Article | Received 31 October 2024; Accepted 13 February 2025; Published 24 February 2025
https://doi.org/10.55092/blockchain20250005

RL-BES: optimizing strategies using reinforcement learning for blockchain economic security

Erdeng Wen[1], Zhuotao Deng[2], Yifan Mo[3], Yuren Zhou[3] and Xiaohong Shi[4,*]

[1] School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2] Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China
[3] School of Software Engineering, Sun Yat-sen University, Zhuhai, China
[4] School of Information Engineering, Guangzhou Panyu Polytechnic, Guangzhou, China
* Correspondence author; E-mail: shixh@gzpyp.edu.cn

Highlights:

- The process of optimizing transaction sequences and template parameters is modeled as a Markov Decision Process in RL-BES, providing a systematic approach to address decision-making challenges in complex MEV detection and offering new insights for future research.
- RL-BES innovatively integrates reinforcement learning with the MCTS algorithm, combining two deep reinforcement learning networks and evaluation algorithms. This results in a more advanced theoretical framework and enhanced MEV extraction performance compared to traditional methods.
- A custom model evaluation tool is developed within RL-BES to intelligently optimize transaction ordering in a simulated blockchain environment and assess performance. This tool facilitates efficient MEV detection and extraction, allowing for customizable adjustments of different network structures and parameters to continuously analyze the best algorithmic solutions for MEV extraction.

Abstract: The rapid growth of decentralized finance (DeFi) has provided numerous benefits, but it has also presented significant economic security challenges. One of the most critical issues is Maximum Extractable Value (MEV). MEV refers to the opportunities for miners or validators to earn additional profits by altering the order of transactions. However, current MEV detection methods have notable limitations. These include poor adaptability of algorithms, the vastness of the search space, and the inefficiency of methods that rely on traditional heuristic approaches. To overcome these challenges, we introduces a reinforcement learning-based MEV optimization system for blockchain—RL-BES (Reinforcement Learning for Blockchain Economic Security). This system employs two deep reinforcement learning networks to optimize transaction ordering and template parameters, integrated with Monte Carlo Tree Search (MCTS) for effective path exploration. Furthermore, we presents a custom model evaluation tool designed to adjust various networks and parameters, facilitating the analysis of the best algorithmic solutions for on-chain MEV extraction. Experimental results indicate that the RL-BES system excels in multiple DeFi applications. It demonstrates faster convergence and consistently surpasses the performance of Flashbot and other similar detection tools.

Keywords: blockchain; reinforcement learning; Maximum Extractable Value (MEV)

1. Introduction

With the rapid development of blockchain technology [1–4], decentralized finance (DeFi) has become a crucial component of the modern financial ecosystem [5]. DeFi refers to a range of financial services built on blockchain technology and smart contracts, including lending, trading, liquidity provision, asset management, and insurance. Unlike traditional financial systems, DeFi eliminates intermediaries, allowing users to transact directly. This reduces transaction costs and enhances transparency and efficiency [6]. Users can freely engage in financial transactions globally, gaining higher returns and liquidity. However, despite the conveniences DeFi offers, it also presents a series of economic security challenges, particularly the issue of Maximum Extractable Value (MEV).

MEV refers to the opportunities for miners or validators to earn extra profits by rearranging, inserting, or deleting transactions within blockchain transactions [7]. This phenomenon not only affects the fairness of transactions but also undermines the economic security of the network. The existence of MEV can lead to unknowing economic losses for some users, especially in high-frequency trading and volatile market environments [8]. Furthermore, MEV strategies vary widely, including arbitrage, liquidation, and sandwich attacks. Arbitrage strategies capitalize on price differences across markets for profit [9], while liquidation strategies involve profiting from the liquidation of distressed assets in the event of loan defaults [10]. Sandwich attacks occur when an attacker inserts their transactions before and after a user's transaction to profit from the price movements [11]. These strategies not only harm the fairness of transactions but can also lead to decreased market liquidity and overall trust.

Detecting, analyzing, and optimizing MEV is challenging due to the complex search space of blockchain transactions and the dynamic nature of trading environments [12]. Although some progress has been made in the research of MEV detection and optimization, existing methods often face issues such as poor adaptability, inefficiency, and dependence on fixed rules, limiting their effectiveness in detecting and mitigating MEV [13].

To address these challenges, we propose a reinforcement learning-based blockchain MEV optimization system—RL-BES (Reinforcement Learning for Blockchain Economic Security) that combines deep learning, reinforcement learning, Monte Carlo Tree Search (MCTS), and Markov Decision Process (MDP).

Deep learning enables the system to process complex transaction data, while reinforcement learning allows it to optimize strategies through interaction with the environment. MCTS provides an efficient way to explore the vast search space of transaction sequences, and MDP offers a systematic framework for decision-making in dynamic environments. Together, these methods enable RL-BES to effectively detect and optimize MEV extraction, enhancing blockchain economic security.

The innovations of RL-BES are highlighted in several key aspects:

- **MDP modeling:** we model the process of optimizing transaction sequences and template parameters as a Markov Decision Process (MDP), providing a systematic approach to address decision-making challenges in complex MEV detection, thereby offering new insights for future research.
- **Multiple algorithms combining:** we innovatively introduce reinforcement learning and the MCTS algorithm, independently designing two deep reinforcement learning networks and evaluation algorithms, achieving a more advanced theoretical framework and improved MEV extraction performance compared to traditional methods.
- **Tool implementation:** we develop a custom model evaluation tool that intelligently optimizes transaction ordering in a simulated blockchain environment and assesses performance. This tool enables efficient MEV detection and extraction while allowing for customizable adjustments of different network structures and parameters to continually analyze the best algorithmic solutions for on-chain MEV extraction.

- **Organizations**: the remaining organization of this paper is listed as follows: Section 2 provides a summary and analysis of existing works related to MEV detection. Section 3 outlines relevant background knowledge. Section 4 details the algorithms and internal implementation of the RL-BES system. Section 5 discusses experimental preparations, methodologies, and results. Section 6 addresses the significance, limitations, and potential improvement directions of the RL-BES system. Finally, Section 7 summarizes the contributions and findings of this research.

2. Related work

Currently, there are four main approaches to optimizing the extraction of MEV: hard coding specific contracts, semantic modeling and formal verification, traditional optimization algorithms along with heuristic methods and advanced machine learning methods.

2.1. Hard coding methods

Hard coding specific contracts is one of the early attempts at MEV optimization. This method involves predefining contracts and trading rules in specific transaction scenarios to directly adjust transaction ordering for MEV extraction. Torres *et al*. [14] and Qin *et al*. [15] were among the first to use this approach to measure historical MEV and optimize extraction on corresponding contracts on Ethereum. The advantages of this method include its simplicity and rapid deployment. However, hard coding relies heavily on fixed contract structures, lacking the flexibility to adapt to different contracts. When new contracts are introduced, the existing rules may become invalid, requiring the system to be re-coded and adjusted. Moreover, hard coding cannot address unforeseen new MEV extraction scenarios, resulting in poor generalization and limited performance in the complex and rapidly evolving DeFi ecosystem.

2.2. Semantic modeling methods

The method utilizing semantic modeling and formal verification tools conducts rigorous semantic analysis of trading behaviors to ensure logical consistency and security in the MEV optimization process. This approach employs formal verification tools to define and assess optimized transaction paths. McLaughlin *et al*. [16] have modeled on-chain arbitrage behaviors and conducted MEV detection by programming the general semantics of smart contracts and simulating MEV detection. Similarly, Babel *et al*. have used comparable methods to measure and optimize on-chain MEV [17], while Li *et al*. have modeled various typical representations of MEV to detect corresponding MEV situations [18]. The advantage of semantic modeling lies in its ability to accurately capture the relationships between trading behaviors. However, these methods often require complex model construction and reasoning, leading to high computational costs. Additionally, the scalability of formal verification tools is limited; when faced with complex trading combinations or new trading patterns, pre-analysis, re-modeling, and verification are often necessary.

2.3. Traditional and heuristic methods

Traditional optimization algorithms and heuristic methods are also common techniques for addressing MEV issues. These methods typically employ traditional optimization techniques, such as Particle Swarm Optimization (PSO), Genetic Algorithms (GA), or Greedy Algorithms, in an attempt to find suboptimal solutions within a limited search space.

Zhou *et al*. utilized the Sequential Least Squares Programming (SLSQP) algorithm to constrain MEV optimization [19]. Their work focused on analyzing the potential vulnerabilities in the DeFi ecosystem through the use of flash loans and attempting to optimize MEV

extraction within a certain framework. However, a significant shortcoming of their approach is that it was highly dependent on the specific assumptions and constraints set by the SLSQP algorithm. In a highly dynamic DeFi environment where market conditions and transaction patterns can change rapidly, this rigidity limited the ability of their method to adapt and capture the full range of MEV opportunities. Moreover, the SLSQP algorithm, like many traditional optimization algorithms, may not be efficient enough when dealing with the large and complex search spaces typical in DeFi transactions.

Babel *et al*. applied a similar genetic algorithm approach to optimize the MEV extraction process [20]. The genetic algorithm they used had the advantage of being relatively straightforward to implement and could potentially find good solutions in smaller-scale problems. However, it also had notable limitations. The genetic algorithm relied on a fixed set of genetic operators and fitness functions, which might not be suitable for all types of MEV scenarios. As the DeFi landscape evolves and new types of transactions and strategies emerge, the lack of adaptability of these fixed rules could lead to suboptimal results.

Therefore, the advantages of heuristic algorithms include their relative ease of implementation and effectiveness in finding near-optimal solutions for smaller-scale problems. However, they exhibit several limitations. First, heuristic methods often rely on fixed rules, which lack the adaptability to handle dynamic and complex DeFi environments. Second, heuristic methods struggle with efficiency in large search spaces, as they typically explore suboptimal solutions within limited computational budgets. Finally, traditional methods lack flexibility in adjusting the ordering of transactions and the parameters, resulting in underestimated results for the extraction of MEV.

2.4. Advanced machine learning methods

Recent studies have demonstrated the potential of advanced machine learning techniques in addressing blockchain optimization challenges. Gai *et al*. [21] proposed the use of large language models (LLMs) to enhance smart contract analysis and transaction processing efficiency, providing a novel approach to handling complex blockchain interactions. Their work highlights the importance of leveraging machine learning for scalable and adaptive blockchain systems. Similarly, Tian *et al*. [22] introduced an interdisciplinary approach combining reinforcement learning and mechanism design to optimize incentive structures in Proof-of-Stake (PoS) Ethereum. In addition, Islam *et al*. [23] emphasizes the significance of consensus algorithms in blockchain networks and the challenge of malicious nodes due to decentralization. It introduces MRL - PoS, a multiagent reinforcement learning based PoS algorithm that adapts to user behavior, manages malicious nodes via rewards and penalties, and learns to counter new threats. Zhao *et al*. [24] explores the application of deep reinforcement learning in forecasting and risk management of the cryptocurrency market trend. It uses the LSTM model for data analysis, combines multiple factors, and the experimental results prove its practical value, with future research suggesting improvements to the LSTM model for better performance. Their research underscores the effectiveness of reinforcement learning in designing economically secure blockchain protocols.

2.5. Improvements of RL-BES

To address these challenges, RL-BES introduces several key improvements. First, RL-BES overcomes the limitations of traditional hard coding methods by incorporating a reinforcement learning framework that can adaptively optimize transaction ordering within blocks containing different contracts. Second, RL-BES mitigates the high computational complexity associated with semantic modeling and formal verification tools by modeling the semantics of important behaviors from common DeFi applications, significantly reducing computational and exploration costs. Additionally, by combining MCTS with deep reinforcement learning algorithms,

RL-BES can efficiently search for optimal transaction sequences within a vast search space, enhancing the flexibility and efficiency of optimization tasks. Table 1 compares RL-BES with some existing tools.

Table 1. Comparison of tool functionality.

Tool names	MEV detection	High generalization	High efficiency	Sorting function	Custom tuning
MEV-explore [25]	✓	×	×	×	×
MEV-inspect [26]	✓	✓	×	×	×
Flash(bot) Pan [27]	✓	×	×	×	×
Lanturn [20]	✓	✓	×	✓	×
RL-BES	✓	✓	✓	✓	✓

3. Preliminaries

3.1. DeFi

Decentralized Finance (DeFi) is one of the key applications of blockchain technology, aimed at replacing traditional financial services with smart contracts and decentralized protocols. DeFi offers services in a wide range of areas, including lending, trading, payments, insurance, derivatives, and stablecoins, with the goal of creating a financial system that is transparent, efficient, and free of intermediaries [28]. Transactions in DeFi are recorded on the blockchain, with all operations automatically executed by smart contracts. This makes DeFi highly transparent and resistant to censorship. Typical DeFi use cases include:

Decentralized Exchanges (DEXs): platforms like Uniswap and SushiSwap allow users to trade tokens directly via smart contracts without relying on centralized exchanges. These exchanges often use Automated Market Maker (AMM) mechanisms, replacing traditional order books with liquidity pools [29].

Lending protocols: platforms like Aave and Compound enable users to borrow assets by collateralizing crypto assets, while also earning interest. This eliminates intermediaries like banks, making borrowing more transparent and accessible [30].

Stablecoins: examples include DAI and USDC, which maintain price stability against fiat currencies through algorithms or collateralization. These are widely used in transactions and payments to mitigate the price volatility of cryptocurrencies [31].

3.2. MEV

Maximum Extractable Value (MEV) refers to the additional profit that block producers (miners or validators) can extract by manipulating transaction ordering, inserting, or deleting transactions during block creation. Before a transaction is included in a block on Ethereum, it resides in the mempool, a publicly accessible pool of pending transactions visible to everyone [32]. Miners can monitor the mempool and reorder transactions to maximize their profit. Common causes of MEV formation include:

Arbitrage: miners exploit price differences between decentralized exchanges by executing buy or sell orders at advantageous prices.

Liquidation: in decentralized lending protocols, miners can prioritize liquidation transactions when the value of collateral falls below a certain threshold, earning liquidation rewards.

Sandwich attacks: attackers insert their transactions before and after a user's transaction to profit from price slippage.

The mathematical definition of MEV is as follows. MEV can be expressed as the profit difference between the original transaction order and the optimized order:

$$\text{MEV} = \sum_{i=1}^{n} \left(P_i(\text{new order}) - P_i(\text{original order}) \right) \tag{1}$$

Where P_i represents the profit of the i-th transaction under different ordering scenarios. During the MEV optimization process, miners attempt to rearrange the transaction sequence to maximize this value, thereby extracting the highest possible value.

Existing solutions are often limited to specific scenarios, lacking generalizability and scalability. Therefore, optimizing transaction ordering and improving economic security have become important research topics in blockchain.

3.3. Reinforcement learning

Reinforcement Learning (RL) is a machine learning method that autonomously learns and optimizes strategies, widely applied to decision-making problems in dynamic environments [33]. Unlike traditional supervised learning, RL relies on interactions with the environment and uses reward signals to guide strategy optimization. The core idea is to balance exploration and exploitation, gradually identifying an optimal strategy that maximizes long-term rewards [34].

Reinforcement learning problems are typically modeled as a Markov Decision Process (MDP) [35], which includes the following elements:

State (s): represents the information available to the agent about the environment at a specific moment.

Action (a): denotes the decision made by the agent when in a particular state.

Transition ($P(s'|s,a)$): describes the probability of transitioning from state s to a new state s' after executing action a.

Reward (r): refers to the immediate feedback received by the agent after taking an action in the environment.

Discount factor (γ): balances immediate rewards and future rewards, where the value of γ is between $[0,1]$.

The goal of reinforcement learning is to maximize the cumulative reward R_t by optimizing the policy $\pi(a|s)$. The expected cumulative future reward is defined as:

$$R_t = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \right] \tag{2}$$

By optimizing the policy, the agent can select the most optimal actions in different states to achieve the maximum expected long-term reward.

3.4. MCTS

Monte Carlo Tree Search (MCTS) is a search algorithm that identifies optimal solutions by simulating future states and progressively expanding a decision tree [36]. It is particularly suitable for path optimization in complex search spaces with a large number of state and action combinations. The key idea is to recursively simulate possible future scenarios, gradually constructing a decision tree that encompasses all possible paths, thus providing optimal strategy guidance for current decisions. MCTS consists of four main steps:

- **Selection**: starting from the root node, a node along an existing path in the tree is selected for expansion. The selection process typically uses a heuristic strategy like the Upper Confidence Bound (UCB1) algorithm [37], which balances exploration and exploitation:

$$\mathrm{UCB1}(s,a) = \overline{Q}(s,a) + c\sqrt{\frac{\ln N(s)}{N(s,a)}} \tag{3}$$

Where $\overline{Q}(s,a)$ represents the average reward received when selecting action a in state s. $N(s)$ is the number of times state s has been visited. $N(s,a)$ is the number of times action a has been selected in state s. c is a constant that balances exploration and exploitation.

- **Expansion**: once a node that is not fully expanded is selected, new child nodes are created to represent the next possible actions.
- **Simulation**: from the newly expanded node, actions are selected randomly, and the decision process is simulated until a terminal state is reached. This step explores future scenarios through random sampling, independent of the existing decision tree.
- **Backpropagation**: the reward information from the simulation is propagated back up the tree, from the newly expanded node to the root, updating the statistics of all nodes along the way.

By repeatedly executing these steps, MCTS gradually optimizes the structure of the decision tree, making it an effective solution for high-dimensional and uncertain search problems. In transaction ordering and MEV extraction, MCTS can dynamically adjust transaction sequences, ensuring a balance between exploration efficiency and optimality during the search process.

4. The RL-BSE system

4.1. Model overview

The RL-BES system we designed combines reinforcement learning and MCTS to address the challenges of transaction reordering and parameter adjustment in DeFi applications. Its goal is to maximize the MEV on the blockchain, thereby enhancing economic security. The system efficiently explores optimal transaction orders within a vast search space using MCTS, and continuously optimizes the transaction sequence and key parameters through reinforcement learning. By modeling transaction reordering and parameter optimization as a MDP, the system can dynamically adjust strategies in complex environments. Additionally, it incorporates attention mechanisms and convolutional neural networks (CNN) to enhance adaptability in sequence and parameter tuning.

The choice of deep learning, reinforcement learning, MCTS, and MDP in RL-BES is motivated by their complementary strengths in addressing the challenges of MEV extraction.

- **Deep learning**: deep learning excels at processing large amounts of transaction data and extracting complex features, which is essential for understanding the relationships between transactions in DeFi environments.
- **Reinforcement learning**: reinforcement learning enables the system to learn optimal strategies through experiments and error, making it well-suited for dynamic and complex tasks like transaction reordering.
- **Monte Carlo Tree Search (MCTS)**: MCTS provides an efficient way to explore the vast search space of possible transaction sequences, allowing RL-BES to identify high-reward paths without exhaustive enumeration.
- **Markov Decision Process (MDP)**: MDP offers a systematic framework for modeling the decision-making process in MEV optimization, ensuring that RL-BES can dynamically adjust strategies based on the current state of the environment.

By integrating these methods, RL-BES achieves a robust balance between exploration and exploitation, enabling it to effectively optimize MEV extraction in complex blockchain environments.

Figure 1 illustrates the execution steps of the RL-BES system.

- **Data input and MDP modeling**: the system begins by inputting a transaction dataset, which has been semantically modeled and initialized with key DeFi application parameters. This provides a simplified semantic representation of the transaction sequence. In addition, the system will model the corresponding tasks as MDP to obtain the basic policy network framework. This process will be explained in detail in Section 4.2.1 and Section 4.2.2.
- **Variable initialization**: next, the input transaction sequence is processed by the reordering network, which predicts a set of initial action selection probabilities. The parameter optimization network then generates a set of initial template parameters.
- **MCTS exploration**: the system uses MCTS to explore the search space of transaction reordering. Each path in the tree represents a potential transaction sequence. MCTS dynamically evaluates these paths to identify promising reordering schemes, which are passed to the local simulation environment for reward calculation. This process will be explained in detail in Section 4.3.1.
- **Transaction simulation and reward extraction**: in the local simulation environment, the system performs a simulated block mining based on the currently explored transaction sequence. The MEV of the block is calculated as the reward for the current output, which is then fed into the reinforcement learning network module for updates.
- **Reinforcement learning network update**: the calculated MEV reward, along with the current transaction sequence, is fed into the attention-based reordering network and the CNN-based parameter optimization network. Through processes such as network optimization and backpropagation, the weights and parameters of the networks are updated. This process will be explained in detail in Section 4.3.2 and Section 4.3.3.
- **Feedback into MCTS exploration**: after the network updates, the system predicts a new set of action selection probabilities and template parameters for the input transaction sequence. This process is repeated, allowing the reinforcement learning network to continuously update while recording the optimal MEV extraction results and transaction order, until the predefined number of iterations is reached.

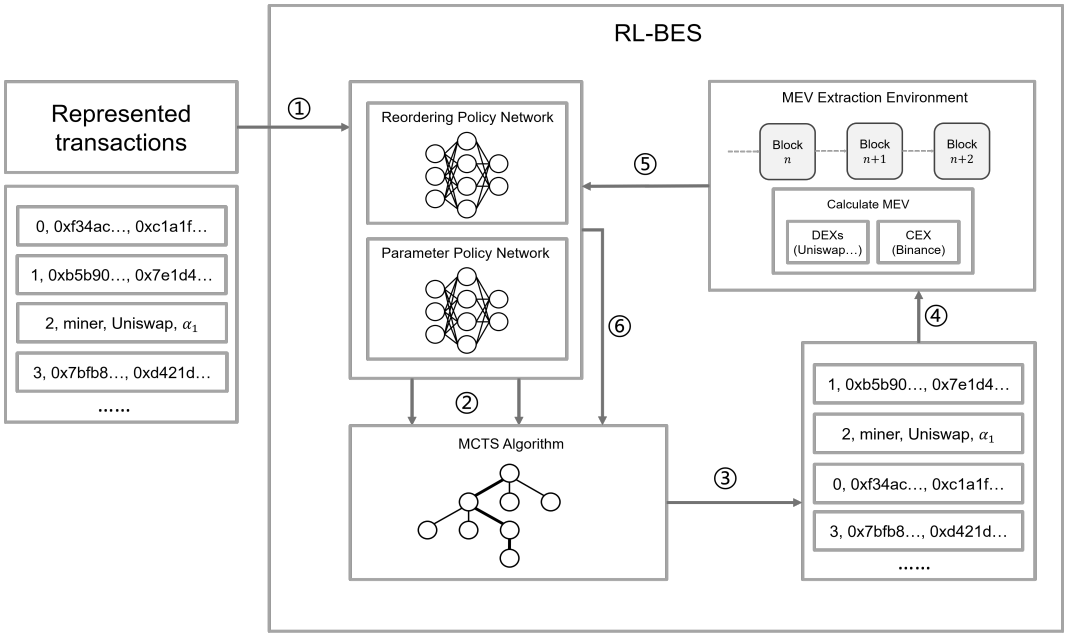

Figure 1. The RL-BES system.

4.2. Problem modeling

In this section, we discuss how to model the problem of transaction reordering and parameter optimization in DeFi as a mathematical problem. This is essential for applying reinforcement learning techniques later. We will cover the modeling of key parameters and the construction of the Markov Decision Process (MDP), with a focus on providing a reasonable environment for reinforcement learning.

4.2.1. Semantic modeling of key parameters

In DeFi applications, various key parameters directly affect transaction outcomes and profits. Examples include the collateral ratio in lending protocols, slippage thresholds in decentralized exchanges, and liquidation thresholds. We perform semantic modeling of these critical parameters so that they can be flexibly adjusted during the optimization process. Here is an example to illustrate the types of these parameters:

In the UniswapV3 protocol, the slippage tolerance parameter affects the execution price of a trade[38]. If the slippage is too high, users may incur significant losses. Conversely, if the slippage is too low, the trade may fail to execute. The formula for the slippage threshold is given by:

$$Slippage = \frac{P_e - P_x}{P_e} \times 100\% \tag{4}$$

Where P_e is the price anticipated by the user before initiating the trade. P_x is the actual price at which the trade is completed.

Similarly, we model relevant key parameters from protocols like UniswapV3 as $\alpha_x(x = 1,2...)$ and convert the raw transaction dataset into a semantically represented format. By doing so, we simplify the modeling of complex contract interactions into a transaction sequence, represented by function names and key parameters. This allows our system to avoid needing in-depth knowledge of contract implementation and meaning. Consequently, the complexity of the modeling process is greatly reduced, enhancing the system's generality and adaptability for detecting various contracts.

To better illustrate this process, Figure 2 presents an example of semantic modeling concerning the behavior of providing liquidity in UniswapV3. Firstly, we initialize key parameters, such as liquidity and range values as different variables $\alpha_x(x = 1,2,3)$. Then, we simplify the entire transaction data into an atomic transaction representation. Finally, we convert this atomic transaction representation into a data form that can be recognized by our model through semantic modeling.

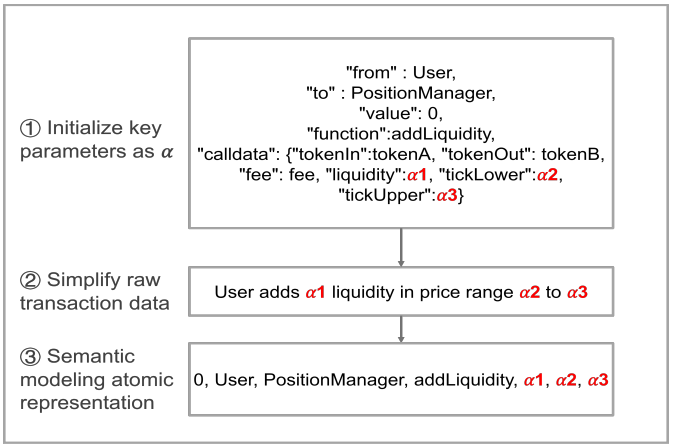

Figure 2. Example of semantic modeling.

4.2.2. Modeling reordering and optimization problem as MDP

We unify the transaction reordering and parameter optimization problem into a MDP framework, allowing reinforcement learning networks to address this issue. Specifically, the four key components of an MDP (state, action, reward, and transition) are used to clearly describe how transaction sequence arrangement and parameter adjustments affect MEV extraction.

- **State space** (S): at time step t, each state s_t represents the current transaction sequence and the associated DeFi parameters. We denote $s_t = \{T, \theta\}$, where T is the transaction sequence, and θ refers to the fixed DeFi parameters (e.g., slippage or collateral ratio).
- **Action space** (A): at time t, action a_t represents the reordering and adjustment of DeFi parameters. The action space is defined as $a_t = \{\text{Reorder}(T), \text{Adjust}(\theta)\}$, where T is the transaction sequence, and θ denotes the DeFi parameters.
- **Transition function** (P): the transition function $P(s_t, a_t)$ defines how the system moves from one state s_t to the next state s_{t+1} after taking action a_t. The transition is determined by the state transition function which takes into account the effect of reordering and parameter adjustment:

$$s_{t+1} = P(s_t, a_t) \tag{5}$$

  where P is the state transition function that determines the new transaction sequence and parameter settings.
- **Reward function** (R): the reward function $R(s_t, a_t)$ quantifies the effect of action a_t on the MEV optimization problem. In the context of MEV extraction, the reward function is defined as:

$$R(s_t, a_t) = \text{MEV}(T_{t+1}, \theta_{t+1}) - \text{Cost}(T_{t+1}, \theta_{t+1}) \tag{6}$$

  where $\text{MEV}(T_{t+1}, \theta_{t+1})$ represents the MEV extracted in the next time step, and $\text{Cost}(T_{t+1}, \theta_{t+1})$ refers to the associated transaction costs.
- **Discount factor** (γ): the discount factor $\gamma \in [0, 1]$ balances immediate and future rewards, ensuring that the model takes future rewards into consideration.

The overall goal of MDP is to find an optimal strategy π^* that maximizes the cumulative discount reward, namely:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right] \tag{7}$$

Figure 3 illustrates the structure of modeling transaction reordering and parameter optimization as an MDP. At time step t, the system is in state $s_t = \{T, \theta\}$, which includes the current transaction sequence T and the related DeFi parameter settings θ. The RL-BES strategy network chooses an action $a_t = \{\text{Reorder}(T), \text{Adjust}(\theta)\}$, either reordering the transactions or adjusting the parameters. After the action a_t is executed, the system transitions to a new state s_{t+1}. The reward function $R(s_t, a_t)$ evaluates the effect of this action. Through multiple iterations, the system learns the optimal policy to maximize MEV extraction.

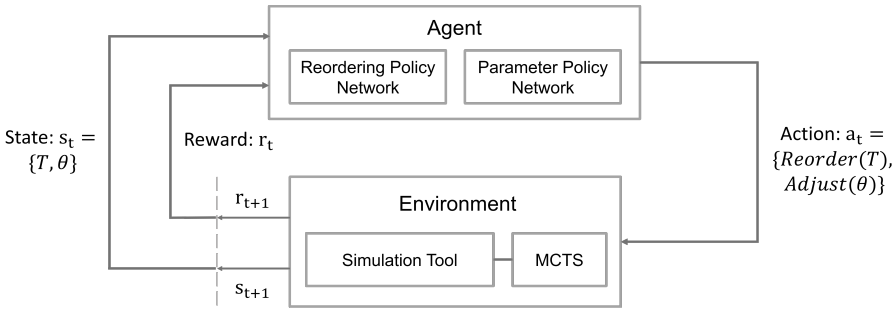

Figure 3. MDP framework.

4.3. Model and algorithm design

In this section, we will provide a detailed explanation of the core algorithm design within the RL-BES system. This includes the exploration method for transaction sorting (based on MCTS), the reordering policy update mechanism (reinforcement learning network based on attention mechanisms), and the process of optimizing transaction parameters (reinforcement learning network based on convolutional neural networks).

4.3.1. Exploration of sorting schemes: MCTS algorithm

To effectively search for the optimal sorting in the vast space of transaction sequences, we employ MCTS as the search algorithm. MCTS combines a policy network for initializing the leaf nodes and evaluating rewards, and uses the UCB1 algorithm for action selection. Algorithm 1 describes the code execution process of the algorithm. Through the four steps of selection, expansion, simulation, and backpropagation, it recursively constructs a decision tree in the search space to find the optimal transaction order. The MCTS algorithm mainly consists of the following phases:

- **Tree initialization and node expansion**: during each iteration of the search process, the algorithm expands the decision tree to include new nodes that correspond to possible transaction sequences. The following information is recorded for each node:
  - N: Number of times the node has been visited.
  - W: Sum of rewards from the root to the current node.
  - Q: Average reward of the node (calculated as $\left(\frac{W}{N}\right)$).
  - P: Action prior from the reordering policy network.
- **UCB1 action selection**: at each non-leaf node, the UCB1 formula is used to select the next action. The UCB1 algorithm balances exploration and exploitation, ensuring that actions with higher uncertainty are explored, while also exploiting actions that are known to yield better results. The UCB1 formula is as follows:

$$UCB1(a) = Q(a) + w_e \times P(a) \times \sqrt{\frac{\sum N}{1+N(a)}} \tag{8}$$

  where $Q(a)$ is the average reward for action a. $P(a)$ is the prior probability provided by the reordering policy network. $N(a)$ is the number of visits to action a. $\sum N$ is the total number of visits to all actions. w_e is the weight for adjust exploration.
- **Model and state transition**: after selecting an action a_t , the system transitions to a new state s_{t+1}, representing a new transaction sequence. The system then evaluates the immediate reward r_t, which is calculated as follows:

$$r_t = A \times (1 - \alpha \times e) \times (1 + p) \times t_d \tag{9}$$

where α is the entropy regularization parameter, used to balance exploration and exploitation. A is the confidence score of the selected action. p reflects the progress of the transaction sequence. t_d is a temporal discount factor, ensuring that actions taken earlier in the sequence are rewarded more heavily.

- **Simulation and backpropagation**:

  in this phase, the system continues to expand the tree by simulating further actions, and then backpropagates the cumulative rewards to update the values of N, W, and Q . This allows the tree to update its knowledge of the optimal transaction sequence and improve its decision-making in future iterations.

- **Termination**:

  the search process terminates when a predefined number of iterations are completed, or when the algorithm converges to a sufficiently optimal solution. The sequence with the highest cumulative reward is selected as the final transaction order.

Algorithm 1 MCTS-Based transaction ordering

Require: Transaction pool, reordering policy network, parameter policy network, initial state
Ensure: Best action sequence, best reward

Initialize the MCTS tree as an empty dictionary
for *iteration* in range(*max_iterations*) **do**
  for *simulation* in range(*num_simulations*) **do**
    state ← initial_state
    path ← empty_list
    cumulative_reward ← 0
    while ¬is_empty(*state_stack*) **do**
      state ← *state_stack*.pop()
      if *state* not in *tree* **then**
        action_probs ← reordering_policy_network.predict(*state*)
        Initialize *tree*[*state*] with N, W, Q, P
      end if
      node ← *tree*[*state*]
      action ← select_action(*state*, *node.P*)
      next_state ← next_state(*state*, *action*)
      reward ← evaluate_DRL(*state*, *action*)
      cumulative_reward += *reward*
      path ← (*node*, *action*, *reward*)
      state_stack ← *next_state*
    end while
    Backpropagate along path to update N, W, Q values
  end for
  best_action_sequence ← sequence with the highest cumulative reward
  params ← parameter_policy_network.predict(*best_action_sequence*)
  reward ← evaluate_MEV(*transaction_pool*, *params*)
  rewards ← *reward*
end for
best_reward ← *MAX*(*rewards*)
return *best_action_sequence*, *best_reward*

4.3.2. Reordering policy network based on attention mechanism

To further optimize the transaction ordering scheme, RL-BES introduces a reinforcement learning network based on the attention mechanism. In this model, the policy network employs

multi-head attention and residual connections. These are combined with layer normalization and entropy regularization to efficiently capture the complex relationships in transaction sequences and enhance the network's exploration capability.

The core idea of the network is to use the self-attention mechanism to compute the weight of each transaction in the sequence. Through the multi-head attention mechanism, the network performs weighted summation[39]. This allows the network to simultaneously focus on information at different positions in the transaction sequence, ensuring effective integration of global information.

The model updates the network weights through the backpropagation algorithm, aiming to maximize the reward function for the transaction sequence. At each update step, the model predicts the probability of the next action based on the input state. These probabilities reflect the likelihood of each transaction being selected as the optimal one. Eventually, the model continuously learns and optimizes the reordering strategy for the transaction sequence. The structure of the network is shown in Figure 4, and the detailed design is as follows:

- **Input and fully connected layers**: the input to the reordering strategy network is the feature vector representing the transaction sequence state. First, two fully connected layers are used for feature extraction, mapping to 128-dimensional and 64-dimensional hidden layers, respectively. Each layer uses the ReLU activation function, allowing the network to capture the nonlinear features of the transaction sequence. During this process, residual connections are introduced to prevent vanishing or exploding gradients. This is especially important in deeper networks, where residual connections help by adding the input directly to the output through shortcut connections, enhancing the network's expressiveness.

  To maintain the stability of the network output, layer normalization is applied to ensure the output of each layer is uniformly distributed, reducing fluctuations during training. Additionally, the Dropout mechanism is used to randomly deactivate some neurons, effectively preventing overfitting.

- **Multi-head attention mechanism**: after processing through the initial fully connected layers, the network introduces a multi-head attention mechanism to further capture global dependencies in the transaction sequence.

  The multi-head attention mechanism allows the network to focus on the sequence from multiple perspectives, improving its ability to model complex transaction relationships. Through multiple attention heads, the network learns the associations in the transaction sequence from different feature spaces, enhancing its ability to capture global dependencies. This step concatenates the outputs from each attention head and passes them through another fully connected layer.

- **Output layer**: after processing through the multi-head attention mechanism, the output layer of the network maps the features back to the same dimensions as the input. This generates a probability distribution for the selection of each transaction.

  The Softmax function is used to convert the output into valid probabilities, ensuring that the probability of all transaction sequences falls between 0 and 1 and sums to 1. This enables effective strategy selection.

- **Loss function and policy update**: the loss function of the network uses negative log-likelihood loss, combined with reward signals and an entropy regularization term for optimization. The loss function is expressed as follows:

$$L = -\sum_{i=1}^{n} \log \pi_\theta(a_i|s_i) \cdot r_i - \lambda H(\pi_\theta) \tag{10}$$

where $\pi_\theta(a_i|s_i)$ represents the probability of selecting action a_i under state s_i with policy network parameters θ. r_i is the corresponding reward. $H(\pi_\theta)$ is the entropy of the policy, and λ is the hyperparameter controlling the strength of exploration.

By calculating the logarithm of the action probabilities, the network can guide policy updates while maximizing the expected reward. The introduction of entropy regularization helps to increase the network's exploration ability, preventing it from getting stuck in local optima too early. The exploration weight allows the network to balance between exploration and exploitation when selecting strategies.

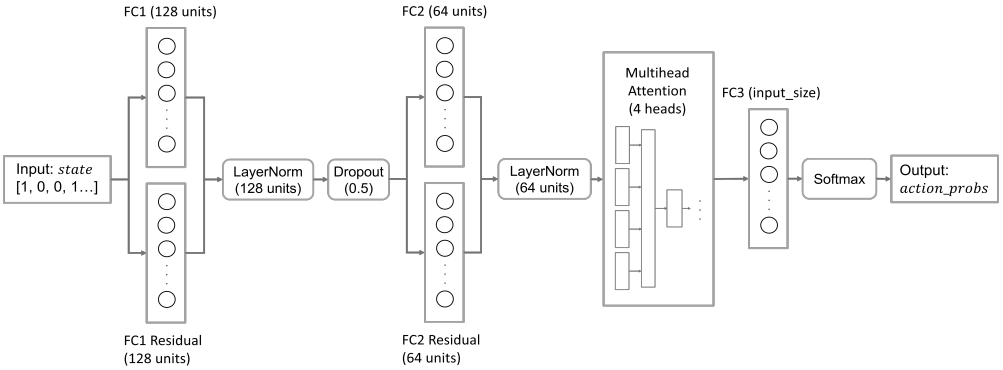

Figure 4. Reordering policy network.

4.3.3. Parameter policy network based on CNN

In terms of parameter optimization, RL-BES uses a reinforcement learning model based on convolutional neural networks (CNNs) to update key parameters in DeFi transactions. CNNs are effective in capturing spatial correlations between parameters and improving optimization performance through layer-by-layer convolution operations[40]. The network represents each parameter as a two-dimensional matrix input, extracts features through convolutional layers, and adjusts adaptively in combination with reinforcement learning. The structure of the network is shown in Figure 5, and the detailed design is as follows:

- **Input and convolutional layers**: the first part of the network uses two convolutional layers with 16 and 32 output channels, respectively, to extract local features from the input transaction state. This state is treated as a one-dimensional data sequence, and the convolutional layers are used to identify important patterns in local regions of the input data.

  Each convolutional layer is followed by a ReLU activation function to introduce nonlinearity. The output of the convolutional layers is flattened into a one-dimensional vector for further processing by the fully connected layers.

- **Fully connected layers**: after the convolutional layers, the output is passed through three fully connected layers. These layers progressively reduce the dimensionality of the data and generate the final parameter values. Each fully connected layer is followed by a ReLU activation function to maintain nonlinearity. Additionally, normalization is applied to each layer to stabilize the training process and ensure standardized outputs across layers.

  Furthermore, after the first two fully connected layers, Dropout layers are introduced to prevent overfitting, ensuring good generalization on unseen data.

- **Output layer**: the output layer of the network is responsible for predicting the specific values of the transaction parameters. After passing through multiple fully connected

layers, the features are mapped back to the same dimensions as the input. The final output is generated through a ReLU activation function.

The main purpose of using the ReLU activation function is to ensure that the predicted parameter values are non-negative, as parameters such as transaction amount and slippage range must be valid non-negative values.

- **Loss function and policy update**: the network's loss function and policy update are consistent with the reordering policy network. It also uses a negative log-likelihood loss, combined with reward signals and an entropy regularization term for optimization.

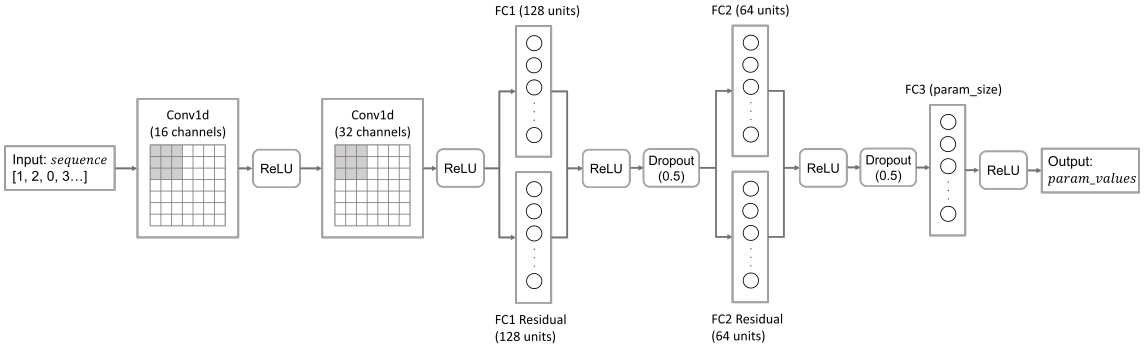

Figure 5. Parameter policy network.

5. Experiment and results

5.1. Tool design

To validate the effectiveness of our RL-BES system in optimizing MEV on the blockchain, we designed a simple experimental tool using Python to simulate a local Ethereum environment. The tool is built on the Hardhat [41] framework, which offers a powerful suite of tools to compile, deploy, and debug smart contracts with ease. One of its standout features is the ability to run a local Ethereum network that can simulate the behavior of the mainnet, allowing for efficient testing and iteration in a safe and controlled environment. We have used it to simulate private forks of Ethereum locally by connecting to an Ethereum archive node via RPC.

Additionally, the evaluation part of the tool incorporates the overall structure of RL-BES, including core functions such as prediction, feedback, and recording. In each experiment, the transaction sequence is processed based on the sorting scheme generated by RL-BES and executed on the local Ethereum private fork. The collected results provide support for subsequent experimental analysis.

5.2. Experiment design

5.2.1. Dataset

For the experiment, we create a dataset which includes historical transaction data from three popular AMMs on the Ethereum blockchain—UniswapV2, Sushiswap, and UniswapV3—along with one of the most popular lending protocols, AaveV2.

Firstly, we choose a set of data primarily consisting of blocks where liquidity events on Sushiswap and UniswapV2 exceed 500 ETH, liquidity events on UniswapV3 exceed 1000 ETH, and liquidation events involving collateral (ETH) on AaveV2. During data collection, each block was uniquely categorized using a priority-based classification approach. Aave, as a

lending protocol, is fundamentally different from decentralized exchange (DEX) protocols. Therefore, when blocks containing transactions that meet the criteria are encountered, those blocks are first selected for inclusion in the Aave-specific dataset, and only after that is the priority relationship between DEX protocols considered. Blocks satisfying the criteria for DEX protocols were assigned with priority to UniswapV3, followed by UniswapV2 and Sushiswap. For example, if a block contained transactions matching both UniswapV3 and UniswapV2 standards, it would be categorized as UniswapV3, and only its UniswapV3 transactions would be semantically modeled. This ensures a one-to-one relationship between blocks and their respective datasets.

Then, to ensure data quality, blocks containing invalid transactions or excessively complex interactions were identified and removed. We define complex transactions as those involving excessively long dependency chains or multiple nested smart contract calls. These transactions tend to introduce significant uncertainty in the modeling process due to their non-linear dependencies, where the outcome of one transaction might depend on multiple preceding transactions. This non-linearity complicates the state space representation, making it harder for the model to accurately predict the effects of these transactions and leading to incorrect estimations of rewards and suboptimal decision-making.

Due to these issues, such blocks with complex transactions are excluded from the dataset. Removing these data helps the model maintain focus on more predictable and consistent patterns, ensuring more reliable and accurate results.

Finally, a total of 286 data entries, representing 3.2% of the dataset, were removed. After carrying out symbolic semantic modeling on these remained data, we have a dataset with a total of 8542 blocks. The data distribution is shown in Table 2.

Table 2. Datasets overview.

Dataset	Blocks
AaveV2	4189
UniswapV3	3121
UniswapV2	869
Sushiswap	363

5.2.2. Experiment baseline and comparison

To measure the performance of RL-BES, we use Flashbots as a baseline model and Lanturn, a similar MEV extraction tool, as the comparison in our experiments.

Flashbots is a widely adopted MEV extraction tool in the blockchain ecosystem, known for its robust transaction bundling mechanism and API standardization. It serves as a practical industry benchmark for evaluating MEV optimization solutions. Flashbots generates optimal transaction orderings through its automatic bidding mechanism, where searchers typically bid for the highest possible MEV profits. Thus, the bid MEV opportunities from the transaction bundles obtained via the Flashbots API [42] serve as the minimum base data for judging the extractable MEV in a given block.

Lanturn proposes a traditional genetic algorithm-based framework for MEV detection, capable of exploring transaction sequences that achieve the optimal MEV within a block, representing traditional heuristic optimization techniques. This makes it an ideal candidate for comparing methodological differences between traditional and reinforcement learning-based approaches. Additionally, its publicly available performance data facilitates objective evaluation. Therefore, To evaluate the advanced nature and efficiency of the RL-BES algorithm,

we compare its performance with that of both Flashbots and Lanturn on the same dataset to verify the effectiveness of our algorithm.

5.3. Experimental setup

Our experiments are run on a high-performance server equipped with 36 Intel(R) Core(TM) i9-10980XE processors and 256 GB of memory. This setup ensures sufficient computational resources and memory support for large-scale transaction reordering tasks. To simulate a real Ethereum blockchain environment, the experiment uses the Ethereum archive node RPC provided by Merkle[43], with Hardhat as the local development environment. The Hardhat version used is 0.8.4, and the simulated private fork is based on the "London" hard fork.

5.4. Experimental results

5.4.1. MEV extraction performance

In order to comprehensively evaluate the MEV extraction performance, we selected a range of Ethereum blocks with block heights from 11834049 to 14986955 as our evaluation range. We conducted 50 independent trials for Lanturn and RL-BES and recorded the average MEV for each block. Within this evaluation, we included several prominent DeFi platforms, including Uniswap, Sushiswap, and AaveV2, to perform the MEV calculation for RL-BES and compare it with Flashbots baseline and Lanturn.

From Figure 6 and Figure 7, we can observe that the RL-BES model significantly outperforms Flashbots baseline in the majority of blocks in terms of MEV extraction, demonstrating both greater stability and higher extraction values. Furthermore, RL-BES exhibits competitive performance against Lanturn, often yielding better results in certain block ranges, thus indicating the efficacy of our reinforcement learning-based optimization for transaction ordering. More importantly, in blocks with higher MEV opportunities, RL-BES consistently extracts more value than the other methods, which demonstrates the significant advantages of RL-BES on these blocks.

To provide a detailed quantitative comparison, we calculated key statistical metrics for each method, including the mean and quartiles (1st quartile, median, and 3rd quartile) of the extracted MEV across the selected blocks. These metrics allow for a deeper understanding of the robustness of the extraction process. Additionally, we also computed the standard deviation and range of the mean to assess the consistency within the MEV values.

Table 3 summarizes the statistical performance of MEV extraction by RL-BES, Flashbots, and Lanturn across the evaluation blocks. RL-BES achieves a mean MEV extraction increase of 337.3% compared to Flashbots baseline and a 57.5% improvement over Lanturn on average.

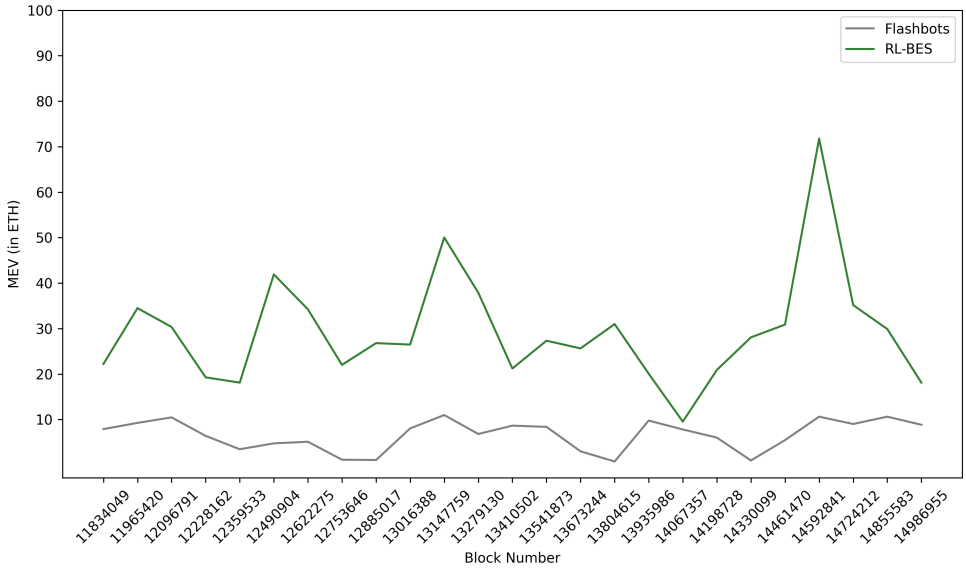

Figure 6. Maximum MEV extracted by RL-BES and flashbots.

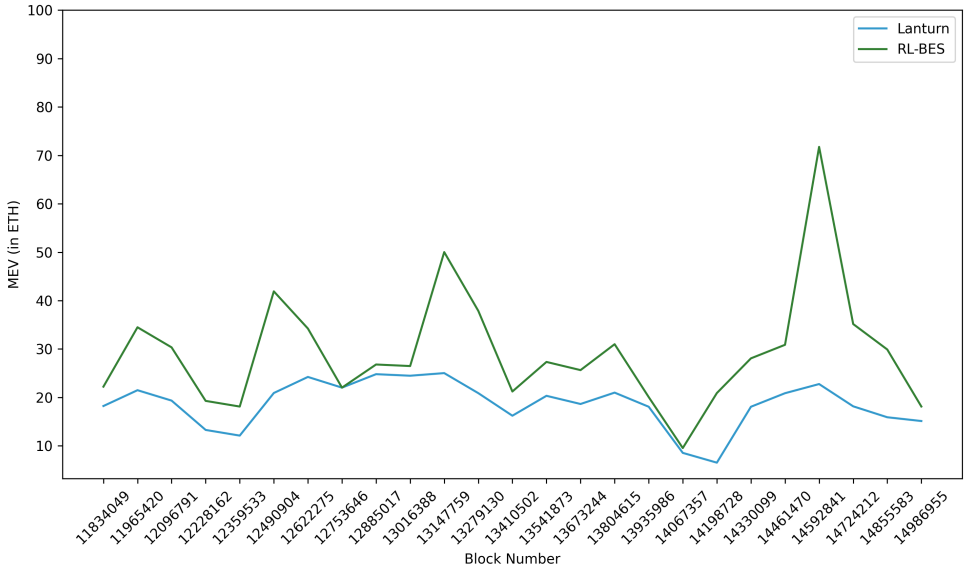

Figure 7. Maximum MEV extracted by RL-BES and Lanturn.

Table 3. Statistical comparison of MEV extraction performance.

Metric	RL-BES (ETH)	Lanturn (ETH)	Flashbots (ETH)
Mean	29.3 (±1.5)	18.6 (±2.2)	6.7
1st Quartile (Q1)	21.2	16.2	5.1
Median	27.3	19.3	7.9
3rd Quartile (Q3)	34.2	21.5	9.0

5.4.2. Runtime performance

In the context of MEV extraction, runtime is a critical factor, especially when comparing different search algorithms. Since the search algorithms of RL-BES and Lanturn differ significantly, it is important to establish a unified method of evaluation. To ensure a fair

comparison, we measured the time taken by each algorithm to find the maximum MEV in their respective search processes. Specifically, we recorded the runtime as the earliest point at which the two algorithms converged on the highest MEV value.

We conducted 50 independent trials for each algorithm, across a block range from 11834049 to 14986955, and recorded the average runtime for each block. By taking the mean of these 50 runs, we obtained a reliable estimate of the average time taken by each algorithm to reach the optimal solution in different blocks.

Figure 8 clearly illustrates the runtime performance comparison between RL-BES and Lanturn. The RL-BES model, which combines reinforcement learning with Monte Carlo Tree Search (MCTS), demonstrates significantly faster convergence than Lanturn. Lanturn, which employs traditional genetic algorithms and Gaussian sampling for exploration, generally requires more time to achieve convergence, as seen by the consistently higher runtimes across the block range. Besides, in blocks with higher transaction complexity, such as block 13541873, 13935986 and 14724212, RL-BES demonstrates a notable reduction in runtime compared to Lanturn.

Table 4 presents a statistical summary of the runtime data collected from the 50 experiments with the associated standard error for both RL-BES and Lanturn. The table includes the mean and variance for each tool's runtime, providing a comprehensive view of their performance. The smaller errors of the mean for RL-BES in most cases indicate that its runtime is more consistent across multiple experiments.

Table 4. Statistical comparison of runtime for RL-BES and lanturn.

Statistic	RL-BES Runtime (s)	Lanturn Runtime (s)
Mean	28.3 (±1.8)	69.1 (±4.5)
Variance	15.3	86.4

As shown in the table, the mean runtime for RL-BES is 28.3 seconds, which is significantly lower than the 69.1 seconds required by Lanturn on average. The variance in runtime for RL-BES is much smaller (15.3 compared to 86.4 for Lanturn), indicating that RL-BES not only converges faster but also exhibits more consistent performance across different blocks.

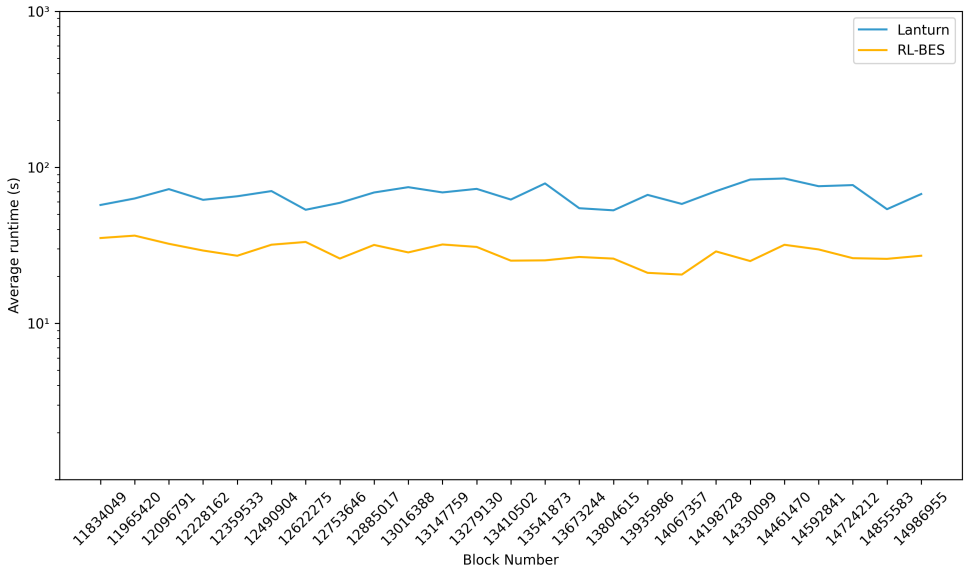

Figure 8. Average runtime of MEV extraction.

5.4.3. Sample efficiency and stability performance

To evaluate RL-BES's performance in a small-sample setting, we conducted experiments on a representative block with height of 14046466, which contains a limited number of complex transactions. The experiments involved 50 independent trials which involve 20 iterations and with different random initializations, focusing on the model's ability to adaptively iterate (sample efficiency) and maintain consistency in convergence behavior (stability).

The results are visualized in Figure 9, which shows the average iterations required for convergence across all experiments. In all experiments, RL-BES achieved convergence in an average of 8 iterations out of a total of 20 iterations. The tight distribution and fast process of convergence reflects its ability to adaptively optimize within a small transaction dataset while maintaining stability under varying initialization conditions.

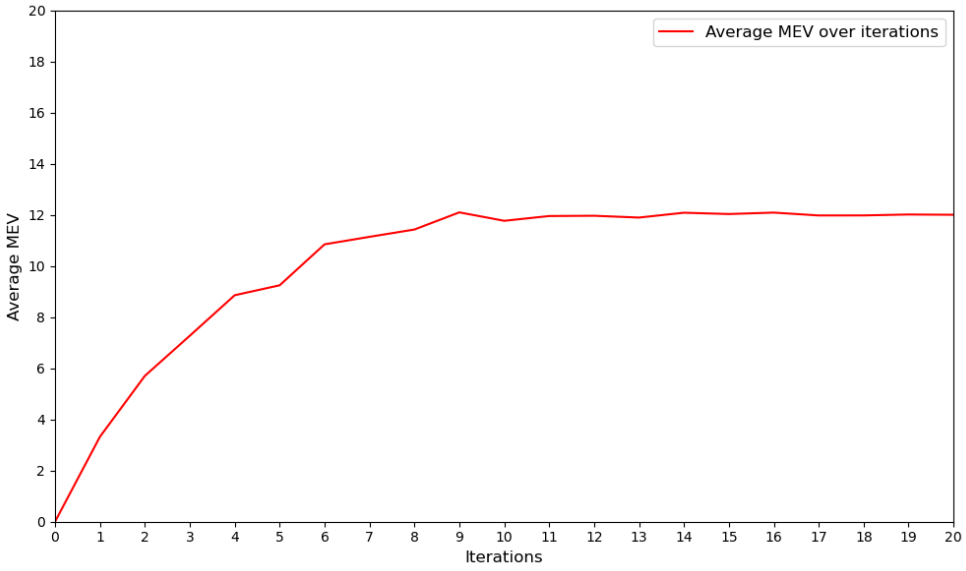

Figure 9. Average iterations for convergence.

6. Discussion

In this paper, we propose and implement a blockchain economic security system based on reinforcement learning, called RL-BES, aimed at optimizing the Maximum Extractable Value (MEV) in blockchain. This section discusses the methodology and the experimental results, analyzes the system's performance advantages and limitations, and suggests potential future improvements.

6.1. Methodology

RL-BES combines model-base and model-free reinforcement learning to optimize transaction sequences and parameters. This choice reflects the strengths and limitations of standard reinforcement learning approaches:

- **Value-based methods:** while effective in discrete action spaces, these methods struggle with RL-BES's continuous action requirements, such as parameter adjustments.
- **Policy-based methods:** though capable of handling continuous tasks, standalone policy optimization methods suffer from high sample inefficiency and slow convergence in complex environments.

- **Actor-critic methods:** these methods strike a balance between value and policy optimization but lack the planning capabilities needed for large-scale search tasks like transaction reordering.
- **Off-policy methods:** off-policy learning, as employed in RL-BES, enables efficient utilization of samples through experience replay, which is critical for MEV extraction tasks that require optimizing over different data.

The hybrid approach in RL-BES leverages MCTS for effective exploration and planning, while policy networks dynamically adapt to blockchain's volatile environments. This integration balances exploration and exploitation, ensuring robust performance in the high-dimensional and dynamic challenges of MEV optimization.

6.2. Effectiveness

The experimental results demonstrate the strong performance of the RL-BES system in MEV extraction. Compared to traditional MEV extraction algorithms, RL-BES's reinforcement learning framework adapts strategies autonomously. Combined with Monte Carlo Tree Search (MCTS), it efficiently compresses the search space for transaction sequences, reducing computational overhead, and quickly identifying hidden arbitrage opportunities. In complex on-chain environments, our method captures interactions between multiple smart contracts with just a simple contract semantics model. It continuously adjusts the transaction ordering strategy to optimize MEV extraction. This versatility gives RL-BES an advantage over traditional methods that rely on static or rule-based searches.

In terms of blockchain economic security, our model enables researchers to quantify and analyze the frequency of MEV attacks and potential security vulnerabilities in various protocols by optimizing MEV extraction and observing outcomes. Besides, our model provides a valuable tool for blockchain security researchers, allowing them to simulate and optimize various scenarios. It also enables the customization of network structures, encouraging researchers to update and improve the model flexibly to further analyze the economic security of blockchains and protocols. Furthermore, by studying improved MEV ordering strategies in historical blocks, researchers can uncover new MEV extraction strategies and potential attack methods.

6.3. Challenges

Despite RL-BES's excellent performance, particularly in handling large-scale blockchain transaction sequences, there are areas that need improvement.

First, the inference and backpropagation processes of the reinforcement learning model demand substantial computational resources. The computational cost of MCTS is primarily driven by the number of simulations and the branching factor of the search tree. RL-BES employs a policy network for leaf node evaluation, reducing the need for exhaustive simulations. Additionally, UCB1-based action selection balances exploration and exploitation, improving efficiency. Reinforcement learning networks also incur costs during training, particularly with attention mechanisms and CNN-based architectures. By utilizing mini-batch training and experience replay, RL-BES mitigate the computational burden while maintaining sample efficiency.

However, as the number of transactions within a block increases, the depth and breadth of the tree grow exponentially. In the face of highly complex transaction sequences, the convergence speed of the model is limited. Therefore, hardware performance and network structure are bottlenecks that must be addressed. Additionally, while MCTS compresses the search space and can explore better transaction sequences, optimizing computational efficiency remains a priority for future work.

The system's scalability across different blockchains is also a significant challenge. Var-

ious blockchains, such as Binance Smart Chain and Polygon, possess distinct technical architectures, transaction rules, and smart contract standards. For instance, some blockchains may adopt different consensus algorithms, which can impact the speed of transaction confirmation and the stability of transaction ordering, thus affecting the RL-BES system's MEV optimization strategies. The differences in smart contract functions and interfaces among blockchains may necessitate readaptation during semantic modeling and parameter adjustment, increasing the complexity of the system.

Besides, the process of modeling new DeFi protocols requires significant manual effort. Specifically, for each protocol, we first gain a comprehensive understanding of the protocol's execution principles, key functions, and interaction methods. Additionally, we also need to obtain relevant transaction data via API calls for the initial semantic modeling process. This step typically takes around one week. Once the semantic model is built, the next step involves replaying and simulating the transactions to verify the correctness of the model by checking the normal execution of blocks, which takes approximately 1 to 2 days.

This dependence on manual modeling and verification impacts the scalability of RL-BES, as the time required for both tasks increases with the addition of more protocols. As such, a future direction of our research will focus on the development of automated tools for protocol modeling and verification, which will improve efficiency and enable the system to scale effectively.

6.4. Potential

The flexibility and scalability of the RL-BES system provide significant potential for its application in real blockchain environments. As blockchain technology and decentralized finance (DeFi) ecosystems continue to evolve, the complexity of on-chain arbitrage and liquidation opportunities will increase. RL-BES offers a promising design approach that can be extended to support more complex smart contracts and transaction logic for MEV detection and extraction, addressing increasingly intricate MEV scenarios.

6.5. Future work

Our future improvements will focus on the following aspects:

- **More efficient model architecture and algorithms**: we plan to introduce more efficient reinforcement learning networks, improving inference speed and feature analysis capabilities by adjusting network architecture and adding new modules. Besides, we aim to improve pruning algorithms and parallelized execution to reduce redundant computations and increase efficiency.
- **Hardware advancements**: RL-BES's runtime is significantly influenced by the performance of computational devices. With advancements in multi-core CPUs and GPUs, the system's runtime can be further reduced. We will explore the integration of high-performance hardware with RL-BES will be crucial for its practical application in real-time environments.
- **Adaptation to diverse DeFi applications**: we aim to enhance RL-BES to support a broader range of DeFi applications, enabling it to better address the challenges posed by the rapidly evolving DeFi landscape.
- **Expansion to multi-chain environments**: while our experiments primarily focused on the Ethereum blockchain, future efforts will extend the RL-BES system to support other major blockchains such as Binance Smart Chain (BSC) and Polygon. Cross-chain MEV extraction will be a key direction for future development.

6.6. *Ethical implications and risks*

While RL-BES provides significant advancements in MEV extraction, its application raises several ethical concerns that warrant careful consideration:

- **Equity and accessibility**: the complexity of RL-BES and its reliance on computational resources may disproportionately benefit teams and institutions with advanced technological capabilities. This could exacerbate inequalities within the blockchain ecosystem, limiting smaller participants' access to the benefits of MEV optimization.
- **Decentralization risks**: large-scale adoption of RL-BES by dominant players may lead to a concentration of MEV rewards, potentially undermining the decentralized ethos of blockchain. Such a scenario risks shifting power dynamics toward a few well-resourced actors, contradicting the principles of financial inclusion.
- **Impact on Ethereum's Proposer-Builder Separation (PBS)**: although PBS is designed to mitigate centralization risks in MEV extraction, the effectiveness of this mechanism could be challenged if advanced tools like RL-BES amplify disparities in technological capacity among builders and validators.

7. Conclusion

In this paper, we present RL-BES, a reinforcement learning-based system for optimizing Maximum Extractable Value (MEV) on blockchains, designed to improve blockchain economic security. By combining deep reinforcement learning with Monte Carlo Tree Search, our system adapts transaction ordering strategies across diverse blockchain transaction sequences, significantly improving MEV extraction efficiency.

Experimental results show that RL-BES performs excellently in various DeFi transaction scenarios, surpassing Flashbot baseline data and traditional MEV extraction methods. Additionally, we developed a customized evaluation tool for the system, allowing us to validate the effectiveness of our models and algorithms while providing support for future research. Moving forward, we plan to continue optimizing the model architecture and inference mechanisms, as well as enhancing its capability to model and adapt to various DeFi applications.

Conflicts of interests

The authors declared that they have no conflicts of interests.

Authors' contribution

Conceptualisation, E.W.; Formal analysis, E.W.; Software, E.W.; Methodology, E.W., Z.D., Y.M.; Writing - original draft, E.W., Z.D., Y.M.; Writing - review and editing, E.W., Z.D., Y.M., Y.Z.; Project administration, X.S., E.W.; Supervision, X.S., E.W., Y.Z.; All authors have read and agreed to the published version of the manuscript.

References

[1] Zheng Z, Xie S, Dai H, Chen X, Wang H. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE international congress on big data (BigData congress)*, Honolulu, HI, USA, June 25–30, 2017, pp. 557–564.
[2] Zheng Z, Xie S, Dai H, Chen X, Wang H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* 2018, 14(4): 352–375.

[3] Dai H, Zheng Z, Zhang Y. Blockchain for Internet of Things: A survey. *IEEE Internet Things J.* 2019, 6(5): 8076–8094.
[4] Zheng Z, Xie S, Dai H, Chen W, Chen X, *et al*. An overview on smart contracts: Challenges, advances and platforms. *Future Gener. Comput. Syst.* 2020, 105: 475–491.
[5] Jensen J R, von Wachter V, Ross O. An introduction to decentralized finance (defi). *Complex Syst. Inform. Model. Q.* 2021, (26): 46–54.
[6] Schueffel P. Defi: Decentralized finance-an introduction and overview. *J. Innov. Manag.* 2021, 9(3): I–XI.
[7] Daian P, Goldfeder S, Kell T, Li Y, Zhao X, *et al*. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE symposium on security and privacy (SP)*. San Francisco, CA, USA, May 18–21, 2020, pp. 910–927.
[8] Kulkarni K, Diamandis T, Chitra T. Towards a theory of maximal extractable value i: Constant function market makers. *arXiv preprint arXiv* 2022:2207.11835.
[9] Makarov I, Schoar A. Trading and arbitrage in cryptocurrency markets. *J. Financ. Econ.* 2020, 135(2): 293–319.
[10] Perez D, Werner S M, Xu J, Livshits B. Liquidations: DeFi on a Knife-edge. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, Revised Selected Papers, Part II 25*. Springer Berlin Heidelberg, March 1—5, 2021, pp 457–476.
[11] Züst P, Nadahalli T, Wattenhofer Y W R. Analyzing and preventing sandwich attacks in ethereum. *ETH Zürich* 2021: 1–29.
[12] Zhou L, Qin K, Cully A, Livshits B, Gervais A. On the just-in-time discovery of profit-generating transactions in defi protocols. In *2021 IEEE Symposium on Security and Privacy (SP)* San Francisco, CA, USA, May 24–27, 2021, pp. 919–936.
[13] Bahrani M, Garimidi P, Roughgarden T. Centralization in block building and proposer-builder separation. *arXiv preprint arXiv* 2024:2401.12120.
[14] Torres C F, Camino R. Frontrunner jones and the raiders of the dark forest: An empirical study of frontrunning on the ethereum blockchain. *30th USENIX Security Symposium (USENIX Security 21)* 2021: 1343–1359.
[15] Qin K, Zhou L, Gervais A. Quantifying blockchain extractable value: How dark is the forest? In *2022 IEEE Symposium on Security and Privacy (SP)* San Francisco, CA, USA, May 22–26 2022, pp. 198–214.
[16] McLaughlin R, Kruegel C, Vigna G. A large scale study of the ethereum arbitrage ecosystem. *32nd USENIX Security Symposium (USENIX Security 23)* 2023: 3295–3312.
[17] Babel K, Daian P, Kelkar M, Juels A. Clockwork finance: Automated analysis of economic security in smart contracts. In *2023 IEEE Symposium on Security and Privacy (SP)* San Francisco, CA, USA, May 21–25, 2023, pp. 2499–2516.
[18] Li Z, Li J, He Z, Luo X, Wang T, *et al*. Demystifying defi mev activities in flashbots bundle. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security* Copenhagen, Denmark, November 26 – 30, 2023, pp. 165–179.
[19] Qin K, Zhou L, Livshits B, Gervais A. Attacking the defi ecosystem with flash loans for fun and profit. *International conference on financial cryptography and data security* 2021: 3–32.
[20] Babel K, Javaheripi M, Ji Y, Kelkar M, Koushanfar F, *et al*. Lanturn: Measuring economic security of smart contracts through adaptive learning. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security* Copenhagen, Denmark, November 26 – 30, 2023, pp. 1212–1226.
[21] Gai Y, Zhou L, Qin K, Song D, Gervais A. Blockchain large language models. *arXiv preprint arXiv* 2023:2304.12749.
[22] Tian X, Zhuang Z, Zhang L. Redesign Incentives in Proof-of-Stake Ethereum: An

Interdisciplinary Approach of Reinforcement Learning and Mechanism Design. In *2024 6th International Conference on Data-driven Optimization of Complex Systems (DOCS)* Hangzhou, China, August 16–18, 2024, pp. 16–24.
[23] Islam T, Bappy F H, Zaman T S, Sajid M S I, Pritom M M A. MRL-PoS: A Multi-agent Reinforcement Learning based Proof of Stake Consensus Algorithm for Blockchain. In *2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC)* Las Vegas, NV, USA, January 08–10, 2024, pp. 0409–0413.
[24] Zhao F, Zhang M, Zhou S, Lou Q. Application of Deep Reinforcement Learning for Cryptocurrency Market Trend Forecasting and Risk Management. *J. Ind. Eng. Appl. Sci.* 2024, 2(5): 48–55.
[25] Alex Obadia. Flashbots—Frontrunning the MEV Crisis. Available: https://writings.flashbots.net/frontrunning-mev-crisis (accessed on 20 January 2024).
[26] Flashbots. An MEV inspector for Ethereum. Available: https://github.com/flashbots/mev-inspect-py (accessed on 20 January 2024).
[27] Weintraub B, Torres C F, Nita-Rotaru C, State R. A flash (bot) in the pan: measuring maximal extractable value in private pools. In *Proceedings of the 22nd ACM Internet Measurement Conference* Nice, France, October 25 – 27, 2022, pp. 458–471.
[28] Makarov I, Schoar A. Cryptocurrencies and decentralized finance (DeFi). *Brook. Pap. Econ. Act.* 2022, 2022(1):141–215.
[29] Capponi A, Jia R. The adoption of blockchain-based decentralized exchanges. *arXiv preprint arXiv* 2021:2103.08842.
[30] Hassija V, Bansal G, Chamola V, Kumar N, Guizani M. Secure lending: Blockchain and prospect theory-based decentralized credit scoring model. *IEEE Trans. Netw. Sci. Eng.* 2020, 7(4): 2566–2575.
[31] Mita M, Ito K, Ohsawa S, Tanaka H. What is stablecoin?: A survey on price stabilization mechanisms for decentralized payment systems. *2019 8th International Congress on Advanced Applied Informatics (IIAI-AAI)* Toyama, Japan, July 07–11, 2019, pp. 60–66.
[32] Piet J, Fairoze J, Weaver N. Extracting godl [sic] from the salt mines: Ethereum miners extracting value. *arXiv preprint arXiv* 2022:2203.15930.
[33] Kaelbling L P, Littman M L, Moore A W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* 1996, 4: 237–285.
[34] Li Y. Deep Reinforcement Learning: An Overview. *arXiv preprint arXiv* 2017:1701.07274.
[35] Puterman M L. Markov decision processes. *Handbooks in operations research and management science* 1990, 2: 331–434.
[36] Świechowski M, Godlewski K, Sawicki B, Mańdziuk J. Monte Carlo tree search: A review of recent modifications and applications. *Artif. Intell. Rev.* 2023, 56(3): 2497–2562.
[37] Kuleshov V, Precup D. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv* 2014:1402.6028.
[38] Aigner A A, Dhaliwal G. Uniswap: Impermanent loss and risk profile of a liquidity provider. *arXiv preprint arXiv* 2021:2106.14404.
[39] Vaswani A. Attention is all you need. *Adv. Neural Inf. Process. Syst.* 2017.
[40] Li Z, Liu F, Yang W, Peng S, Zhou J. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE Trans. Neural Netw. Learn. Syst.* 2021, 33(12): 6999–7019.
[41] NomicFoundation. Hardhat. Available: https://github.com/NomicFoundation/hardhat (accessed on 20 January 2024).
[42] Flashbots. Flashbots API. Available: https://blocks.flashbots.net (accessed on 20 January 2024).
[43] Merkle Software. MEV-protected mempool for Ethereum, BSC, Base and Polygon. Available: https://eth.merkle.io (accessed on 20 January 2024).