
ADAPTIVE LIQUIDITY PROVISION IN UNISWAP V3 WITH DEEP REINFORCEMENT LEARNING

Haochen Zhang

Department of Electrical and Computer Engineering
University of California, Los Angeles
haochen235@ucla.edu

Xi Chen

Leonard N. Stern School of Business
New York University
xc13@stern.nyu.edu

Lin F. Yang

Department of Electrical and Computer Engineering
University of California, Los Angeles
linyang@ee.ucla.edu

ABSTRACT

Decentralized exchanges (DEXs) are a cornerstone of decentralized finance (DeFi), allowing users to trade cryptocurrencies without the need for third-party authorization. Investors are incentivized to deposit assets into liquidity pools, against which users can trade directly, while paying fees to liquidity providers (LPs). However, a number of unresolved issues related to capital efficiency and market risk hinder DeFi's further development. Uniswap V3, a leading and groundbreaking DEX project, addresses capital efficiency by enabling LPs to concentrate their liquidity within specific price ranges for deposited assets. Nevertheless, this approach exacerbates market risk, as LPs earn trading fees only when asset prices are within these predetermined brackets. To mitigate this issue, this paper introduces a deep reinforcement learning (DRL) solution designed to adaptively adjust these price ranges, maximizing profits and mitigating market risks. Our approach also neutralizes price-change risks by hedging the liquidity position through a rebalancing portfolio in a centralized futures exchange. The DRL policy aims to optimize trading fees earned by LPs against associated costs, such as gas fees and hedging expenses, which is referred to as loss-versus-rebalancing (LVR). Using simulations with a profit-and-loss (PnL) benchmark, our method demonstrates superior performance in ETH/USDC and ETH/USDT pools compared to existing baselines. We believe that this strategy not only offers investors a valuable asset management tool but also introduces a new incentive mechanism for DEX designers.

Keywords Uniswap V3, automated market maker, constant product market maker, deep reinforcement learning, loss versus rebalancing

1 Introduction

Traditionally, intermediaries play essential roles in connecting market participants in transactions. However, with the emergence of decentralized finance (DeFi), decentralized exchange (DEX) - as the most important part of DeFi - enables transactions on a peer-to-peer permissionless network and eliminates the necessity of relying on intermediaries, such as brokers and banks [1]. DEX employs an automated market maker (AMM) where the rules to determine exchange rates, as well as to clear demand and supply, are very different from traditional limit order books (LOBs). The constant function market maker (CFMM) is a representative of AMM, and the majority of AMMs belong to this category. CFMMs use a deterministic function $f(x, y) = k$, where x and y represent the reserves of two tokens X and Y , and k represents the depth of the pool, and a few additional rules to build an environment that allows traders to swap tokens and also enable liquidity providers (LPs) to deposit tokens. CFMMs with concentrated liquidity (CL) are the most popular type of CFMMs, and they also present LPs with both opportunities and challenges. In CFMMs without CL, LPs provide liquidity uniformly across the entire price range $(0, \infty)$. However, in CFMMs with CL, LPs specify a

range of contract prices, i.e., prices of token X with respect to token Y in the Uniswap V3 pool, to provide liquidity by depositing the corresponding assets. When the contract price is within this range, LPs are rewarded with trading fees from traders as a fixed percentage of the transaction size. As a prime example of CFMMs with CL, Uniswap V3 has gained significant attention and has attracted more than \$9 billion dollars of value in its liquidity pool. Therefore, a number of papers have studied the approaches on liquidity provision in Uniswap V3[2, 3, 4, 5, 6, 1].

Table 1: Different objective functions in designing liquidity provision strategies.

Reference	Objective Function
[2]	trading fee
[3]	trading fee + change of position value
[4]	trading fee + change of position value
[5]	trading fee + impermanent loss
[6]	trading fee + impermanent loss
[1]	trading fee + loss-versus-rebalancing
this study	trading fee + loss-versus-rebalancing + gas fee

To optimally manage liquidity provision, it is essential to construct an objective function that quantifies the return on investment (ROI) for a liquidity provider (LP). This objective function must account for all relevant costs and returns. For the purpose of this discussion, we assume that both the initial investment and the final ROI are denominated in U.S. dollars (USD). One primary cost incurred by an LP is the blockchain gas fee, significant during the reallocation of liquidity. These fees should not be underestimated, as they substantially influence the frequency of liquidity reallocation. Another variable cost involves changes in position value, attributed to asset price fluctuations. Despite these considerations, existing literature often falls short in offering an accurate profit and loss (PnL) definition for an LP. Table 1 highlights the objective functions adopted in existing works. Notably, many studies, such as [2], neglect gas fees, while others incorporate metrics that do not capture an LP’s true PnL comprehensively.

To rectify this, we propose a PnL definition that includes trading fees, gas fees, and changes in position value, aligning closely with the LP’s actual ROI. However, the absence of an effective hedging mechanism can make the returns unstable due to market volatility. To mitigate this, we propose to adopt a hedging strategy by taking an opposing position in a separate market, such as a futures market. In this framework, we introduce a new PnL metric called Loss-Versus-Rebalancing (LVR), which is defined as the value difference between a rebalanced portfolio and the original liquidity position [7]. Combined with the gas fee and trading fee, our strategy ensures a more realistic ROI despite market fluctuations.

In sum, this paper sets out to accomplish two primary goals:

1. To hedge liquidity provision risks effectively through a rebalancing portfolio, thereby addressing the risk of liquidity provision due to the change of position value and only leaving a residual called LVR which can be seen as the cost of providing liquidity [7, 1].
2. To employ Deep Reinforcement Learning (DRL) algorithms for decision-making. DRL is favored as it is model-free and offers greater flexibility compared to traditional parameterized models like geometric Brownian motion [3, 4].

The main contributions of this paper can be summarized as follows:

(1) We present a more pragmatic measure of PnL tailored specifically for LPs in Uniswap V3. This comprehensive metric includes trading fees, LVR, and gas fees. When LPs hedge their liquidity positions through a rebalancing portfolio, this metric precisely reflects their actual profits.

(2) Our strategy is pioneering in its systematic exploration of a DRL-based strategy for liquidity provision in Uniswap V3. We empirically demonstrate the effectiveness of DRL for optimizing liquidity provision, particularly when market risks are hedged. Our simulation results corroborate that optimizing the PnL of an unhedged liquidity position presents a more complex challenge compared to a hedged position.

(3) Our proposed method enhances liquidity provision strategies, thereby increasing the potential profits for LPs. Importantly, this strategy does not require substantial computational resources and is thus accessible to individual investors. Furthermore, our approach shows considerable advantages over baseline methods, especially for investors with limited initial capital.

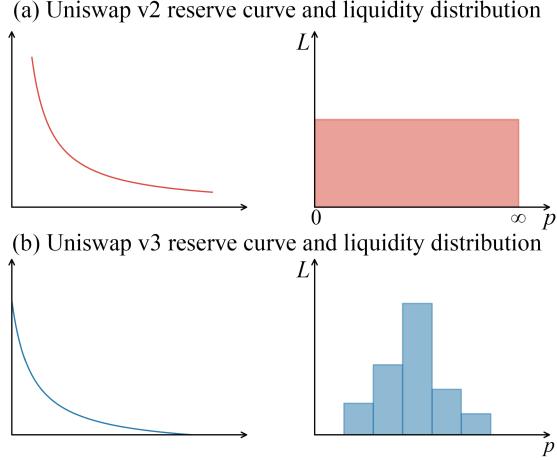


Figure 1: A toy example to show the difference in reserve curve and liquidity distribution between Uniswap V2 and Uniswap V3

The structure of this paper is organized as follows: Section 2 reviews a broad range of applications of RL in finance. Section 3 delineates the Uniswap V3 contract mechanics, introduces the concept of LVR, and provides an overview of RL and the specific DRL algorithm employed in this study. Section 4 outlines how the liquidity provision problem in Uniswap V3 is formulated as a reinforcement learning problem. Section 5 discusses the experimental setup and presents key findings from simulations conducted on historical Uniswap V3 data. Lastly, Section 6 concludes.

2 Related Work

This section provides an overview of the existing literature on the application of DRL in finance, with a focus on domains other than liquidity provision in Uniswap V3. The literature can be broadly categorized into four key areas: Optimal Execution, Portfolio Optimization, Option Pricing and Hedging, and AMMs.

Optimal execution involves the strategic buying or selling of a given asset within a specific time frame while minimizing market impact and price uncertainty. A notable contribution in this area is by [8], which employs both supervised learning and reinforcement learning techniques to develop an optimal stopping formulation for this problem.

Portfolio Optimization addresses the challenge of maximizing expected returns while controlling the risk across a basket of assets. [9] employs a deep Q-learning agent that outperforms traditional benchmarks. It effectively tackles the dimensionality issue by using a discrete action space and introduces a mapping function to handle infeasible actions. [10] streamlines automated trading strategies with their *FinRL* framework.

In options trading, the goal is often to balance the costs and benefits of various hedging strategies. While classical models like Black-Scholes [11] and Merton [12] offer theoretical insights, these models often fall short in real-world conditions due to trading frictions. [13] explores this challenge and demonstrates that reinforcement learning can effectively optimize hedging strategies under realistic market conditions.

In the rapidly evolving field of AMMs, reinforcement learning is also gaining traction for designing more efficient market-making algorithms. For instance, [14] employs an actor-critic method to explore the profitability of liquidity provision in constant product markets, suggesting the inclusion of flexible fee tiers for liquidity providers. Complementing this, [15] integrates LSTM with Q-learning to improve the performance of predictive AMMs, building upon the foundation laid by Uniswap V3.

3 Preliminary

3.1 Uniswap V3

A pool in Uniswap V3 is specified by three features: token1, token2, and fee tier. Uniswap V3 discretizes prices using ticks, where the price corresponding to the i^{th} tick is 1.0001^i . Tick spacing, the finest width of the liquidity interval, is determined by the fee tier [16]. For instance, 1% pools have a tick spacing of 200 ticks, 0.3% pools have a tick spacing

of 60 ticks, and 0.05% pools have a tick spacing of 10 ticks. This paper focuses on Uniswap V3 pools with a stable coin and a volatile coin, but can be expanded to pools with two volatile coins without difficulty. We assume token1 is a volatile coin and token2 is a stable coin herein. We use X , Y , x , y , δ , and d to represent the volatile coin, the stable coin, reserve of the volatile coin, reserve of the stable coin, fee tier, and tick spacing, respectively.

The key difference between Uniswap V3 and Uniswap V2 lies in the introduction of Concentrated Liquidity (CL) in Uniswap V3, leading to a non-uniform distribution of liquidity, as illustrated in Figure 1. In Uniswap V2, the reserve curve is straightforward and neat, described by the equation $x \cdot y = L^2$. In contrast, the reserve curve in Uniswap V3 operates within a defined price range $[p_a, p_b]$, where a consistent amount of liquidity units denoted as L is maintained. The relationship between the reserves of assets X and Y is expressed as $(x + L/\sqrt{p_b}) \cdot (y + L\sqrt{p_a}) = L^2$. The contract price of asset X , represented as p , can be calculated using the equation $p = (y + L\sqrt{p_a}) / (x + L/\sqrt{p_b})$ when the price falls within the interval $[p_a, p_b]$. The reserves of assets X and Y within a pool can be expressed in terms of p , as shown in equation (1).

$$(x, y) = \begin{cases} \left(L \left(\frac{1}{\sqrt{p_a}} - \frac{1}{\sqrt{p_b}} \right), 0 \right) & p \leq p_a \\ \left(L \left(\frac{1}{\sqrt{p}} - \frac{1}{\sqrt{p_b}} \right), L \left(\sqrt{p} - \sqrt{p_a} \right) \right) & p_a < p < p_b \\ (0, L \left(\sqrt{p_b} - \sqrt{p_a} \right)) & p \geq p_b \end{cases} \quad (1)$$

Then we consider how to calculate trading fees for an LP who has a liquidity position with L units of liquidity in the range $[p_a, p_b]$. If a swap happens and the contract price moves from p to p' . The LP can obtain a trading fee from the trader who initiates the swap in two scenarios as depicted in equation (2). The first case corresponds to an upward price movement when a trader holds Y and swaps it for X , whereas the second case involves a downward price movement when some X flows into the pool and some Y flows out.

$$\begin{cases} [p_a, p_b] \cap [p, p'] \neq \emptyset & \text{if } p \leq p' \\ [p_a, p_b] \cap [p', p] \neq \emptyset & \text{if } p > p' \end{cases} \quad (2)$$

To better explain how to calculate trading fees, let's consider a specific scenario where both p and p' fall within the interval $[p_a, p_b]$, and the liquidity distribution in this interval is uniform, serving as a simple example. If $p \leq p'$, then the change in the reserve of Y in the liquidity pool, denoted by Δy , is positive and can be calculated using (1): $\Delta y = L_{\text{total}} (\sqrt{p'} - \sqrt{p})$. Here, L_{total} represents the total liquidity within $[p_a, p_b]$. Since a proportion of $(1 - \delta)$ of Y enters the liquidity pool, while a proportion of δ of Y is skimmed as LPs' fee, the total fee for all LPs holding liquidity in the range $[p_a, p_b]$ is $\frac{\delta \Delta y}{1 - \delta} = \frac{\delta}{1 - \delta} L_{\text{total}} (\sqrt{p'} - \sqrt{p})$. The total trading fee is distributed to LPs in proportion to the amount of liquidity units they provide. Thus, the trading fee for an LP holding L liquidity units is $\frac{\delta}{1 - \delta} L (\sqrt{p'} - \sqrt{p})$, and this value is independent of L_{total} . In the event of a downward price movement, the trading fee value in terms of Y for the LP can be calculated in a similar manner [5]. Equation (3) summarizes the formulas for the two cases mentioned above.

$$fee = \begin{cases} \frac{\delta}{1 - \delta} L (\sqrt{p'} - \sqrt{p}) & \text{if } p \leq p' \\ \frac{\delta}{1 - \delta} L \left(\frac{1}{\sqrt{p'}} - \frac{1}{\sqrt{p}} \right) p' & \text{if } p > p' \end{cases} \quad (3)$$

In a broader context, it is essential to acknowledge that (2) may not hold true. However, a swap can be decomposed into several parts if the price movement extends beyond the bounds of an LP's liquidity position [5]. As shown in Figure 2, a price movement from p to p' when $p \in [p_a, p_b]$ but $p' \notin [p_a, p_b]$ can be decomposed into two sequential movements: first from p to p_b , and then from p_b to p' . And the LP is rewarded with trading fees only in the first price movement from p to p_b . It is noteworthy that a price movement leads to a change in the liquidity position's value, which can be expressed as equation (4), where x' and y' can be calculated by substituting p' into equation (1).

$$\Delta V = p'x' + y' - (px + y) \quad (4)$$

3.2 Loss-Versus-Rebalancing of A Liquidity Provider

Loss-versus-rebalancing (LVR) is proposed by [7], where the PnL of a liquidity position without receiving any trading fee is shown to be composed of the PnL of a rebalancing strategy that can be hedged by its opposite position and a predictable residue that represents the cost of providing liquidity, i.e., LVR. For clarity, this subsection first defines LVR for a discrete price trajectory, which directly determines the reward function of a DRL agent, then recapitulates

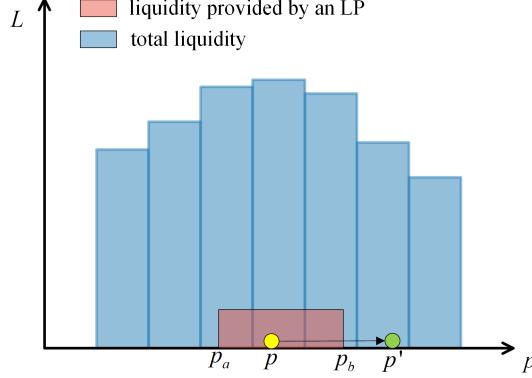


Figure 2: An example of swap decomposition when calculating LP's income

the derivation of LVR under the setting of a geometric Brownian motion contract price and explains LVR is always non-positive.

Given a discrete trajectory of contract price $\{p_t\}_{t=1\dots N}$, LVR can be expressed by (5).

$$\begin{aligned} \text{LVR} &= \sum_{t=1}^{N-1} \{V(p_{t+1}) - V(p_t) - x(p_t)(p_{t+1} - p_t)\} \\ &= \sum_{t=1}^{N-1} \{p_{t+1}[x(p_{t+1}) - x(p_t)] + y(p_{t+1}) - y(p_t)\} \end{aligned} \quad (5)$$

where $V(p_{t+1}) - V(p_t)$ represents the change of position value, and $-x(p_t)(p_{t+1} - p_t)$ is the return of shorting rebalancing portfolio.

Based on equation (5), to avoid the market risk of a liquidity position, one can invest in a portfolio that consists of a fixed liquidity position and the opposite position of its corresponding rebalancing strategy. The liquidity position is used to earn trading fees in Uniswap V3, whereas the short position is for hedging the market risk of the liquidity position. Therefore, the PnL of the whole portfolio can be defined as $\text{PnL}_{\text{static}} = \text{Fee} + \text{LVR}$, where Fee is the cumulative trading fee during the period when the liquidity position is not adjusted from $t = 1$ to $t = N$.

For a given liquidity position, i.e., with fixed p_a , p_b , and L , its value can be written as equation (6), where $x(p_t)$ and $y(p_t)$ denote the amounts of X and Y reserved in the liquidity position. If the contract price evolves according to a geometric Brownian motion, i.e., $dP_t/P_t = \sigma dB_t$, where B_t is a Brownian motion, dV_t can be obtained by applying the Itô lemma to $V(p_t)$ [7], as shown in equation (7).

$$V(p_t) = p_t x(p_t) + y(p_t) = \begin{cases} L \left(\frac{1}{\sqrt{p_a}} - \frac{1}{\sqrt{p_b}} \right) p_t & p \leq p_a \\ 2L\sqrt{p_t} - \frac{L}{\sqrt{p_b}} p_t - L\sqrt{p_a} & p_a < p < p_b \\ L \left(\sqrt{p_b} - \sqrt{p_a} \right) & p \geq p_b \end{cases} \quad (6)$$

$$\begin{aligned} dV(p_t) &= V'(p_t) dp_t + V''(p_t) (dp_t)^2 \\ &= V'(p_t) dp_t + V''(p_t) \sigma^2 p_t^2 dt \\ &= x(p_t) dp_t + V''(p_t) \sigma^2 p_t^2 dt \end{aligned} \quad (7)$$

where $V'(p_t)$ and $V''(p_t)$ denote the first derivative and the second derivative of V with respect to p_t .

A rebalancing strategy with respect to a liquidity position is a self-financing strategy whose holdings are defined by a stochastic process $(x_r(t), y_r(t))$, representing the holdings of X and Y in the rebalancing portfolio at time t , where $x_r(t) = x(p_t)$ for all t . Under the assumption of frictionless trading and in the absence of trading fees, the PnL of a rebalancing strategy is $x(p_t) dp_t$, precisely mirroring the first term on the right-hand side of (7). Therefore, a part of $dV(p_t)$ can be hedged by taking the opposite position of the rebalancing portfolio[7]. By taking the second derivative

of equation (6), $V''(p_t)$ can be expressed by equation (8); and it is always non-positive. Hence the second term on the right-hand side of (7) can be regarded as a loss term and it is defined as loss-versus-rebalancing (LVR). Therefore, LVR can be regarded as the expense associated with providing liquidity, while the PnL of the rebalancing strategy can be seen as the market risk of providing liquidity.

$$V''(p_t) = \begin{cases} -\frac{L}{2p_t^{1.5}} & p_a < p_t < p_b \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

3.3 Deep Reinforcement Learning

Given that the liquidity provision problem is structured as a Markov decision process with a discrete action space, this research employs the Dueling DDQN [17], to acquire adaptive strategies for liquidity provision. In this section, we commence by introducing the notations commonly utilized in general reinforcement learning problems. Subsequently, we delve into the fundamental principles of Dueling DDQN.

We formulate the optimal liquidity provision problem as a policy search problem in a Markov decision process denoted by $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r)$. \mathcal{S} and \mathcal{A} denote state space and action space, $\mathcal{T} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \times \mathbb{R} \rightarrow \mathbb{R}$ represents state transition probability, i.e., dynamic of the environment, $\mathcal{T}(s', r, s, a) = \Pr\{S_{t+1} = s', R_t = r | S_t = s, a_t = a\}$. $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$ denotes reward function, $r(s_t, a_t)$ is reward received by the agent after make action a_t at state s_t . An agent's action is determined by a policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, where π must satisfy $\sum_{a_t \in \mathcal{A}} \pi(a_t | s_t) = 1$. $\pi(a_t | s_t)$ represents the probability of taking action a_t at state s_t . The optimal policy maximizes discounted return defined as $G_t = \sum_{\tau=t+1}^{\infty} \gamma^{\tau-t-1} r_{\tau}$, where $\gamma \in (0, 1)$ is discount factor. State-value function is defined as $V^{\pi}(s) = \mathbb{E}_{\pi, \mathcal{T}}[G_t | S_t = s]$ and action-value function is defined as $Q^{\pi}(s, a) = \mathbb{E}_{\pi, \mathcal{T}}[G_t | S_t = s, A_t = a]$. The optimal action-value function is defined as

$$Q^*(s, a) = \max_{\pi} \mathbb{E}_{\pi, \mathcal{T}}[G_t | S_t = s, A_t = a]$$

Q^* satisfies Bellman optimality equation, as shown in (9). Advantage function is defined as $A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$, it provides a relative measure of the importance of each action. One way to estimate $Q^*(s, a)$ is using the Bellman optimality equation to do an iterative update, as shown in (10) [18].

$$Q^*(s, a) = \sum_{s', r} \mathcal{T}(s', r, s, a) \left[r + \gamma \max_{a'} Q^*(s', a') \right] \quad (9)$$

$$Q^{(i+1)}(s, a) \leftarrow \sum_{s', r} \mathcal{T}(s', r, s, a) \left[r + \gamma \max_{a'} Q^{(i)}(s', a') \right] \quad (10)$$

Deep Q-learning is a value-based algorithm that uses a neural network (DQN) to estimate action-value function, i.e., $\hat{Q}(s, a; \theta) \approx Q^*(s, a)$. In particular, Deep Q-learning collects transitions by applying ϵ -greedy policy and stores them in a replay buffer, and after making an action at a state s_t , it samples a minibatch of transitions from the replay buffer and performs gradient descent to minimize loss function defined by (11). Details of DQN can be seen in **Algorithm 1** in [19].

$$L(\theta) = \mathbb{E}_{(S_t, A_t) \sim D} \left[\left(\hat{Q}(S_t, A_t; \theta) - Y_t^{\text{DQN}} \right)^2 \right] \quad (11)$$

where $Y_t^{\text{DQN}} = R_t + \gamma \max_{a'} \hat{Q}(S_{t+1}, a'; \theta)$ is the target for a transition from the replay buffer.

[20] shows that using the same value for selection and evaluation leads to overoptimistic estimates. And [21] shows that overestimations in DQN is severe and proposes double DQN borrowing the idea from double Q-learning. Double DQN adopts a target network and replace the target with $Y^Q = R_t + \gamma \hat{Q}(S_{t+1}, \text{argmax}_a \hat{Q}(S_{t+1}, a; \theta'); \theta)$, where θ and θ' are parameters of local Q-network and target Q-network, respectively. Dueling DDQN [17] is a further improvement of DDQN that introduces a dueling network architecture, which estimates action-value function by applying (12). Pseudo-code of Dueling DDQN can be seen in **Algorithm 1** in [17].

$$\hat{Q}(s, a; \theta, \alpha, \beta) = \hat{V}(s; \theta, \beta) + \left[\hat{A}(s, a; \theta, \alpha) - \frac{1}{|\mathcal{A}|} \sum_{a'} \hat{A}(s, a'; \theta, \alpha) \right] \quad (12)$$

Table 2: Set of features adopted to construct state

Feature Name	number of features
hourly open price	1
hourly highest price/hourly open price	1
hourly lowest price/hourly open price	1
hourly close price/hourly open price	1
hourly trading volume in USD	1
Double Exponential Moving Average/hourly open price	1
Parabolic SAR/hourly open price	1
Average Directional Movement Index	1
Absolute Price Oscillator	1
Aroon Oscillator	1
Balance Of Power	1
Commodity Channel Index	2
Chande Momentum Oscillator	1
Directional Movement Index	1
Minus Directional Movement	1
Momentum	1
Plus Directional Movement	1
TRIX	1
Ultimate Oscillator	1
Stochastic Momentum Indicator	3
Stochastic Fast Momentum Indicator	3
Normalized Average True Range	1
True Range	1
Hilbert Transform - Dominant Cycle Period	1
Hilbert Transform - Dominant Cycle Phase	1

4 Method

4.1 Optimal Liquidity Provision

The optimal liquidity provision study focuses on the strategy that maximizes the profit of an LP who holds a certain amount of capital and participates in providing liquidity in Uniswap V3. This problem is studied in discrete time in this context, allowing an LP to adjust liquidity positions at evenly distributed time steps $t \in \{1, 2, \dots, T\}$.

4.2 Reinforcement Learning Formulation

This subsection provides a description of states, actions, rewards, and system dynamics in the reinforcement learning formulation of optimal liquidity provision in Uniswap V3. The time period is divided into hourly intervals, and LPs make decisions at the end of each hour. To reduce the learning difficulty for RL agents, this paper does not consider a general liquidity provision strategy that can hold a liquidity position at any tick range, which is intractable due to the dimensionality of state and action. Instead, it focuses on studying the optimal uniform interval reallocation strategy. Under this problem setting, an LP can hold a liquidity position in a tick interval $[tick_l, tick_r]$ with liquidity units of L . At the end of each hour, the LP can choose whether to reallocate the liquidity position.

4.2.1 State

The state at time step t , denoted as $s_t \in \mathcal{S}$, consists of market information and the state of the liquidity position. Specifically, we define $s_t = [\mathbf{f}_t, c_t, m_t, w_t, l_t] \in \mathbb{R}^{32}$, where $\mathbf{f}_t \in \mathbb{R}^{28}$ and c_t, m_t, w_t , and l_t are real numbers. \mathbf{f}_t represents the market information features at time step t . c_t is the amount of USD the agent holds at time step t . m_t denotes the central tick of the liquidity position, while w_t represents the width of the liquidity interval. l_t is the value of the liquidity position at time step t , measured in USD. In the experiment, l_0 is set to different values to demonstrate the differences in optimal liquidity provision with varying initial funds.

The feature vector \mathbf{f}_t includes both basic market information, such as hourly open, high, low, and close prices, and technical indicators that incorporate information about the market trend in the past few hours, as presented in Table 2.

Using w_t and m_t , the lower bound and the upper bound of an LP's liquidity position can be determined by $m_t - d \cdot w_t$ and $m_t + d \cdot w_t$. The amount of liquidity is determined by variables such as w_t , m_t , l_t , and the hourly close price. Therefore, the information integrated into the state s_t is comprehensive.

4.2.2 Action

In Uniswap V3, continuous prices are mapped to discrete ticks. The action space in this study is accordingly set to be discrete: $\mathcal{A} = \{0, 1, 2, \dots, N_a\}$, where N_a represents the maximum width of the interval. An action $a_t = 0$ indicates that the agent does not make any adjustment to the liquidity position and maintain the current liquidity position. For $a_t > 0$, the agent intends to withdraw its liquidity investment, reinvesting the funds (including trading fees earned) into a new liquidity position centered at the current tick with a width of $w_{t+1} = a_t$.

4.2.3 Reward

This study adopts two settings and the corresponding two reward functions. In the first setting, the LP can reallocate the liquidity position and rebalance the short position to hedge against market risk related to the liquidity position. In this context, it is necessary to consider the trading fee and LVR of a fixed liquidity position during the time between the end of the t^{th} hour and the end of the $t + 1^{\text{th}}$ hour, along with the gas fee incurred by the reallocation at the end of the t^{th} hour. These factors are taken into account when calculating the reward for each transition (s_t, a_t, r_t, s_{t+1}) . Therefore, the reward for a transition is defined as $r_t := -\mathbf{1}_{\{a_t \neq 0\}} + \text{Fee}_t + \text{LVR}_t$, where $\mathbf{1}_{\{\cdot\}}$ is the indicator function, and $-\mathbf{1}_{\{a_t \neq 0\}}$ denotes the gas fees incurred by liquidity reallocation. The values of Fee_t and LVR_t can be computed using the formulas mentioned in Section 3.2.

In the second setting, where an LP reallocates the liquidity position without hedging, it is necessary to consider the trading fee and change in position value of a fixed liquidity position during the time between the end of the t^{th} hour and the end of the $t + 1^{\text{th}}$ hour, along with the gas fee incurred by the reallocation at the end of the t^{th} hour. Therefore the reward function is defined as $r_t := -\mathbf{1}_{\{a_t \neq 0\}} + \text{Fee}_t + \Delta V_t$.

4.2.4 System Dynamics Modeling

To let readers better understand reinforcement learning modeling herein, the authors give a simple example that the agent takes action $a_t \neq 0$ to explain the environment dynamics. Given state $s_t = [\mathbf{f}_t, c_t, m_t, w_t, l_t]$ and a_t , $s_{t+1} = [\mathbf{f}_{t+1}, c_{t+1}, m_{t+1}, w_{t+1}, l_{t+1}]$ can be obtained by the following procedure:

- 1): $m_{t+1} \leftarrow d \cdot \text{round}(\text{price2tick}(c_t), d)$
- 2): $w_{t+1} \leftarrow a_t$
- 3): calculate u using $c_t + l_t$, m_{t+1} , w_{t+1} , and close_t
- 4): obtain \mathbf{f}_{t+1} after moving to the end of the $(t + 1)^{\text{th}}$ hour
- 5): $c_{t+1} \leftarrow c_t + \text{Fee}_t$
- 6): calculate l_{t+1} by using m_{t+1} , w_{t+1} , u , and close_{t+1} .

where close_t denotes the close contract price at the t^{th} hour, u denote the amount of liquidity units, $\text{price2tick}(\cdot)$ denotes the function that converts contract price to tick.

5 Experiment

5.1 Experiment setting, Baselines, and Indices for Evaluation

In the experiments presented herein, a few assumptions are adopted as follows:

Assumption 1 *The gas fee for each liquidity reallocation, which encompasses the sum of the gas fees to withdraw liquidity, adjust holdings, and invest liquidity again, is assumed to be a flat fee.*

Assumption 2 *Dynamic trading applying the rebalancing strategy is conducted without friction.*

Assumption 3 *Slippage is neglected when the agent adjusts holdings after each liquidity withdrawal.*

Assumption 4 *The actions taken by the agent have no effect on the behaviors of other players in the market.*

Assumption 1 is reasonable and this assumption is also adopted in [1]. Assumption 2 is reasonable because trading friction on Binance is at most 0.1% and the trading volume of a rebalancing strategy is small. Assumption 3 is

Table 3: Comparison between The Proposed Method and Baselines

	Methods	Adaptive?	Controllable Hold Period?	Controllable Liquidity Spread?	Consider Gas Fee?
M1	the proposed method	✓	✓	✓	✓
M2	uniform τ -reset strategy[2]	✗	✓	✗	✗
M3	EWA[3]	✓	✗	✓	✗
M4	dynamic programming[1]	✓	✗	✓	✗

Table 4: Partition of Dataset

	training set	validation set	test set
(1)	2021/08/02~2022/07/01	2022/07/01~2022/08/11	2022/08/12~2022/09/22
(2)	2021/09/12~2022/08/11	2022/08/12~2022/09/22	2022/09/22~2022/11/03
(3)	2021/10/24~2022/09/22	2022/09/22~2022/11/03	2022/11/03~2022/12/14
(4)	2021/12/05~2022/11/03	2022/11/03~2022/12/14	2022/12/15~2023/01/25

reasonable when the LP’s position value is not too large. Assumption 4 is reasonable when the LP’s position value is not too large, and it is necessary for simulation based on historical data.

The proposed method is a Dueling DDQN-based approach that can adaptively control both the hold period and the liquidity spread. During training, the Dueling DDQN method employs an ϵ -greedy strategy, while during testing, it employs a deterministic strategy. The baseline methods include the uniform τ -reset strategy [2], the exponential weights adaptive strategy (EWA) [3], and the dynamic programming-based method proposed by Cartea et al. [1]. The differences between the proposed method and the baselines are outlined in Table 3. Specifically, the uniform τ -reset strategy involves allocating liquidity uniformly over a range of τ tick ranges centered on the current contract price and resetting when the contract price moves out of the liquidity interval. Thus, the uniform τ -reset strategy can control the hold period, but the liquidity spread is determined by the hyperparameter τ . EWA adopted herein differs slightly from the original version; it does not reallocate the liquidity interval every hour, but introduces an additional hyperparameter T_{re} which denotes the minimum number of hours the agent holds the liquidity position after each reallocation. EWA is a bandit algorithm that cannot balance short-term and long-term profit. The pseudo-code for EWA is provided in **Algorithm 1** in Appendix A. The dynamic programming-based method from [1] can adaptively adjust the liquidity spread but is unable to determine a suitable time to reallocate the liquidity position. For simplicity, the methods are referred to as M1~M4 in the subsequent content.

M1 is trained and tested on data from the ETH/USDC-0.3% pool and ETH/USDT-0.3% pool during the period from August 2, 2021, to January 25, 2023. The original dataset is obtained from the Subgraph (<https://thegraph.com/hosted-service/subgraph/uniswap/uniswap-V3>) and is divided into four periods, as detailed in Table 4. Each period comprises 8000 hours of data for training, 1000 hours for validation, and 1000 hours for testing. Figure 3 displays the contract price in the four test periods for the ETH/USDC-0.3% pool. Due to the similar general trend of the contract price in the ETH/USDT-0.3% pool, it is not shown here. For the hyperparameters in M2~M4, we select the best hyperparameters based on their performance on the test set. It is important to note that this approach yields stronger baselines than those encountered in real-world scenarios since the data in the test set is unknown in practice. While the hyperparameters in M1 are not fine-tuned, early stopping is employed to prevent overtraining. Detailed information about important hyperparameters and further implementation details for all methods can be found in Appendix A. For a more comprehensive understanding of the implementations of the aforementioned methods, please refer to our code on GitHub: <https://github.com/HaochenZhang717/Uniswap-v3.git>. The repository will become publicly accessible upon the acceptance of this paper.

To evaluate the performance of different method, relative PnL is adopted herein. relative PnL in a period is defined as

$$\text{PnL}_{relative} = \frac{\sum_{t=0}^{T-1} r_t}{l_0},$$

where l_0 denotes the initial fund, and the definition of r_t can be seen in Section 4.2.3, which can be different depending hedging is used or not.

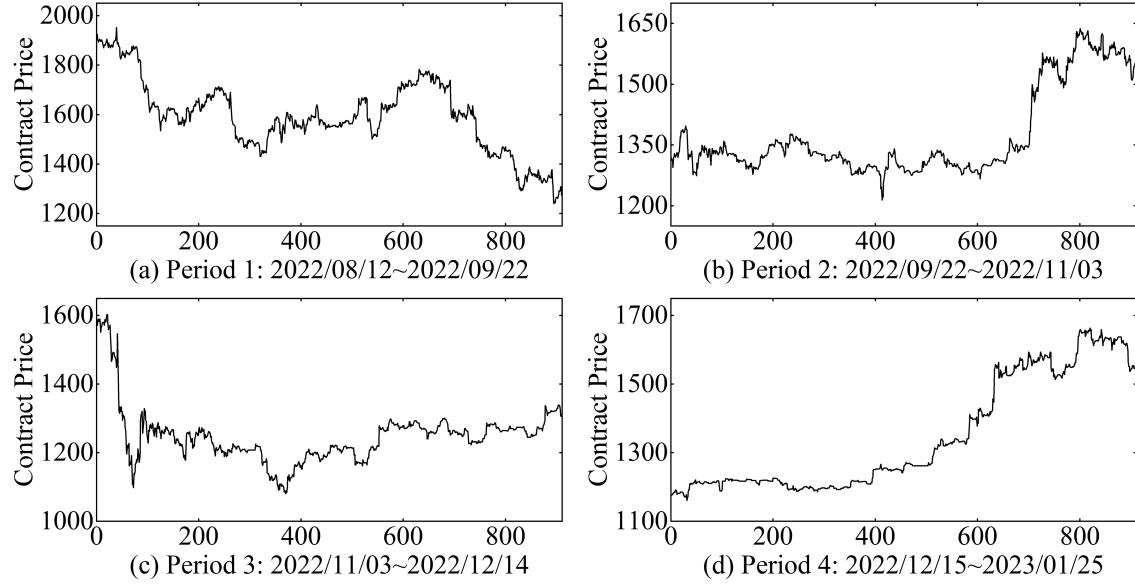


Figure 3: Contract price of ETH/USDC-0.3% pool in different periods

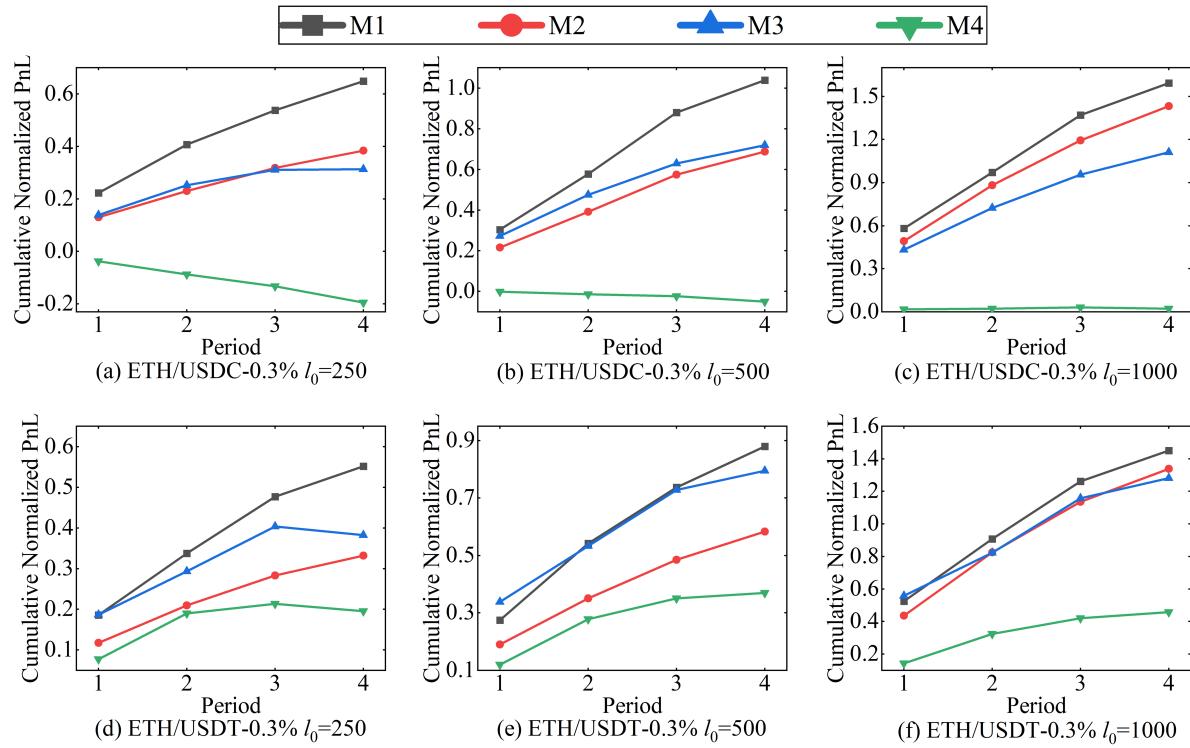


Figure 4: Cumulative normalized PnL of different methods from Period 1 to Period 4 in ETH/USDT 0.3% pool and ETH/USDC 0.3% pool with different initial fund $l_0 = 250, 500, 1000$

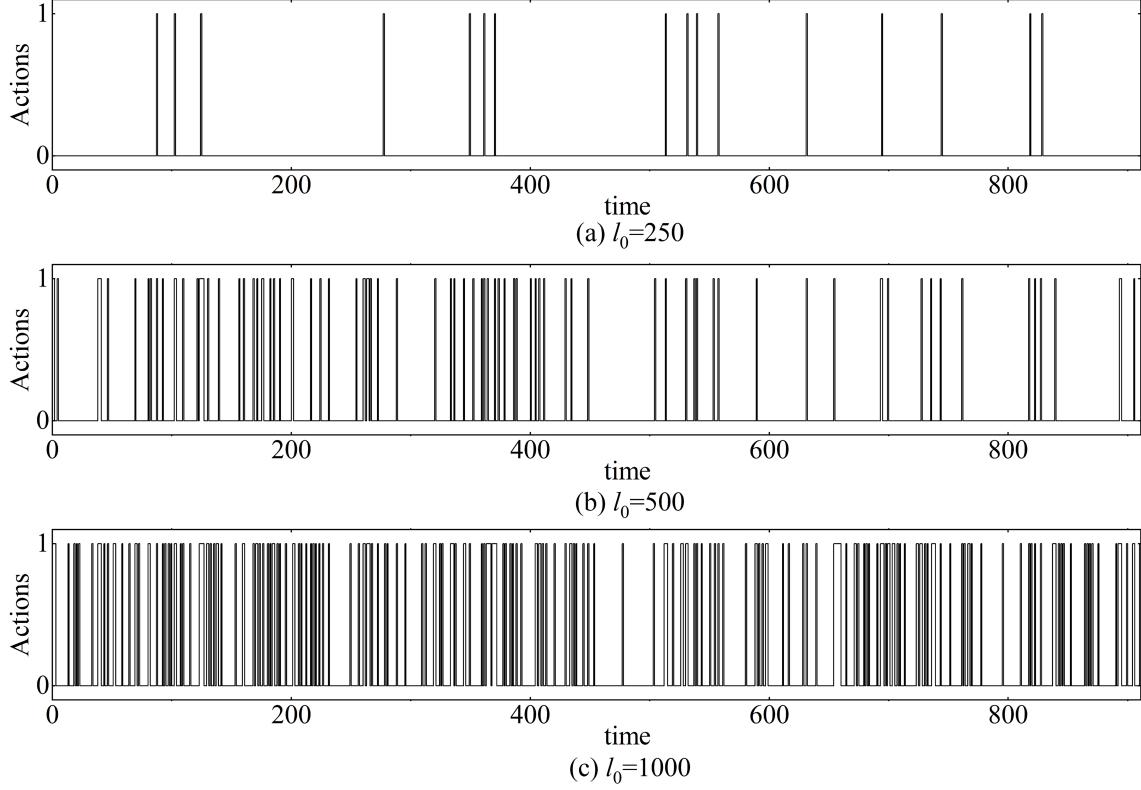


Figure 5: Actions made by M1 in Period 1 in ETH/USDT-0.3% pool (M1 also optimizes liquidity spread, but the result shows it always prefers narrow interval with width equals 1)

5.2 Result and Discussion

This section’s first part discusses the efficacy of the proposed method from two perspectives. Firstly, it is shown that M1 has a superiority over M2~M4 in the sense that M1 is more profitable when hedging is combined with liquidity provision. Secondly, it is demonstrated that the RL agent can hardly learn a good strategy without hedging, indicating that the combination of hedging and liquidity provision can reduce the learning difficulty for the RL agent. The second part of this section shows the optimal LP behavior with different initial funds. As strictly optimal strategies are challenging to determine, M1’s strategy is taken as an approximation.

When market risk is hedged by shorting the rebalancing portfolio, M1 is the most rewarding strategy, as depicted in Figure 4. M1~M4 are tested with varying initial funds. It is observed that the cumulative relative PnL of M1 increases approximately from 0.6 to 1.5 as l_0 increases from 250 to 1000 through 500, indicating that investments with more initial funds are more profitable. As l_0 increases, the gap between M1 and the baselines becomes smaller. Note that a significant difference between M1 and the baselines is that M1 can adaptively control the hold period, whereas baselines cannot. The smaller gap between the relative PnL obtained by M1 and those obtained by baselines with larger l_0 indicates that it is less important for LPs to possess the knowledge of the optimal time to reallocate the liquidity position when LPs have a large amount of capital and gas fees are relatively cheap. This is a natural outcome since an LP can frequently reallocate the liquidity position to ensure the contract price remains within the range of their liquidity interval after each swap when the LP has infinite initial funds, rendering gas fees negligible. Conversely, when an LP’s initial funds are limited (i.e., when gas fees are relatively expensive), the performance gap between baselines and M1 becomes larger. In this case, deciding to adjust or hold the liquidity position becomes more crucial for LPs.

Table 5 presents the relative PnL of M1 in two scenarios: when it is combined with hedging and when it is not. This result highlights a significant benefit of adopting hedging and training the Dueling DDQN agent with the corresponding reward function in the sense that the relative PnL is less influenced by the trend of the contract price. Notably, the contract price exhibits a general downward trend in Period 2 and Period 4, while showing an upward trend in Period 1 and Period 3, as depicted in Figure 3. This trend aligns with the relative PnL obtained by M1†, which is negative in

Table 5: Relative PnL of M1 with hedging and without hedging

		ETH/USDC-0.3%		ETH/USDT-0.3%	
		M1	M1 \dagger	M1	M1 \dagger
$l_0 = 250$	period 1	0.223	-0.259	0.186	-0.259
	period 2	0.183	0.034	0.151	0.079
	period 3	0.130	-0.131	0.139	-0.141
	period 4	0.110	0.062	0.074	0.029
$l_0 = 500$	period 1	0.305	-0.291	0.275	-0.270
	period 2	0.273	0.043	0.267	0.020
	period 3	0.302	-0.178	0.195	-0.125
	period 4	0.159	0.023	0.143	0.080
$l_0 = 1000$	period 1	0.582	-0.283	0.525	-0.278
	period 2	0.388	0.061	0.383	0.050
	period 3	0.400	-0.119	0.354	-0.105
	period 4	0.222	0.079	0.189	0.071

M1 \dagger means Dueling DDQN is trained with the reward function defined as PnL when hedging is not used as in section 4.2.3.

Table 6: Detailed test performance in terms of trading fee, gas fee, and LVR in ETH/USDC-0.3% pool and ETH/USDT-0.3% pool with $l_0 = 250$ (highest trading fee, lowest gas fee, lowest LVR, and highest PnL are in bold.)

		ETH/USDC-0.3%				ETH/USDT-0.3%			
		M1	M2	M3	M4	M1	M2	M3	M4
Period 1	relative trading fee	0.691	0.63	0.774	0.088	0.605	0.614	0.921	0.402
	relative gas fee	0.113	0.207	0.293	0.120	0.096	0.207	0.293	0.133
	relative LVR	0.205	0.205	0.248	0.030	0.199	0.211	0.316	0.139
	relative PnL	0.373	0.218	0.232	-0.062	0.310	0.196	0.311	0.130
Period 2	relative trading fee	0.607	0.453	0.611	0.052	0.503	0.459	0.587	0.452
	relative gas fee	0.118	0.153	0.253	0.120	0.089	0.147	0.253	0.140
	relative LVR	0.182	0.132	0.168	0.015	0.161	0.159	0.156	0.125
	relative PnL	0.307	0.168	0.190	-0.083	0.252	0.154	0.178	0.187
Period 3	relative trading fee	0.541	0.569	0.550	0.089	0.686	0.391	0.777	0.295
	relative gas fee	0.104	0.167	0.280	0.120	0.119	0.060	0.280	0.153
	relative LVR	0.220	0.257	0.174	0.045	0.335	0.209	0.313	0.103
	relative PnL	0.217	0.145	0.096	-0.076	0.232	0.122	0.184	0.039
Period 4	relative trading fee	0.370	0.318	0.365	0.031	0.463	0.348	0.373	0.149
	relative gas fee	0.064	0.100	0.253	0.120	0.068	0.107	0.293	0.120
	relative LVR	0.121	0.108	0.107	0.014	0.271	0.159	0.115	0.059
	relative PnL	0.185	0.109	0.004	-0.104	0.124	0.082	-0.035	-0.030

Period 2 and Period 4, but positive in Period 1 and Period 3. However, the relative PnL gained by M1 is not correlated with the direction of the price movement, demonstrating the benefit derived from hedging.

Table 6 displays the relative trading fee, relative gas fee, and relative LVR in both pools with $l_0 = 250$. These three relative terms are obtained by dividing the actual values by the initial fund l_0 . Among the four methods tested herein, M1 learns the most profitable strategy. However, upon closer examination of the trading fee, gas fee, and LVR, it becomes evident that M1 does not consistently obtain the highest trading fee or incur the lowest gas fee and LVR. Instead, M1 finds the best trade-off between the reward of providing liquidity and its associated costs—namely, the sum of the gas fee and LVR.

An intriguing observation is that whenever M1 reallocates the liquidity position with initial funds of $l_0 = 250, 500$, and 1000 , it consistently favors the narrowest price interval with the smallest width, as depicted in Figure 5. In other words, despite the action space being defined as $\mathcal{A} = \{0, 1, \dots, N\}$, M1 consistently opts for either $a_t = 0$ or $a_t = 1$. Nevertheless, the difference in M1’s behavior across different initial fund values lies in the frequency of liquidity position reallocation—it occurs less frequently with lower initial funds.

6 Conclusion

This study introduces an adaptive uniform interval reallocation strategy for providing liquidity in Uniswap V3 by integrating hedging and deep reinforcement learning. The counterbalancing effect of the rebalancing strategy can mitigate risks for LPs in Uniswap V3, though leaving behind a predictable residue in terms of LVR. Deep reinforcement learning is employed to address two practical challenges for LPs: determining the optimal timing to adjust the liquidity position and establishing the appropriate width for the liquidity interval. Simulations based on historical data from the ETH/USDC 0.3% pool and ETH/USDT 0.3% pool empirically validate the viability of applying reinforcement learning methods to optimize liquidity provision in Uniswap V3. The Dueling DDQN model yields profits that are 9% to 69% higher than those of the strongest baseline across different pools with different initial funds.

References

- [1] Álvaro Cartea, Fayçal Drissi, and Marcello Monga. Decentralised finance and automated market making: Predictable loss and optimal liquidity provision. *Available at SSRN 4273989*, 2022.
- [2] Michael Neuder, Rithvik Rao, Daniel J Moroz, and David C Parkes. Strategic liquidity provision in uniswap v3. *arXiv preprint arXiv:2106.12033*, 2021.
- [3] Yoge Bar-On and Yishay Mansour. Uniswap liquidity provision: An online learning approach. *arXiv preprint arXiv:2302.00610*, 2023.
- [4] Robin Fritsch. Concentrated liquidity in automated market makers. In *Proceedings of the 2021 ACM CCS Workshop on Decentralized Finance and Security*, pages 15–20, 2021.
- [5] Zhou Fan, Francisco J Marmolejo-Cossío, Ben Altschuler, He Sun, Xintong Wang, and David Parkes. Differential liquidity provision in uniswap v3 and implications for contract design. In *Proceedings of the Third ACM International Conference on AI in Finance*, pages 9–17, 2022.
- [6] Lioba Heimbach, Eric Schertenleib, and Roger Wattenhofer. Risks and returns of uniswap v3 liquidity providers. *arXiv preprint arXiv:2205.08904*, 2022.
- [7] Jason Milionis, Ciamac C Moallemi, Tim Roughgarden, and Anthony Lee Zhang. Automated market making and loss-versus-rebalancing. *arXiv preprint arXiv:2208.06046*, 2022.
- [8] Ciamac C Moallemi and Muye Wang. A reinforcement learning approach to optimal execution. *Quantitative Finance*, 22(6):1051–1069, 2022.
- [9] Hyungjun Park, Min Kyu Sim, and Dong Gu Choi. An intelligent financial portfolio trading strategy using deep q-learning. *Expert Systems with Applications*, 158:113573, 2020.
- [10] Xiao-Yang Liu, Hongyang Yang, Jiechao Gao, and Christina Dan Wang. Finrl: Deep reinforcement learning framework to automate trading in quantitative finance. In *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–9, 2021.
- [11] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654, 1973.
- [12] Robert C Merton. Theory of rational option pricing. *The Bell Journal of economics and management science*, pages 141–183, 1973.
- [13] Petter N Kolm and Gordon Ritter. Dynamic replication and hedging: A reinforcement learning approach. *The Journal of Financial Data Science*, 1(1):159–171, 2019.
- [14] Marc Sabate-Vidales and David Siska. The case for variable fees in constant product markets: An agent based simulation. *Available at SSRN*, 2022.
- [15] Tristan Lim. Predictive crypto-asset automated market making architecture for decentralized finance using deep reinforcement learning. *arXiv preprint arXiv:2211.01346*, 2022.
- [16] Atis Elsts. Liquidity math in uniswap v3. 2021.
- [17] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.
- [18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

- [20] Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.
- [21] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

A Implementation Details of The Proposed Method and Baselines

Algorithm 1 Exponential Weights Adaptive Strategy (EWA)

```

1: Input  $N \in \mathbb{N}_+$ ,  $\eta > 0$ ,  $T_{re} \in \mathbb{N}_+$ 
2: Initialize  $r_0(n) \leftarrow 0$  for all  $n \in [N]$ 
3: for  $1 \leq t \leq T$  do
4:   if  $\text{mod}(t, T_{re}) == 0$  then
5:      $p_t(n) \leftarrow \frac{\exp(\eta \sum_{0 \leq \tau \leq t} r_\tau(n))}{\sum_{1 \leq \mu \leq N} \exp(\eta \sum_{0 \leq \tau \leq t} r_\tau(\mu))}$ 
6:     invest  $p_t(n)$  proportion of funds into a liquidity position with width  $n$ , and get reward  $r_t(n)$ 
7:   end if
8: end for

```

Hyperparameters of EWA are chosen by maximizing its performance on validation data. Hyperparameters of uniform τ -reset strategy are determined by using testing set. In reality, testing data is unknown, so this setting gives uniform τ -reset strategy an advantage, thus this baseline is stronger than in practice. Hyperparameters of Dueling DDQN are provided in Table 9, respectively. Hyperparameters of Dueling DDQN are fixed for all four periods. In this paper's experiment, hyperparameters of Dueling DDQN are tuned roughly, it is believed that further fine tuning is likely to improve their performance.

Table 7: hyperparameters of EWA

		$l_0 = 250$			$l_0 = 500$			$l_0 = 1000$		
		N	η	T_{re}	N	η	T_{re}	N	η	T_{re}
ETH-USDC 0.3%	Period 1	10	1	21	10	1	14	10	1	6
	Period 2	10	10	24	10	10	24	10	10	9
	Period 3	10	1	22	10	4	15	10	1	13
	Period 4	10	7	24	10	1	21	10	1	18
ETH-USDC 0.3%	Period 1	10	1	21	10	1	6	10	1	6
	Period 2	10	10	24	10	10	24	10	10	12
	Period 3	10	1	22	10	7	22	10	10	3
	Period 4	10	7	21	10	1	21	10	1	21

Table 8: hyperparameters of uniform τ -reset strategy

		$l_0 = 250$			$l_0 = 500$			$l_0 = 1000$		
		τ	τ	τ	τ	τ	τ	τ	τ	τ
ETH-USDC 0.3%	Period 1	6		4		1				
	Period 2	5		2		1				
	Period 3	6		3		2				
	Period 4	4		3		1				
ETH-USDT 0.3%	Period 1	6		4		1				
	Period 2	5		2		1				
	Period 3	10		3		1				
	Period 4	4		3		1				

Table 9: hyperparameters of Dueling DDQN

hyperemeters	values
hidden unites	[64, 64]
activation	ReLU
final activation	None
learning rate	10^{-4}
batch size	256
buffer size	10^6
discounted factor	0.9
target update rate	0.01
gradient clipping norm	0.7