021b). An end-to-end optimal trade execution framework based on proximal policy optimization. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence* (pp. 4548–4554).

Liu, X. (2023). Dynamic coupon targeting using batch deep reinforcement learning: An application to livestream shopping. *Marketing Science*, *42*(4), 637–658.

Liu, Y., Liu, Q., Zhao, H., Pan, Z., & Liu, C. (2020). Adaptive quantitative trading: An imitative deep reinforcement learning approach. *vol. 34*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 2128–2135).

Liu, X.-Y., Xia, Z., Rui, J., Gao, J., Yang, H., Zhu, M., et al. (2022). FinRL-Meta: Market environments and benchmarks for data-driven financial reinforcement learning. *Advances in Neural Information Processing Systems*, *35*, 1835–1849.

Liu, X.-Y., Yang, H., Chen, Q., Zhang, R., Yang, L., Xiao, B., et al. (2020). FinRL: A deep reinforcement learning library for automated stock trading in quantitative finance. arXiv preprint arXiv:2011.09607.

Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in Neural Information Processing Systems*, 6379–6390.

Lucarelli, G., & Borrotti, M. (2020). A deep Q-learning portfolio management framework for the cryptocurrency market. *Neural Computing and Applications*, *32*, 17229–17244.

Macrì, A., & Lillo, F. (2024). Reinforcement learning for optimal execution when liquidity is time-varying. arXiv preprint arXiv:2402.12049.

Madhavan, A. (2002). VWAP strategies. *Trading*, *1*, 32–39.

Malekzadeh, P., Poulos, Z., Chen, J., Wang, Z., & Plataniotis, K. N. (2024). EX-DRL: Hedging against heavy losses with extreme distributional reinforcement learning. In *Proceedings of the 5th ACM international conference on AI in finance* (pp. 370–378).

Mani, M., Phelps, S., & Parsons, S. (2019). Applications of reinforcement learning in automated market-making. In *Proceedings of the GAIW: games, agents and incentives workshops* (pp. 13–14).

Markowitz, H. M., & Markowitz, H. M. (1967). *Portfolio selection: efficient diversification of investments*.

Markowitz, H. M., & Todd, G. P. (2000). *Mean-variance analysis in portfolio choice and capital markets*.

Marzban, S., Delage, E., Li, J. Y., Desgagne-Bouchard, J., & Dussault, C. (2021). WaveCorr: Correlation-savvy deep reinforcement learning for portfolio management. arXiv preprint arXiv:2109.07005.

Millea, A. (2021). Deep reinforcement learning for trading—A critical survey. *Data*, *6*(11), 119.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., et al. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928–1937).

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Nawathe, S., Panguluri, R., Zhang, J., & Venkatesh, S. (2024). Multimodal deep reinforcement learning for portfolio optimization. arXiv preprint arXiv:2412.17293.

Neuneier, R. (1997). Enhancing Q-learning for optimal asset allocation. *Advances in Neural Information Processing Systems*, 936–942.

Nevmyvaka, Y., Feng, Y., & Kearns, M. (2006). Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on machine learning* (pp. 673–680).

Ning, B., Lin, F. H. T., & Jaimungal, S. (2021). Double deep q-learning for optimal execution. *Applied Mathematical Finance*, *28*(4), 361–380.

Niu, H., Li, S., & Li, J. (2022). MetaTrader: An reinforcement learning approach integrating diverse policies for portfolio optimization. In *Proceedings of the 31st ACM international conference on information & knowledge management* (pp. 1573–1583).

Noh, E., & Weston, K. (2022). Price impact equilibrium with transaction costs and TWAP trading. *Mathematics and Financial Economics*, *16*(1), 187–204.

Pan, F., Zhang, T., Luo, L., He, J., & Liu, S. (2022). Learn continuously, act discretely: Hybrid action-space reinforcement learning for optimal execution. arXiv preprint arXiv:2207.11152.

Park, H., Sim, M. K., & Choi, D. G. (2020). An intelligent financial portfolio trading strategy using deep Q-learning. *Expert Systems with Applications*, *158*, Article 113573.

Patel, Y. (2018). Optimizing market making using multi-agent reinforcement learning. arXiv preprint arXiv:1812.10252.

Peng, B., Li, X., Gao, J., Liu, J., Wong, K.-F., & Su, S.-Y. (2018). Deep dyna-q: Integrating planning for task-completion dialogue policy learning. arXiv preprint arXiv:1801.06176.

Pineda, L., Amos, B., Zhang, A., Lambert, N. O., & Calandra, R. (2021). Mbrl-lib: A modular library for model-based reinforcement learning. arXiv preprint arXiv:2104.10159.

Prudencio, R. F., Maximo, M. R. O. A., & Colombini, E. L. (2023). A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 1–10.

Sadighian, J. (2019). Deep reinforcement learning in cryptocurrency market making. arXiv preprint arXiv:1911.08647.

Sadighian, J. (2020). Extending deep reinforcement learning frameworks in cryptocurrency market making. arXiv preprint arXiv:2004.06985.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Sekkat, H., Moutik, O., Ourabah, L., ElKari, B., Chaibi, Y., & Tchakoucht, T. A. (2024). Review of reinforcement learning for robotic grasping: Analysis and recommendations. *Statistics, Optimization & Information Computing*, *12*(2), 571–601.

Sharma, A., Chen, F., Noh, J., DeJesus, J., & Schlener, M. (2024). Hedging beyond the mean: A distributional reinforcement learning perspective for hedging portfolios with structured products. arXiv preprint arXiv:2407.10903.

Shen, Y., Huang, R., Yan, C., & Obermayer, K. (2014). Risk-averse reinforcement learning for algorithmic trading. In *2014 IEEE conference on computational intelligence for financial engineering & economics (cIFEr)* (pp. 391–398).

Shi, S., Li, J., Li, G., Pan, P., Chen, Q., & Sun, Q. (2022). GPM: A graph convolutional network based reinforcement learning framework for portfolio management. *Neurocomputing*, *498*, 14–27.

Soleymani, F., & Paquet, E. (2021). Deep graph convolutional reinforcement learning for financial portfolio management–DeepPocket. *Expert Systems with Applications*, *182*, Article 115127.

Spooner, T., & Savani, R. (2020). Robust market making via adversarial reinforcement learning. arXiv preprint arXiv:2003.01820.

Sun, R., Jiang, Z., & Su, J. (2021). A deep residual shrinkage neural network-based deep reinforcement learning strategy in financial portfolio management. In *2021 IEEE 6th international conference on big data analytics* (pp. 76–86).

Sun, Q., Wei, X., & Yang, X. (2024). GraphSAGE with deep reinforcement learning for financial portfolio optimization. *Expert Systems with Applications*, *238*, Article 122027.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. *vol. 30*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 2094–2100).

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., et al. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, *575*(7782), 350–354.

Wang, Q., Hu, J., Wu, Y., & Zhao, Y. (2023). Output synchronization of wide-area heterogeneous multi-agent systems over intermittent clustered networks. *Information Sciences*, *619*, 263–275.

Wang, Z., Huang, B., Tu, S., Zhang, K., & Xu, L. (2021). Deeptrader: A deep reinforcement learning approach for risk-return balanced portfolio management with market conditions embedding. *vol. 35*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 643–650).

Wang, T. R., Pradeep, J., & Chen, J. Z. (2022). Objective driven portfolio construction using reinforcement learning. In *Proceedings of the third ACM international conference on AI in finance* (pp. 264–272).

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning* (pp. 1995–2003).

Wang, R., Wei, H., An, B., Feng, Z., & Yao, J. (2021). Commission fee is not enough: A hierarchical reinforced framework for portfolio management. *vol. 35*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 626–633).

Wang, J., Zhang, Y., Tang, K., Wu, J., & Xiong, Z. (2019). Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1900–1908).

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279–292.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256.

Xu, K., Zhang, Y., Ye, D., Zhao, P., & Tan, M. (2021). Relation-aware transformer for portfolio policy learning. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence* (pp. 4647–4653).

Yang, S. (2023). Deep reinforcement learning for portfolio management. *Knowledge-Based Systems*, *278*, Article 110905.

Ye, H., Li, G. Y., & Juang, B.-H. F. (2019). Deep reinforcement learning based resource allocation for V2v communications. *IEEE Transactions on Vehicular Technology*, *68*(4), 3163–3173.

Ye, Y., Pei, H., Wang, B., Chen, P.-Y., Zhu, Y., Xiao, J., et al. (2020). Reinforcement-learning based portfolio management with augmented asset movement prediction states. *vol. 34*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 1112–1119).

Zha, L., Dai, L., Xu, T., & Wu, D. (2022). A hierarchical reinforcement learning framework for stock selection and portfolio. In *2022 international joint conference on neural networks* (pp. 1–7).

Zhang, Y., Zhao, P., Wu, Q., Li, B., Huang, J., & Tan, M. (2020). Cost-sensitive portfolio selection via deep reinforcement learning. *IEEE Transactions on Knowledge and Data Engineering*, *34*(1), 236–248.

Zhao, M., & Linetsky, V. (2021). High frequency automated market making algorithms with adverse selection risk control via reinforcement learning. In *Proceedings of the second ACM international conference on AI in finance* (pp. 1–9).

Zhong, Y., Bergstrom, Y. M., & Ward, A. (2021). Data-driven market-making via model-free learning. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence* (pp. 4461–4468).

Zhou, G., Tian, W., Buyya, R., Xue, R., & Song, L. (2024). Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions. *Artificial Intelligence Review*, *57*(5), 124.