Strategic Liquidity Provision in Uniswap V3

Zhou Fan ✉
Harvard University, Cambridge, MA, USA

Francisco Marmolejo-Cossio ✉
Harvard University, Cambridge, MA, USA
IOG, USA

Daniel Moroz ✉
Harvard University, Cambridge, MA, USA

Michael Neuder ✉
Harvard University, Cambridge, MA, USA

Rithvik Rao ✉
Harvard University, Cambridge, MA, USA

David C. Parkes ✉
Harvard University, Cambridge, MA, USA

Abstract

Uniswap v3 is the largest decentralized exchange for digital currencies. A novelty of its design is that it allows a liquidity provider (LP) to allocate liquidity to one or more closed intervals of the price of an asset instead of the full range of possible prices. An LP earns fee rewards proportional to the amount of its liquidity allocation when prices move in this interval. This induces the problem of *strategic liquidity provision*: smaller intervals result in higher concentration of liquidity and correspondingly larger fees when the price remains in the interval, but with higher risk as prices may exit the interval leaving the LP with no fee rewards. Although reallocating liquidity to new intervals can mitigate this loss, it comes at a cost, as LPs must expend gas fees to do so. We formalize the dynamic liquidity provision problem and focus on a general class of strategies for which we provide a neural network-based optimization framework for maximizing LP earnings. We model a single LP that faces an exogenous sequence of price changes that arise from arbitrage and non-arbitrage trades in the decentralized exchange. We present experimental results informed by historical price data that demonstrate large improvements in LP earnings over existing allocation strategy baselines. Moreover we provide insight into qualitative differences in optimal LP behaviour in different economic environments.

2012 ACM Subject Classification Computing methodologies → Modeling methodologies; Computing methodologies → Neural networks

Keywords and phrases blockchain, decentralized finance, Uniswap v3, liquidity provision, stochastic gradient descent

Digital Object Identifier 10.4230/LIPIcs.AFT.2023.25

Related Version *Full Version*: `https://arxiv.org/abs/2106.12033`

Funding This work is supported in part by two generous gifts to the Center for Research on Computation and Society at Harvard University, both to support research on applied cryptography and society.

1 Introduction

Decentralized finance (DeFi) is a large and rapidly growing collection of projects in the cryptocurrency and blockchain ecosystem. From May 2019 to May 2023, the TVL (*total value locked*, meaning the sum of all liquidity provided to the protocol) into DeFi protocols

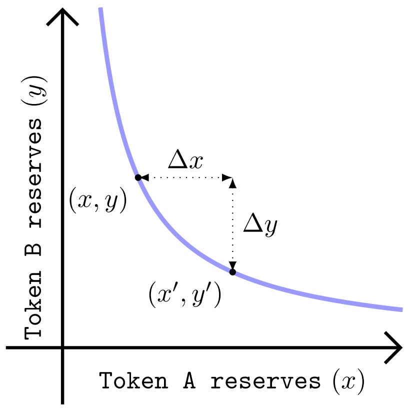


Figure 1 The reserve curve for Uniswap v2. If the pool has reserves (x, y) where x and y represent units of token A and B respectively, then the contract price of token A is $P = y/x$. A trader can send Δx units of token A to receive Δy units of token B, such that $x'y' = L^2$, where $x' = x + \Delta x$ and $y' = y - \Delta y$. The contract price of token A after the trade is $P' = y'/x'$.

has increased 100x from 500 million USD to 50 billion USD. During this time period, TVL rapidly increased in 2021, reaching a peak of roughly 176 billion USD in November 2021, but it also suffered large drops in 2022 leading to current TVL levels.[1]

DeFi aims to provide the function of financial intermediaries and instruments through smart contracts executed on blockchains (typically Ethereum). The importance of decentralized exchanges (DEXes) is that traders can swap tokens of different types without a trusted intermediary. Most DEXes, including Uniswap, fall into the category of *constant function market makers* (CFMMs). Instead of using an order book as is done in traditional exchanges, CFMMs are smart contracts that use an *automated market maker* (AMM) to determine the price of a trade.

In *Uniswap v2*, token pairs can be swapped for each other using *liquidity pools*, which contain quantities of each of the pair of tokens (say, token A and token B). Permitted trades are determined by the *reserve curve*, $xy = L^2$, where x and y denote the the number of tokens of type A and B respectively in the liquidity pool, and the value of L must be maintained across a trade. *Liquidity providers (LPs)* add tokens to liquidity pools for the traders to swap against, and are rewarded through the fees traders pay. An illustrative reserve curve is shown in Figure 1. Assuming that the v2 contract holds x units of token A and y units of token B with $xy = L^2$, then in order to sell some quantity $\Delta x > 0$ of token A for some quantity $\Delta y > 0$ of token B, the trader must keep the product of reserves constant, with Δy such that $(x - \Delta x)(y + \Delta y) = L^2$. This defines an effective contract price for token A in units of token B, i.e., $P = -dy/dx$. In the context of the $xy = L^2$ curve, we have $y = L^2/x \implies P = L^2/x^2 = y/x$. By convention, we take the contract price to be the price of token A, which we may assume is volatile relative to token B. In Uniswap v2, traders pay LPs a fee of 0.3% of the transaction amount in return for using the liquidity to execute a swap [2]. In v2, the liquidity of every LP can be used for swaps at every possible price $P \in (0, \infty)$, and an LP earns fees based on their fraction of the total liquidity in the pool.

Uniswap v3 launched on Ethereum on May 3, 2021 and introduced *concentrated liquidity* [3], where an LP can provide liquidity to each of any number of price intervals, these called *positions*. In particular, liquidity allocated by an LP to position $[P_a, P_b]$ earns fees when the

[1] `https://defillama.com/`

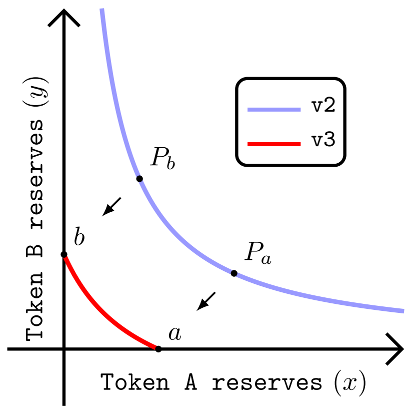


Figure 2 The reserve curve of Uniswap v2 over all prices, and of Uniswap v3 over the price interval $[P_a, P_b]$. When trades give rise to contract prices in this interval, LP assets are swapped according to the v3 curve, which is an affine transformation of the v2 curve and defined to respect the price limits of interval $[P_a, P_b]$.

contract price is in that interval. If multiple LPs allocate liquidity over an interval containing the current price, each is rewarded proportionally to the fraction of the liquidity they own at that price. Figure 2 visualizes the functional invariant respected by the overall assets provided by LPs to support trades over $[P_a, P_b]$. For trades in this interval, a v3 contract effectively shifts the reserve curve of Uniswap v2 via an affine transformation to intercept the axes at a and b, which depend on the end points of interval $[P_a, P_b]$. This shifted curve is governed by the equation: $\left(x + L/\sqrt{P_b}\right)\left(y + L\sqrt{P_a}\right) = L^2$, and the intercepts, a and b, can be calculated by letting x or y equal zero respectively [3] . This description of Uniswap v3 is inherently local, as it describes trade dynamics for a specific price interval. Gluing the local dynamics together for all prices gives rise to an *aggregate reserve curve* that governs arbitrary trades across all possible prices. This global reserve curve is in turn a function of the aggregate distribution of liquidity provided by all LPs. From the perspective of traders, when more liquidity is allocated to a given price interval, there is less trade slippage for prices in that interval. This reduced slippage corresponds to a flatter section of the aggregate reserve curve, as visualized in Figure 3.

Uniswap v3 supports a diversity of LP strategies in regard to the allocation of liquidity, as an LP can mint multiple positions, each on a different interval. Each LP is presented with a tradeoff between choosing large positions that cover many possible prices but earn less fees and smaller more concentrated positions that are more risky (since they cover fewer prices). Additionally, an LP can reallocate its liquidity as prices change, but this comes at two costs. First, an LP may need to trade between assets to mint new liquidity positions in a reallocation, potentially suffering losses from slippage in such trades. Second, since liquidity allocations are transactions that must be written as updates to the contract and included in a block, they also incur gas fees. Both of these costs must be incorporated in more complex LP strategies that make use of liquidity reallocation. Given the increased complexity in LP actions from v2 to v3, it is important to understand potential ways LPs can benefit from strategically allocating liquidity as prices change, as this ultimately impacts the performance of v3 contracts as DEXs.

In this regard, much relevant work has studied ways in which LPs in Uniswap v3 can optimize their earnings when faced with uncertain beliefs over how trades will evolve over time. Most relevant to our work are two papers, [11, 14], the first of which provides insight

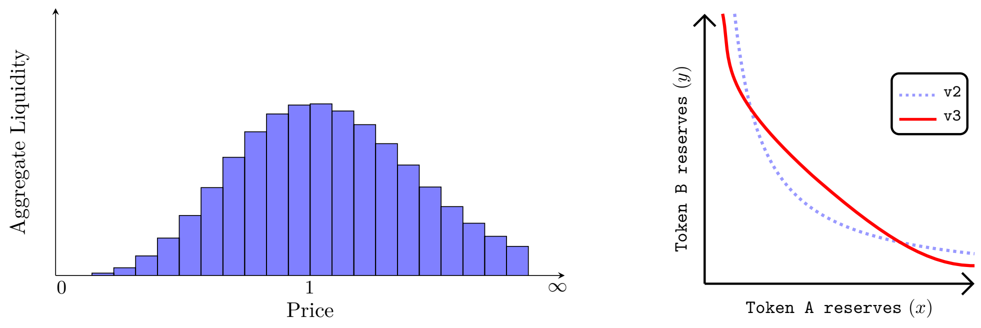


Figure 3 An aggregate distribution of liquidity for a Uniswap v3 contract (left plot), with most liquidity allocated close to unit price $P = 1$. This results in an aggregate reserve curve (right, red line), which is flatter than the corresponding v2 curve (dotted blue) at prices close to $P = 1$ and supports a larger volume of trades at these prices with less slippage.

into how LPs can profit from liquidity reallocations with simple positions, and the second of which focuses on how LPs can profit from complicated static liquidity positions over a given time horizon.[2] In more detail, the authors of [11] provide a closed form solution for computing optimal LP allocations that dynamically readjust positions to different intervals as prices evolve over time. Though these strategies make use of dynamic reallocation of liquidity, the only allocations explored in the optimization are individual v3 positions over a single price range (which forcibly change at each reallocation) instead of the full class of potential allocations available to an LP in Uniswap v3. The authors of [14] compute optimal arbitrary v3 positions for small LPs who seek to maximize profit and loss over a fixed time horizon, but only consider static LP strategies which do not make use of reallocations as prices change.

Our work addresses the gaps from these papers by exploring optimal liquidity provision strategies that simultaneously make use of liquidity reallocations and the full complexity of v3 positions. As in [14], we adopt the perspective of an LP with stochastic beliefs over how market prices evolve over a given time horizon, and how contract prices may change along with market prices via arbitrage and non-arbitrage trades. In addition, we also make the assumption that the LP is small enough that these beliefs are independent of capital allocated by the LP. We provide an optimization framework for computing optimal dynamic liquidity provision strategies over a given time horizon, and we show that such strategies can provide large gains to LPs over strategies that are static or strategies that make use of simple liquidity positions.

We define *dynamic liquidity provision strategies* as a means by which LPs can mitigate potential losses by using earned capital to reallocate liquidity to different price intervals in a given time horizon. In particular, we focus on the family of *τ-reset strategies*, which allocate over an interval of prices centered on the current price and whose width is controlled by τ (including the possibility of declining to allocate liquidity) and reallocate whenever the price moves outside this interval. Moreover, within the space of potential dynamic liquidity provision strategies, we distinguish between those that are context-aware and context-independent depending on whether they incorporate historical price and contract information in their decision-making. We develop methods of stochastic optimization for optimizing over τ-reset strategies when an LP has stochastic beliefs on market and contract

[2] Both of these papers were written after the first version of the present paper [21]

prices, and with different levels of risk-aversion. We give empirical results based on historical Ethereum price data to show that incorporating both of these aspects into LP allocation strategies gives rise to large gains in performance for LPs of various levels of risk-aversion, especially for context-aware reallocation strategies, which we optimize for with a neural network. In addition, our results provide insight into how optimal LP behaviour varies depending on relevant aspects of their economic environment. In particular, we find that more risk-averse LPs spread their liquidity over larger price ranges, especially when faced with a larger volume of non-arbitrage trades. In addition we also find that, as expected, optimal reset frequencies are sensitive to the reallocation costs.

1.1 Related work

The Uniswap v1 protocol was defined by [1], followed up with v2 by [2], and most recently amended in v3 by [3]. There has been a growing body of work studying LP incentives in Uniswap v2. [6] present an analysis of Uniswap, and more broadly of constant-product AMMs, and demonstrate conditions for which the markets closely track the price on an external reference market. [4] extend this line of research by demonstrating that the more general class of CFMMs incentivize participants to report the true price of an asset on an external market, demonstrating their value as price oracles.

[7] study the equilibrium liquidity provision of constant product AMMs, and show that strategic LPs in the Uniswap v2 environment may have a non-monotonic best response when parameterized with the opponents' liquidity provision. [5] extend this line of work to CFMMs and calculate bounds on the LP rewards based on the curve that defines the CFMM. [10] studies the adopation of decentralized exchanges more generally, using a sequential game framework to model interactions between LPs and traders. In addition, [23] and [15] provide an axiomatic framework for general CFMMs similar in nature to Uniswap v2, with the latter focusing on connections with CFMMs in the prediction market setting. [12] consider *geometric mean market makers* (G3Ms), and show that passive liquidity provision can be used to replicate payoffs of financial derivatives and more active trading strategies. [24] and [25] analyze the growth in wealth of an LP in CFMM for a geometric Brownian motion price process. [13] extend this to more general LP objectives and diffusion processes.

All above work applies to Uniswap v2 and similarly structured CFMMs but not to v3. An early blog post by [18] describes a "passive rebalancing" strategy for v3, which aims at maintaining a 50-50 ratio of value for the assets of the LP. In addition to [14] and [11], further related work on v3 includes [20], who decompose divergence/impermanent loss into hedgeable market risk and profit made by arbitrage traders at a loss to the exchange. (This is related to [11], who decompose divergence/impermanent loss into two components: the loss due to arbitrage (convexity cost) and the cost of locking capital). [9] uses regret-minimization from online learning to provide liquidity provision strategies under adversarial trading. [19] and [16] study the construction of optimal CFMMs from the perspective of LP beliefs, with the latter providing a Myersonian framework for creating incentive compatible AMMs, and the former employing techniques from convex optimization to determine optimal trading functions based on LP beliefs on future trades. [17] study the risks inherent to LP returns for multiple fixed strategies in different economic environments, concluding that liquidity provision in v3 is a sophisticated game where uninformed retail traders can suffer large disadvantages relative to more informed agents. In addition to studying optimal static liquidity provision, [14] provide insights on how aspects of a v3 contract, notably the partition of price space, have implications on LP profit as well as gas fees incurred by traders.

1.2 Outline

Section 2 introduces the Uniswap v3 protocol. Section 3 formalizes the earnings to an LP from a dynamic liquidity provision strategy and introduces the family of τ-reset strategies. Section 4 introduces the computational methods for optimizing over τ-reset strategies and specifically defines the context-aware/independent dynamic liquidity strategies we empirically study as well as simple baselines to which we compare performance. Section 5 provides details regarding the economic environments we modulate to study optimal LP behaviour, and Section 6 presents empirical results. Section 7 gives open problems for future research and concludes.

2 The Mechanics of Uniswap

In this section, we provide a brief overview of Uniswap v3 contracts. In all that follows, we consider a v3 contract that enables trades between two types of tokens that we designate token A and token B. Furthermore, without loss of generality, we assume token B is the numeraire, hence when we refer to the price in the contract, we refer to the price of a unit of A in terms of B. As mentioned in the introduction, liquidity providers (LPs) provide bundles of A and B tokens to the contract as liquidity to be traded against. The following sections largely follow the mathematical formalism of [14].

2.1 v3 Contracts

Partitioned Price Space

A Uniswap contract maintains the *contract price* of token A given by $P \in (0, \infty)$. This is the infinitesimal price that traders obtain for trading with the contract. In Uniswap v2 contracts, the liquidity that LPs provide is used to support every trade, whatever the trade price. Uniswap v3 contracts provide a richer set of actions for an LP, where they can specify a price range where liquidity is to be used for trading. In order to enable this functionality, a v3 contract partitions token A prices into a finite set of *price buckets*: $\boldsymbol{\mu} = \{B_{-m}, \dots, B_0, \dots, B_n\}$ with buckets $B_i = [a_i, b_i]$. We also require $a_0 < 1 < b_0$, so that the parity price lies in the 0-th bucket, and that $b_i = a_{i+1}$ for $i \in \{-m, \dots, n-1\}$.

Minting and Burning Liquidity

LPs provide (or *mint*) liquidity in a particular bucket, $B_i = [a_i, b_i]$, referred to as *B_i-liquidity*, by sending a bundle of A and B tokens to the contract. The token bundle required to mint L units of B_i-liquidity is given by the *liquidity value function* [14], and is tuple $\mathcal{V}^{(3)}(L, P, B_i)$, where the first component is the quantity in token A and the second is the quantity in token B. For $a < b$, let $\Delta^x_{b,a} = \frac{1}{\sqrt{a}} - \frac{1}{\sqrt{b}}$ and $\Delta^y_{a,b} = \sqrt{b} - \sqrt{a}$.

▶ **Definition 1** (B_i-Liquidity Value [14]). *For contract price P, bucket $B_i = [a_i, b_i]$, and number of units $L > 0$, the* bundle liquidity value $\mathcal{V}^{(3)}(L, P, B_i) \in \mathbb{R}^+ \times \mathbb{R}^+$ *is defined as*

$$\mathcal{V}^{(3)}(L, P, B_i) = \begin{cases} (L\Delta^x_{b_i,a_i}, 0) & \text{if } P < a_i, \\ (0, L\Delta^y_{a_i,b_i}) & \text{if } P > b_i, \text{ or} \\ (L\Delta^x_{b_i,P}, L\Delta^y_{a_i,P}) & \text{if } P \in B_i, \end{cases} \tag{1}$$

and specifies the bundle of A and B tokens, respectively, to mint L units of B_i-liquidity.

LPs can also remove liquidity they have a claim to from the contract by means of the value function. When this happens, we say that an LP *burns* L units of B_i liquidity, and the LP receives token bundle $\mathcal{V}^{(3)}(L, P, B_i)$ if the contract price is P. Since $\mathcal{V}^{(3)}$ is linear in L, we also adopt shorthand $\mathcal{V}^{(3)}(P, B_i) = \mathcal{V}^{(3)}(1, P, B_i)$ for the token bundle value of a single unit of liquidity, with $\mathcal{V}^{(3)}(L, P, B_i) = L \cdot \mathcal{V}^{(3)}(P, B_i)$.

Trading and Fees

When the contract price is in a given bucket, the trade dynamics respect the shifted reserve curve of Figure 2. Fees are governed by a fee rate, $\gamma \in (0, 1)$, such that if a given trader sends Δx units of token A to the contract, $\gamma \Delta x$ is first skimmed as fees to be shared proportionally amongst LPs who have allocated liquidity to the bucket containing the current price. The remaining $(1 - \gamma)\Delta x$ units of token A are used for trading via the v3 reserve curve. We refer the reader to the full version of the paper for a detailed discussion on trading and fees in Uniswap v3.

3 Liquidity Allocation Strategies and LP Earnings

In this section, we describe a rich set of strategies that LPs can use to maximize their earnings over a given time horizon. As token B is the numeraire, we measure all earnings in terms of units of token B and we assume that the LP begins with a fixed budget consisting of $W > 0$ units of token B. We model price discovery between A and B tokens as occurring outside of Uniswap contracts, and in addition to the contract price there is a *market price* that is determined by external markets. We denote the market price by P_m and contract price by P_c, and we extend price P so that $P = (P_m, P_c)$ denotes a *contract-market price pair*. Furthermore, we assume that arbitrage trade can be performed by traders at price P_m which in turn brings P_c close to P_m (see Section 5.1). In what follows, we consider time horizons that are characterized by a single sequence of $T > 0$ contract-market prices, denoted by $\mathbf{P} = (P_0, \dots, P_T)$, where $P_t = (P_{c,t}, P_{m,t})$ is the t-th contract-market price in the sequence (time steps are indexed t).

3.1 Static Liquidity Provision Strategies

We begin by using a similar mathematical formalism and notation from [14] to express an LP's earnings over price sequence $\mathbf{P}$ for a simple family of liquidity allocation strategies.

▶ **Definition 2** (Static liquidity provision strategy). *An LP with an initial budget of $W > 0$ units of token B uses a* static liquidity provision strategy *when they*

1. *mint an initial liquidity allocation at P_0,*
2. *accrue token fees over the course of $\mathbf{P}$ as prices change, and*
3. *burn the existing liquidity allocation at P_T, the end of the time horizon, to recover invested capital from the contract.*

We focus on a single LP, and suppose they mint liquidity positions at the beginning of $\mathbf{P}$, when contract-market prices are given by $P_0 = (P_{c,0}, P_{m,0})$, with their initial budget of $W > 0$ units of token B. For this, let $\mathbf{x} = (x_{-m}, \dots, x_n)$ denote a *proportional liquidity allocation*, where for $i \in \{-m, \dots, n\}$, $x_i \geq 0$ represents the proportion of capital used to mint B_i-liquidity (with $\sum_{i=-m}^{n} x_i \leq 1$ so that $\mathbf{x} \in \Delta^{m+n+2}$, the $(m + n + 2)$-dimensional simplex). The LP uses $W x_i$ units of token B to mint B_i-liquidity for each of $i \in \{-m, \dots, n\}$. Let $x_{n+1} = 1 - \sum_{i=-m}^{n} x_i \in [0, 1]$ denote the proportion of capital that the LP does not invest and keeps as units of token B.

Let $\mathcal{B} : (\mathbb{R}^+)^2 \times \mathbb{R}^+ \to \mathbb{R}^+$, defined as $\mathcal{B}((z_1, z_2), P_m) = P_m \cdot z_1 + z_2$, return the *token B market worth* of a bundle of A and B tokens when token A has market price P_m. For a given contract-market price sequence, $\mathbf{P}$, let $w_i = \mathcal{B}(\mathcal{V}^{(3)}(P_{c,0}, B_i), P_{m,0})$ denote the amount of B tokens required to mint one unit of B_i-liquidity at the initial contract-market price of $P_0 = (P_{c,0}, P_{m,0})$. With this in hand, let vector $\boldsymbol{\ell} = (\ell_{-m}, \dots, \ell_n)$ denote the *absolute liquidity allocation* induced by initial budget (W), proportional liquidity allocation ($\mathbf{x}$) and initial contract-market price (P_0). It follows that $\ell_i = \frac{Wx_i}{w_i}$ units of B_i-liquidity for each $i \in \{-m, \dots, n\}$. This implies that $\boldsymbol{\ell}$ is linear as a function of each of $\mathbf{x}$ and W.

3.1.1 Linearity of Fee Rewards in $\mathbf{x}$

We are ultimately interested in expressing an LP's token B value of earnings as a function of their liquidity allocation over the contract-market price sequence. We begin by determining the amount of trading fees earned by an LP. Interestingly, these earnings are not only independent of other LP allocations, but also linear in $\boldsymbol{\ell}$ (and consequently $\mathbf{x}$).

▶ **Theorem 3** (Section 3.1 [26]). *For a fixed contract-market price sequence $\mathbf{P}$, the amount of A tokens and B tokens accrued from fees is linear in $\boldsymbol{\ell}$ and independent of the liquidity of other LPs in the contract.*

That the fees that a single LP earns are independent of other LPs' liquidity allocations follows from the assumption that contract-market prices are independent of the liquidity allocation of this LP. Indeed, for a fixed price sequence, allocating liquidity by an LP has two effects. First, increasing the liquidity means that a larger volume of trade needs to happen to effect the same price change, resulting in more fees to be paid out to LPs. Second, the same LP has a proportionally larger amount of liquidity across the relevant price interval. The net effect is that fees are only a function of a single LP's proportional (or absolute) liquidity allocation. Theorem 3 justifies the following definition of a trading fee function for a single LP and a given contract-market price sequence.

▶ **Definition 4** (Trading Fee Functions). *Suppose that $\mathbf{P}$ is a fixed contract-market price sequence and $W > 0$ an initial token B budget. For a proportional (or absolute) liquidity allocation given by $\mathbf{x}$ (or $\boldsymbol{\ell}$), we let $F^A(\mathbf{x}, W, \mathbf{P})$ (or $F^A(\boldsymbol{\ell}, \mathbf{P})$) denote the units of A tokens earned from fees over $\mathbf{P}$ from downward price movements. Similarly, we let $F^B(\mathbf{x}, W, \mathbf{P})$ (or $F^B(\boldsymbol{\ell}, \mathbf{P})$) denote the units of B tokens earned from fees over $\mathbf{P}$ from upward price movements.*

When the resulting absolute liquidity allocation $\boldsymbol{\ell}$ is treated as a function of $\mathbf{x}$ and W, it is linear in $\mathbf{x}$ and W, and it follows that both $F^A(\mathbf{x}, W, \mathbf{P})$ and $F^B(\mathbf{x}, W, \mathbf{P})$ are linear in $\mathbf{x}$ and W. For this reason, we let $F^A(\mathbf{x}, \mathbf{P}) = F^A(\mathbf{x}, 1, \mathbf{P})$ and $F^B(\mathbf{x}, \mathbf{P}) = F^B(\mathbf{x}, 1, \mathbf{P})$. This in turn implies that $F^A(\mathbf{x}, W, \mathbf{P}) = W \cdot F^A(\mathbf{x}, \mathbf{P})$, and similarly $F^B(\mathbf{x}, W, \mathbf{P}) = W \cdot F^B(\mathbf{x}, \mathbf{P})$ for arbitrary $\mathbf{x}, W$, and $\mathbf{P}$.

3.1.2 Burning Liquidity Allocations at P_T

All that remains to fully quantify the earnings of an LP over $\mathbf{P}$ is to take into account the capital they obtain by burning their liquidity positions at time T under contract-market price $P_T = (P_{c,T}, P_{T,m})$, obtaining a final quantity of token B. For this, let $w'_i = \mathcal{B}(\mathcal{V}^{(3)}(P_{c,T}, B_i), P_{m,T})$ be the token B worth of the capital obtained from burning 1 unit of B_i-liquidity at the final contract-market price of $P_T = (P_{c,T}, P_{m,T})$. Given absolute liquidity position $\boldsymbol{\ell}$, the overall token B value of capital obtained from burning is $C(\boldsymbol{\ell}, \mathbf{P}) =$

$\sum_{i=-m}^{n} w'_i \ell_i$, and linear in $\boldsymbol{\ell}$. Let $C(\mathbf{x}, W, \mathbf{P})$ denote the final token B worth (at P_T) of a liquidity position minted at P_0 with $\mathbf{x}$ and budget W. Since proportional allocations also allow an LP to maintain funds in terms of token B (i.e., when $x_{n+1} > 0$), we obtain the expression $C(\mathbf{x}, W, \mathbf{P}) = C(\boldsymbol{\ell}, \mathbf{P}) + W x_{n+1}$, where $\boldsymbol{\ell}$ is the absolute liquidity allocation corresponding to $\mathbf{x}$. Once more, since $\boldsymbol{\ell}$ is in turn linear in $\mathbf{x}$ and W, it follows that C is linear in $\boldsymbol{\ell}$ and W. For this reason, we let $C(\mathbf{x}, \mathbf{P}) = C(\mathbf{x}, 1, \mathbf{P})$, so that $C(\mathbf{x}, W, \mathbf{P}) = W \cdot C(\mathbf{x}, \mathbf{P})$ for arbitrary $\mathbf{x}, W$, and $\mathbf{P}$.

3.1.3 Linearity of Overall Earnings in $\mathbf{x}$

We now define an LP's earnings over a contract-market price sequence.

▶ **Definition 5.** *Suppose that $\mathbf{P} = (P_0, \ldots, P_T)$ is a contract-market price sequence and that $\mathbf{x} \in \Delta^{m+n+2}$ is a proportional liquidity allocation. An LP's earnings (in units of token B) under $\mathbf{P}$ with an initial budget of $W > 0$ is,*

$$V(\mathbf{x}, W, \mathbf{P}) = P_{m,T} \cdot F^A(\mathbf{x}, W, \mathbf{P}) + F^B(\mathbf{x}, W, \mathbf{P}) + C(\mathbf{x}, W, \mathbf{P}).$$

From this definition, we conclude that an LP's earnings from a fixed contract-market price sequence and with a static liquidity provision strategy are linear in $\mathbf{x}$ and W.[3]

▶ **Theorem 6.** *V is linear in both W and $\mathbf{x}$ for any contract-market price sequence, $\mathbf{P}$.*

Proof. This is an immediate corollary of the fact that F^A, F^B and C are each linear in $\mathbf{x}$ and W for any contract-market price sequence $\mathbf{P}$. ◀

Similar to before, we use the shorthand $V(\mathbf{x}, \mathbf{P}) = V(\mathbf{x}, 1, \mathbf{P})$ such that $V(\mathbf{x}, W, \mathbf{P}) = W \cdot V(\mathbf{x}, \mathbf{P})$ for arbitrary $\mathbf{x}, W$, and $\mathbf{P}$.

3.2 Dynamic Liquidity Provision Strategies

In this section, we introduce the notion of *dynamic liquidity provision* strategies, where an LP can reallocate their liquidity at any time step of the contract-market price sequence, $\mathbf{P}$.

At a high level, if an LP chooses to reallocate liquidity at time t, they burn their existing liquidity position and use their overall earnings at time t, denoted by W_t, to mint a new proportional allocation, $\mathbf{x}$, given the contract-market price P_t. Reallocation comes at a cost however, which represents the fact that burning and minting positions on a Uniswap contract requires paying gas fees, and that minting new positions may require the LP to trade between A and B tokens. We model reallocation costs as proportional to overall earnings used to mint the position $\mathbf{x}$ (i.e. the funds corresponding to capital kept token B outside the contract (x_{n+1}) do not incur a cost). Cost is specified by a single parameter, $\eta \in [0, 1]$, such that the LP retains $\eta W_t(1 - x_{n+1})$ of the funds they intend to use for minting a new position after paying reallocation costs at time t (i.e., by paying $(1 - \eta)W_t(1 - x_{n+1})$ in reallocation cost).

We partition price sequence $\mathbf{P}$ into *epochs*, which are sequences of contract-market prices from $\mathbf{P}$ uninterrupted by liquidity reallocations. For an LP that burns and reallocates liquidity positions at time steps $\mathbf{t} = (t^0, \ldots t^k)$, where $t^0 = 0$ and $t^k = T$, there are $k \geq 1$ epochs, where the j-th epoch is $E^j = (P_{t^j}, \ldots, P_{t^{j+1}})$. When indexing over epochs we use superscripts, and when indexing over time-steps in $\mathbf{P}$ we use subscripts.

[3] As a side note, with the definition of LP's earnings here we are modeling the profit and loss (PnL) of an LP who holds their Uniswap v3 liquidity position without hedging. The strategy optimization approach in this paper can be naturally extended for maximizing the PnL of a delta-hedged LP as well by incorporating Loss versus Rebalancing (LVR) from [20] since LVR is also linear in $\mathbf{x}$ and W.

Each epoch, E^j, is associated with the total earnings, W^j, the LP has accrued at the beginning of the epoch, and the proportional liquidity allocation, $\mathbf{x}^j$, the LP uses to mint positions with W^j over E^j. From the static liquidity allocation analysis, it follows that the LP's earnings over the epoch are given by $W^j \cdot V(\mathbf{x}^j, E^j)$. After incorporating the proportional reallocation cost, the earnings available for the LP to use for E^{j+1} are $W^{j+1} = \eta W^j \cdot V(\mathbf{x}^j, E^j)(1 - x^{j+1}_{n+1})$. We encode the collection of all proportional allocations as a $(k \times (m+n+1))$ matrix $\mathbf{X}$, such that the j-th row of $\mathbf{X}$ corresponds to $\mathbf{x}^j$.

▶ **Definition 7** (Dynamic liquidity provision strategy). *We say that* Λ *is a* dynamic liquidity provision strategy *if it takes as input a contract-market price sequence,* $\mathbf{P} = (P_1, \ldots, P_T)$, *and defines:*

- $\mathbf{t} = (t^0, \ldots, t^k)$, *with* $k \geq 1$, $t^0 = 0$, *and* $t^k = T$. *These are time-steps where a reallocation occurs.*
- $\mathbf{X} \in mat\,(k \times (m+n+1))$ *such that the* j*-th row of* $\mathbf{X}$ *encodes* $\mathbf{x}^j$, *the proportional liquidity allocation profile to be used at* E^j.

We write $\Lambda(\mathbf{P}) = (\mathbf{t}, \mathbf{X})$ *and say that this is the* realized dynamic liquidity provision strategy *of an LP under* Λ *for contract-market price sequence* $\mathbf{P}$.

For an initial budget $W(= W_0 = W^0)$, we let $V(\Lambda, W, \mathbf{P})$ denote the overall earnings an LP obtains over $\mathbf{P}$ by employing strategy Λ, which can be computed recursively over the epochs of $\mathbf{P}$. As in previous sections, we let $V(\Lambda, \mathbf{P}) = V(\Lambda, 1, \mathbf{P})$.

3.2.1 Reset Liquidity Strategies

In practice, a strategy Λ may not be implementable, for example requiring an LP to know the full contract-market price sequence, $\mathbf{P}$, before it is realized. In this section, we focus on a specific family of implementable dynamic liquidity provision strategies, the *reset liquidity strategies*. For this, an LP at time-step t with accumulated earnings W_t may choose to *trigger* a liquidity reallocation based on the contract-market price sequence up to time t, denoted $\mathbf{P}_{\leq t}$. For reset liquidity strategies, the LP maintains a *reference bucket index* $Z_t \in \{-m, \ldots, n\}$ (correspondingly a reference bucket $B_{Z_t} \in \boldsymbol{\mu}$). We let $S_t = (Z, W_t, \mathbf{P}_{\leq t})$ denote the *system state*, and we let $\mathcal{S}$ denote the space of all possible system states. A liquidity reset consists in updating the reference bucket index and using the W_t units of B tokens at their disposal to mint a liquidity position relative to the reference bucket index Z_t.

▶ **Definition 8** (Reset liquidity provision strategy). *A* reset liquidity provision strategy (reset-LP strategy) *is composed of:*

1. *A* reference bucket update function, g, *which takes as input an arbitrary system state* $S \in \mathcal{S}$ *and updates the reference bucket index to* $Z \leftarrow Z'$ *where* $Z' = g(S)$.
2. *An* allocation function, $A : \mathcal{S} \times \mathbb{Z} \to [0, 1]$, *which specifies the fraction of budget an LP allocates to mint liquidity in each bucket relative to* Z *after a liquidity reset is triggered. More specifically,* A *gives rise to the proportional allocation* $\mathbf{x}$ *such that* $x_i = A(S, i - Z)$.
3. *A* reset condition, $h(S) \in \{0, 1\}$, *which is an indicator function for whether a reset is triggered in system state* $S \in \mathcal{S}$ *and specifies which contract-market prices, relative to the reference bucket, will trigger a liquidity reset. In the event of a trigger, a new reference bucket is computed via update function* g.

We denote a reset-LP strategy by tuple (g, h, A).

Of particular interest is the family of τ*-reset strategies*. These strategies have LPs reset liquidity when the index of the bucket containing the contract price is more than τ away from the reference index, Z. In the case of a reset, the reference bucket changes to the bucket containing the current contract price.

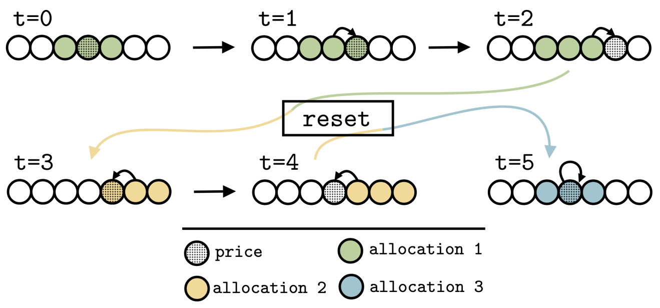


Figure 4 An illustration of how a τ-reset strategy with $\tau = 1$ can play out. Buckets are represented by circles, and for simplicity we assume that market and contract prices move together at each time step. The shaded circle represents the bucket that contract/market prices are in, and the dynamics of price movements are expressed by the smaller arrows between buckets. Colored buckets represent the contiguous $2\tau + 1$ buckets centered around an epoch's reference bucket. For this sequence, we see that price movements at $t_1 = 2$ and $t_2 = 4$ trigger resets, as the shaded bucket escapes the contiguous $2\tau + 1$ colored buckets. The specific reallocation after each trigger is specified by the allocation function A in the τ-reset strategy.

▶ **Definition 9** (τ-reset Strategy)**.** *Suppose that τ is a non-negative integer. We let $h_\tau : \mathcal{S} \to \{0, 1\}$ and $g_\tau : \mathcal{S} \to \{-m, \dots, n\}$ denote trigger and reference bucket update functions, defined for system state $S_t = (Z_t, W_t, \mathbf{P}_{\leq t}) \in \mathcal{S}$ as*

- *$h_\tau(Z_t, W_t, \mathbf{P}_{\leq t}) = 1$ if and only if $P_{c,t} \in B_i$ and $|Z_t - i| > \tau$, and*
- *$g_\tau(Z_t, W_t, \mathbf{P}_{\leq t}) = P_{c,t}$.*

We say that (g_τ, h_τ, A) is a τ-reset strategy*, for any allocation function, $A : \mathcal{S} \times \mathbb{Z} \to [0, 1]$.*

We illustrate the versatility of τ-reset strategies through some examples:

1. (Static Strategies): For $\tau > T$, i.e., the time horizon of $\mathbf{P}$, we recover static strategies.
2. (Uniform τ-Reset Strategies): Allocating liquidity uniformly on a range of contiguous buckets centered around the current reference bucket B_{Z_t} and resetting when prices move outside of this range.
3. (Context-Independent Allocation Strategies): Setting $A(S, i) = A_i \in \mathbb{R}$ for all $S \in \mathcal{S}$; i.e., the proportional allocations relative to baseline bucket index are always the same at the time of a reset trigger.

4 Optimizing Earnings

In this section, we formulate the earnings optimization problem faced by an LP with *belief, $\mathcal{P}$,* defining a distribution on contract-market price sequences in a given time horizon. In the most general case, belief $\mathcal{P}$ would depend on the liquidity allocation strategy used by an LP. For example, if the LP provides a large amount of liquidity for a given price interval, this would in turn reduce the slippage of trades at those prices, which may in turn increase the volume of trades facilitated by the contract, and hence change $\mathcal{P}$. As in [14], we make the simplifying assumption, reasonable for a small LP, that their belief $\mathcal{P}$ is a *liquidity-independent distribution*, and independent of the strategic liquidity strategy used by the LP. Going forward, we limit our attention to liquidity-independent beliefs. In particular, we will model non-arbitrage traders who trade to a particular buy or sell contract price whose value is unaffected by this LP's liquidity allocation, along with arbitrage traders whose trades are triggered by considerations of market price vs contract price.

4.1 Optimal τ-reset Strategies

We've seen that τ-reset strategies are a versatile framework for dynamic liquidity provision. For a given value of τ, the only choice in defining a τ-reset strategy is the allocation function A, and we let $\Lambda_\tau(A) = (g_\tau, h_\tau, A)$ denote the resulting τ-reset strategy.

In this section, we provide a means of optimizing expected earnings for a given τ. For this, we assume that $A \in \mathcal{A}$, where $\mathcal{A}$ is a family of allocation functions. The space of all allocation functions is large, with an allocation function potentially depending on the entire history of contract-market price sequences and LP actions up to the point when a liquidity reset is triggered.

In defining an LP's optimization problem, we consider LPs with different levels of *risk-aversion*, encoded by a *utility function*, $u : \mathbb{R} \to \mathbb{R}$ (we provide example utility functions below), and we assume that an LP wants to select an allocation function to maximize $V^u_{\tau,\mathcal{P}}(A) = \mathbb{E}_{\mathbf{P}\sim\mathcal{P}}\left[u(V(\Lambda_\tau(A), \mathbf{P})\right]$. With this in hand, we let $OPT(\tau, \mathcal{P}, u, \mathcal{A}) = \max_{A\in\mathcal{A}} V^u_{\tau,\mathcal{P}}(A)$, and denote an allocation function in family $\mathcal{A}$ that achieves optimal earnings by A^*.

In general, convex u and concave u correspond to risk-seeking and risk-averse LPs, respectively, and linear u corresponds to a risk-neutral LPs (where we can adopt $u(x) = x$ without loss of generality). Going forward, we adopt as the utility function that with constant Arrow-Pratt measures of absolute risk-aversion [8, 22].

▶ **Definition 10** (Constant Absolute Risk Aversion Utility). *For a given $a \in \mathbb{R}$, the* constant absolute risk aversion utility function, *$u_a : \mathbb{R} \to \mathbb{R}$, is given by:*

$$u_a(x) = \begin{cases} \left(1 - e^{-ax}\right)/a & \textit{if } a \neq 0, \textit{ and} \\ x & \textit{otherwise.} \end{cases} \tag{2}$$

For $a < 0$, $a = 0$, and $a > 0$, utility function u_a models a risk-averse, risk-neutral, and risk-seeking agent, respectively.

4.2 Sampling to Approximate OPT

In order to optimize $V^u_{\tau,\mathcal{P}}(A)$, we approximate the objective by taking a discrete sample of paths from $\mathcal{P}$. As such, suppose that $\mathbf{P}_1, \ldots, \mathbf{P}_N \sim \mathcal{P}$. We define the empirical average earnings of an LP given the sample paths as $\hat{V}^u_\tau(A \mid \mathbf{P}_1, \ldots, \mathbf{P}_N) = \frac{1}{N}\sum_{q=1}^{N} u(V(\Lambda_\tau(A), \mathbf{P}_q)$. Going forward, we approximate $OPT(\tau, \mathcal{P}, u, \mathcal{A})$ by taking sufficiently many samples from $\mathcal{P}$ and optimizing $\hat{V}^u_\tau$.

4.3 Computing Optimal τ-reset Strategies with Neural Networks

We compute optimal τ-reset strategies by letting the allocation function, A, be parametrized by a feedforward neural network (NN) with parameters given by $\theta \in \boldsymbol{\theta}$. We let A_θ denote the specific allocation function for a given parameter choice $\theta \in \boldsymbol{\theta}$ and we let $\mathcal{A}_{\boldsymbol{\theta}}$ denote the space of all possible parametric settings of the NN. Our objective is to maximize $\hat{V}^u_\tau(A_\theta \mid \mathbf{P}_1, \ldots, \mathbf{P}_N)$ for a given sample of contract/market price paths $\mathbf{P}_1, \ldots, \mathbf{P}_N \sim \mathcal{P}$.

When a reallocation is triggered at the beginning of epoch $j (j = 1, 2, \ldots, k)$, the NN takes as input a set of features C^j that contains context information. The set of context features includes the current time step, the current wealth, the current pool price, the current bucket that the pool price lies in, and an exponentially-weighted moving average (the smoothing parameter value is 0.1) of non-arbitrage trade volume that a hypothetical 1 unit of liquidity over the entire price range would have achieved given the price trajectory.

We use a fully connected neural network architecture with 5 hidden layers, and the size of each hidden layer is $m = 16$. The size of the input layer is $n = 5$ (the number of context features), and the output layer is of size $s = 2\tau + 2$ (the first $2\tau + 1$ dimensions are the proportional capital to be allocated into the corresponding buckets, and there is also a special dimension for not allocating some of the wealth if needed). All the hidden layers are associated with the ReLU activation function. In addition, a soft-max function is added for the final output in order to produce a vector of sum 1. The architecture we use is visualized in the right image of Figure 5.

Unpacking the objective, $\hat{V}^u_\tau(A_\theta \mid \mathbf{P}_1, \ldots, \mathbf{P}_N)$ shows a fundamental recurrence in the given allocation function A_θ. This is because an allocation produced by A_θ is deployed into the pool and affects the value of wealth when the next reallocation is triggered, and wealth is used as part of the input to A_θ for the new reallocation as visualized in the left diagram of Figure 5. However, the NN representation of A allows gradients to be pushed through the recurrence with standard back propagation methods used for recurrent neural networks. Given this, we find optimal $\theta \in \boldsymbol{\theta}$ via standard gradient descent methods.

In more detail, for optimization of the NN (ODRA) and the constant allocation (OIRA)[4], we use stochastic gradient descent based on sampled price trajectories. The number of training steps is 10000 for both optimize methods. The learning rate for the NN is 10^{-3} while the learning rate for the constant allocation is 10^{-2}. In addition, the Adam optimizer is used for both methods.[5]

As risk aversion parameter a increases, the relative difference between the utility values of two wealth values $(u_a(x_1) - u_a(x_2))/u_a(x_1)$ becomes smaller and this could pose a challenge to the optimization of ODRA and OIRA when the improvement of utility value is numerically very small. To resolve this issue, we apply a positive affine transformation to the utility values as $u^*_a(x) = (u_a(x) - u_a(1))/(u_a(1.1) - u_a(1))$ for all x and use the transformed utility values $u^*_a(x)$ in the loss function during training of ODRA and OIRA. This helps the optimization process and at the same time does not alter the problem formulation of the optimization as utility functions u_a and u^*_a represent the same set of underlying preferences.

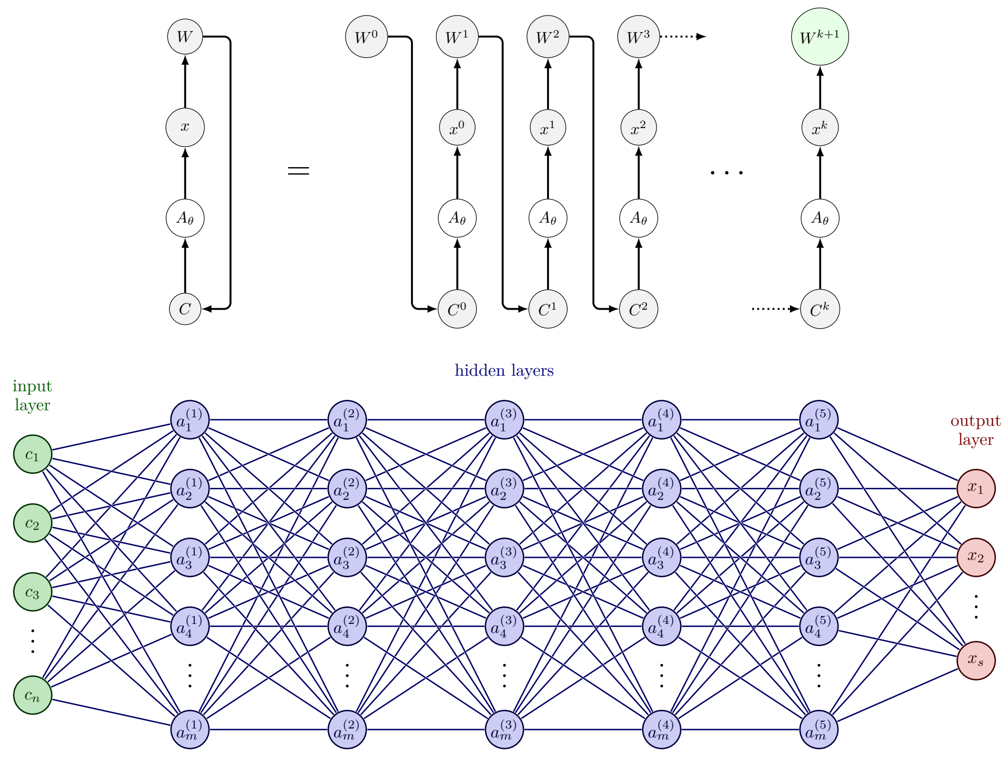
4.4 Liquidity Provision Strategies

Below we outline the main strategies we compare in different regimes:

1. Optimal static allocation (OSA): This strategy computes $\mathbf{x}$ that optimizes the value of $u_a(V(\mathbf{x} \mid \mathbf{P}_1, \ldots, \mathbf{P}_N))$ from Section 3.1. This is the only strategy that does not explicitly use liquidity reallocations (though it can be seen as a τ-reset strategy with $\tau > T$), and it coincides with the work of [14].
2. Uniform liquidity τ-reset allocation (ULRA): For a fixed τ, this strategy mints an equal μ units of liquidity for each of the $2\tau + 1$ contiguous buckets considered in a reallocation. τ is chosen to be as large as possible so the LP makes use of the entire wealth at their disposal at a reset to reallocate liquidity.

[4] In the appendix of the full version of the paper we also provide a natural variant of OIRA for LPs exhibiting risk-aversion via logarithmic utilities as in [11]. For this model we provide convex optimization methods to solve for optimal allocations.

[5] The codebase we use to run experiments is available open-source at `https://github.com/Evensgn/uniswap-active-lp`.



■ **Figure 5** The top image provides a visualization of the recurrence in A_θ for the overall objective $\hat{V}^u_\tau(A_\theta \mid \mathbf{P}_1, \ldots, \mathbf{P}_N)$ which we exploit to compute gradients in a similar fashion to recurrent neural networks. In this image, C denotes the context that is fed to the neural network A_θ as features. A relevant feature at each epoch is the wealth that the LP has accumulated at the beginning of the epoch W^i, which is exemplified via an arrow in the figure. The overall objective is the given utility function applied to the wealth at the end of the final epoch W^{k+1}. The bottom image provides a visualization of the neural network architecture we use for A_θ. There is a soft-max function applied to the output layer. The recurrent structure of the objective's dependence with respect to the NN parameters, $\theta \in \boldsymbol{\theta}$ allow us to use techniques from recurrent neural networks to compute the gradient of the objective $u(W^{k+1})$ with respect to θ.

3. Uniform proportional τ-reset allocation (UPRA): For a fixed τ, this allocates wealth in equal proportions to each of the $2\tau + 1$ buckets after a reset (in general this does not result in a uniform liquidity allocation as the cost per unit of liquidity in each bucket may be different).
4. Optimal context-independent τ-reset allocation (OIRA): For a fixed τ, this computes the optimal single allocation vector to be used at every reset. In other words, the LP computes an optimal $(A_{-\tau}, \ldots, A_\tau)$, to be used to allocate liquidity around the reference bucket at each reallocation.
5. Optimal context-dependent τ-reset allocation (ODRA): For fixed τ, this is solved with the Neural Network formulation of Section 4.3.

5 Experimental Setup: Contract-Market Prices

In this section, we describe a family of empirically-informed contact-market price sequences against which we will optimize τ-reset strategies and we use historical data to inform this stochastic price model.

5.1 Modeling Contract-Market Prices

For this, we use a similar approach to [14], which is in turn inspired by [10], to provide a family of liquidity-independent contract-market price distributions. This makes use of an external stochastic process to define a sequence of market prices, together with non-arbitrage trades that affect the contract price and arbitrage trades that act to bring contract prices closer to market prices.

We assume that contract-market prices are generated over each of $R > 0$ *rounds*. At the beginning of each round, market prices change randomly according to a stochastic process $\mathcal{P}_M$. During the r-th round, after the contract-market price update, there are some number, $k_r \geq 0$, of non-arbitrage trades which impact contract price, P_c, only. Each non-arbitrage trade is either a purchase or a sale, this determined uniformly at random with probability $1/2$. The effect of such a trade is that the contract price changes to $(1+\lambda_r)P_c$ or $(1+\lambda_r)^{-1}P_c$ respectively, where $\lambda_r > 0$, depending on whether a purchase or sale occured. Crucially, trades are price-based in our model rather than volume based, which in turn ensures that contract-market prices evolve independent of liquidity provided by an LP. It is precisely this exogenous uncertainty to LP actions that will allows us to compute optimal τ-reset strategies via the methods of Section 4, as this allows us to sample price paths first and then optimize LP allocation functions.

We also model arbitrage trades whose role is to bring contract prices close to market prices. For this, we follow [14], and with a Uniswap contract fee rate, $\gamma \in (0,1)$, we let $I_\gamma(P_m) = [(1-\gamma)P_m, (1-\gamma)^{-1}P_m]$ be the *no-arbitrage interval* around the market price P_m. If the contract price exits this no-arbitrage interval, we assume a arbitrage trade brings the contract price to the closest price in the interval. That is, if $P_c < (1-\gamma)P_m$ we assume that arbitrage trade moves the contract price to $(1-\gamma)P_m$, and if $P_c > (1-\gamma)^{-1}P_m$ we assume that arbitrage trade moves the contract price to $(1-\gamma)^{-1}P_m$.

▶ **Definition 11** (Round-Based Liquidity-Independent Price Distribution). *We say that $\mathcal{P}$ is a* round-based liquidity-independent price distribution *when it is a distribution that is parameterized by:*

- $R > 0$*: the number of rounds,*
- $\gamma \in (0,1)$*: the fee rate of the Uniswap contract,*
- $\mathcal{P}_M$*: the stochastic process governing market price updates at the beginning of each round,*
- $\mathbf{k} = (k_r)_{r=1}^R$ *with* $k_r > 0$*: the number of non-arbitrage trades in each round* $r \in \{1,\dots,R\}$*, and*
- $\boldsymbol{\lambda} = (\lambda_r)_{r=1}^R$ *with* $\lambda_r > 0$*: the multiplicative impact of a non-arbitrage trade on contract price for each round* $r \in \{1,\dots,R\}$*.*

When we wish to specify the resulting round-based price distribution, we write this as $\mathcal{P}(R,\gamma,\mathcal{P}_M,\mathbf{k},\boldsymbol{\lambda})$*.*

We model $\mathcal{P}_M$ as a geometric Brownian motion with parameters estimated from historical price data between token pairs. We also explore multiple regimes of time-varying non-arbitrage trade by varying $\boldsymbol{\lambda}$ (the framework is flexible enough to permit arbitrary values of λ_r for each round).

5.2 Market Prices as a Geometric Brownian Motion

We model the stochastic nature of market prices, $\mathcal{P}_M$, as a Geometric Brownian Motion (GBM). If the time series is given by $X_1,\dots,X_T$, then the successive multiplicative increments of the time series are i.i.d lognormally distributed. If we let $Z_i = \log\left(\frac{X_i}{X_{i-1}}\right)$, then $Z_2,\dots Z_T \sim$

iid $\mathcal{N}(\mu, \sigma^2)$ with *drift*, μ, and *diffusion*, σ. We estimate these parameters on per-minute time series data for ETH/BTC prices (the *low volatility regime*) and ETH/USDT (the *high volatility regime*) from March 2022 through February 2023. For each time series, we estimate the drift and diffusion via standard MLE methods. The estimates obtained were $(\hat{\mu}, \hat{\sigma}^2) = (4.835 \times 10^{-8}, 1.946 \times 10^{-7})$ and $(\hat{\mu}, \hat{\sigma}^2) = (-1.140 \times 10^{-6}, 8.329 \times 10^{-7})$ for the low and high volatility regimes respectively.

5.3 Contract Price Updates

Whereas arbitrage trades are specified by the fee rate of the contract, non-arbitrage trades are parametrized by $\mathbf{k} = (k_r)_{r=1}^R$ and $\boldsymbol{\lambda} = (\lambda_r)_{r=1}^R$, which specify the number and multiplicative magnitude of price change updates arising from non-arbitrage trades in a given round. In our experiments, we fix $k_r = 10$ for each round and introduce time-varying non-arbitrage price flow by explicitly modulating $\boldsymbol{\lambda}$ before sampling from $\mathcal{P}$. In particular we explore $\boldsymbol{\lambda}$ such that $\lambda_r = \bar{\lambda} + \alpha \cdot tanh(10(t/T - 0.5))$, where $\bar{\lambda} > 0$ is the average λ_r value over the time horizon and $\alpha > 0$ is the variation exhibited in λ_r around the mean.

6 Experimental Results

In this section, we explore the increase in earnings that LPs can gain through the use of dynamic allocation strategies, studying the performance of various liquidity provision strategies in a multitude of economic environments modulated by contract/market price volatility, LP risk-aversion, and reallocation costs. Most importantly, we find many settings in which optimal τ-reset strategies outperform simpler liquidity provision strategies. In all the experiments that follow, we assume a default setting of $(W, \gamma, R, k_r, \bar{\lambda}, \alpha, \eta) = (1, 0.003, 1000, 10, 0.00005, 0.00005, 0.01)$. When deviating from the default setting we clarify which parameters are changed. In addition, we assume that the buckets of the v3 contract $\boldsymbol{\mu} = \{B_{-m}, .., B_n\}$ are given by $B_i = [a_i, b_i] = [\phi^i, \phi^{i+1}]$ for $\phi = 1.0001^{10}$.

6.1 The Impact of Price Volatility

In Figures 6 and 7 we plot the performance of all LP strategies as we modulate $\mathcal{P}_M$ from low to high volatility as well as the risk-aversion of the LP. Figure 6 focuses on only comparing OIRA, ODRA and OSA to tease out the relative performance of ODRA vs. OIRA. Figure 7 incorporates UPRA and ULRA, from where we can see that their performance is almost identical. The NN-based ODRA outperforms all strategies, especially OSA which does not make use of reallocations. As risk-aversion increases, we see that the distinction between ODRA and OIRA becomes more clear in the plots, however, this does not imply a greater magnitude of performance due to the fact that different risk-aversion values give rise to different scales. In addition, we see that OIRA generally exhibits optimal performance with $\tau > 1$ whereas all other τ-reset strategies in this setting perform better with $\tau = 1$.

We see that for lower τ values, higher $\mathcal{P}_M$ leads to a greater separation between ODRA and OIRA in performance. Moreover, Figure 8 plots allocation profiles for ODRA and OIRA as we modulate risk-aversion and $\mathcal{P}_M$. As expected, with higher risk-aversion we see a larger spread in allocations, as LPs may seek to decrease the variance in their earnings with wider positions. On the other hand, as $\mathcal{P}_M$ increases in volatility, we see that LP positions for both ODRA and OIRA become narrower. This is likely due to the fact that the expected number of reallocations is higher in the high volatility setting than in the low volatility setting for the same τ. For a lower frequency of reallocations, the allocated liquidity is used for longer time periods, hence an LP may wish to spread liquidity over various buckets.

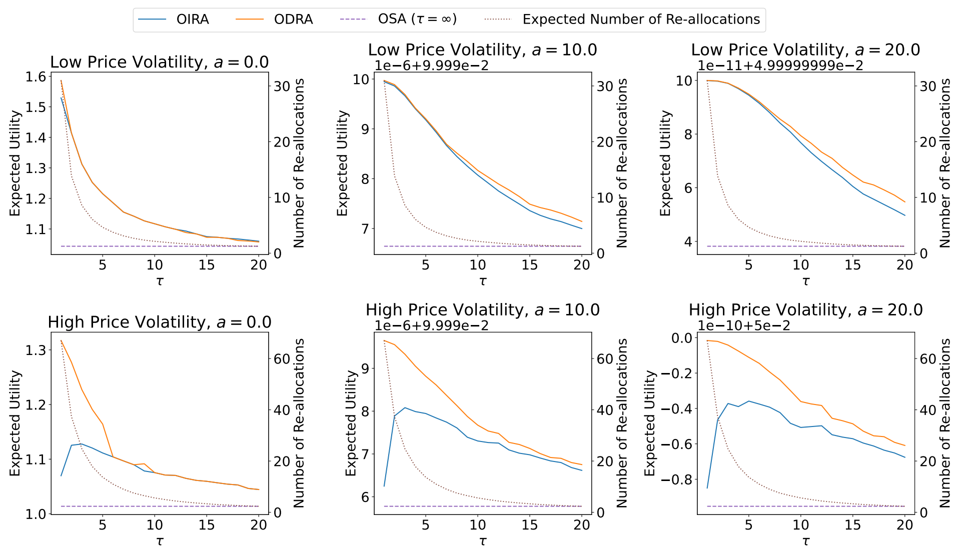


Figure 6 The performance of OIRA, ODRA and OSA strategies as we modulate both risk-aversion and $\mathcal{P}_M$. For each strategy, we plot the expected utility it achieves as a function of τ, and we also plot the expected number of reallocations that occur as a function of τ. The top row corresponds to a low volatility $\mathcal{P}_M$, empirically informed from ETH/BTC prices, and the bottom row corresponds to high volatility $\mathcal{P}_M$, empirically informed from ETH/USDC prices. The columns correspond to risk-aversion values $a = 0, 10$, and 20, respectively from left to right. When scientific notation is used for the y-axis values in certain subplots, it is denoted by a number above the respective y-axis.

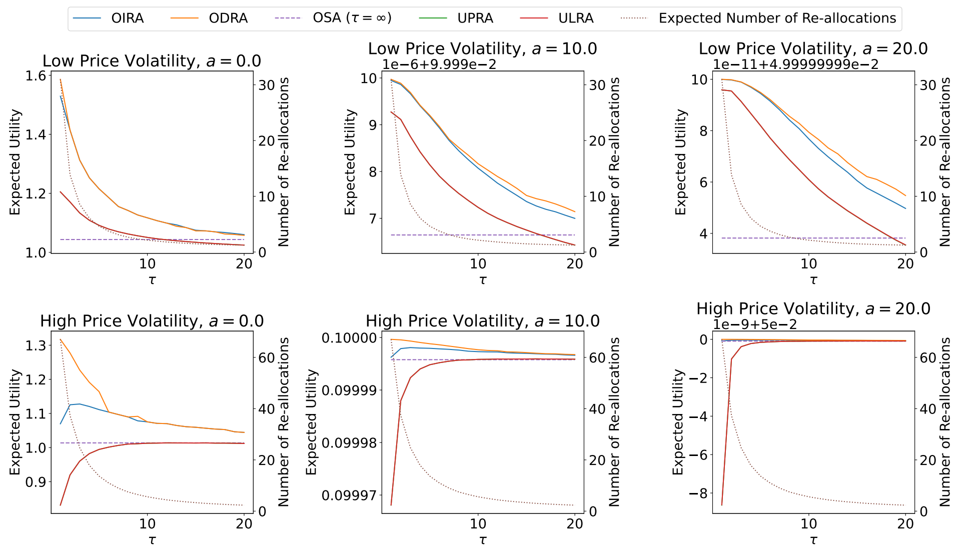


Figure 7 The performance of all strategies as we modulate both risk-aversion and $\mathcal{P}_M$. For each strategy we plot the expected utility it achieves as a function of τ, and we also plot the expected number of reallocations that occur as a function of τ. The top row corresponds to a low volatility $\mathcal{P}_M$, empirically informed from ETH/BTC prices, and the bottom row corresponds to high volatility $\mathcal{P}_M$, empirically informed from ETH/USDC prices. The columns correspond to risk-aversion values $a = 0, 10$, and 20, respectively from left to right. When scientific notation is used for the y-axis values in certain subplots, it is denoted by a number above the respective y-axis.

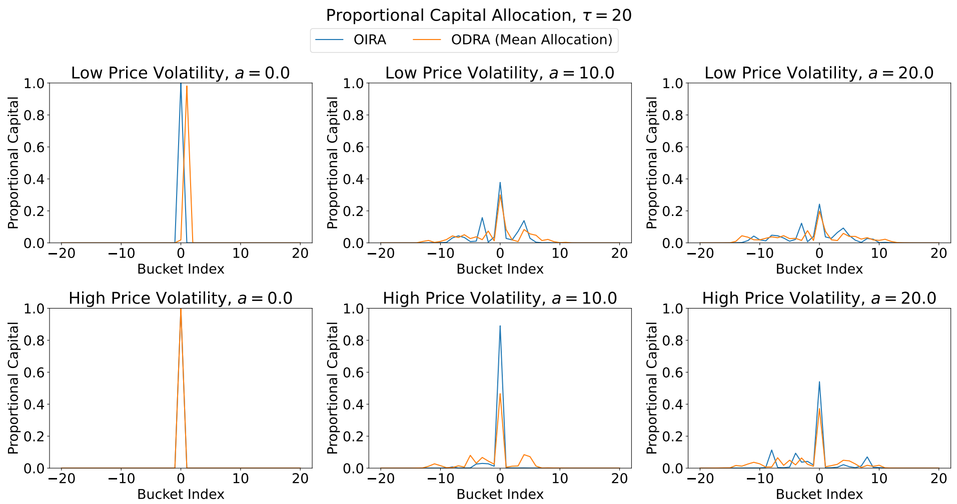


Figure 8 OIRA allocation and ODRA average allocations for $\tau = 20$ as we modulate both risk-aversion and $\mathcal{P}_M$. The top row corresponds to a low volatility $\mathcal{P}_M$, empirically informed from ETH/BTC prices, and the bottom row corresponds to high volatility $\mathcal{P}_M$, empirically informed from ETH/USDC prices. The columns correspond to risk-aversion values $a = 0, 10$, and 20, respectively from left to right.

6.2 Varying Non-arbitrage Flow

In Figure 9 we modulate the volatility of $\mathcal{P}_M$ and the magnitude of non-arbitrage flow in $\boldsymbol{\lambda}$ by modulating α, the amplitude of change in $\boldsymbol{\lambda}$ while keeping mean $\boldsymbol{\lambda}$ the same. The most salient observation is that as $\boldsymbol{\lambda}$ becomes more time-varying, the NN-based approach of ODRA increases its performance relative to OIRA. This is to be expected due to the fact that ODRA can incorporate temporal context in deciding an allocation after a reset, and the non-arbitrage flow inherently has the temporal context of increased importance as α increases.

In Figure 10 and 11 we also modulate $\boldsymbol{\lambda}$ albeit by jointly modulating amplitude(α) and mean of λ_r values, $\bar{\lambda}$. Once more we see that the NN-based ODRA strategy outperforms all strategies, and we see that the optimal τ values for ODRA drastically differ in the high volatility $\mathcal{P}_M$ over those of Figure 9. Moreover in Figure 11 we see that both increased non-arbitrage flow and $\mathcal{P}_M$ volatility contribute to more spread allocations. LPs make profits from non-arbitrage trades, hence they stand to obtain more fees with wider positions for larger flows of non-arbitrage trade.

6.3 The Impact of Risk-aversion

As mentioned in the previous sections, risk-aversion mostly impacts the allocations used in ODRA and OIRA LP strategies. In Figure 12 we make fine-grained modulations of risk-aversion and see that indeed LPs spread their liquidity more as they become more risk-averse. A larger spread of liquidity allocation typically implies lower expected earnings for an LP as they have less proportional liquidity at prices that are traded at, but at the same time, there is less risk of missing out on fees due to prices escaping their position or suffering impermanent loss due to price deviating from the initial price.

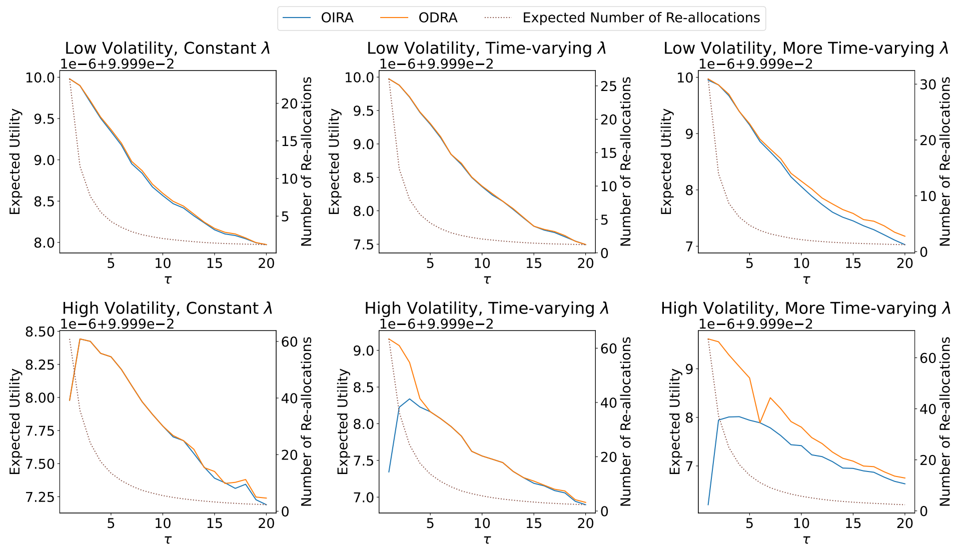


Figure 9 The performance of OIRA and ODRA strategies as we modulate $\mathcal{P}_M$ and $\boldsymbol{\lambda}$. For all plots, we use $a = 10$ for risk aversion and $\bar{\lambda} = 0.00005$ and we modulate the α in $\{0.0, 0.00003, 0.00005\}$ in columns from left to right. The top row plots low volatility $\mathcal{P}_M$ and the bottom row plots high volatility $\mathcal{P}_M$. When scientific notation is used for the y-axis values in certain subplots, it is denoted by a number above the respective y-axis.

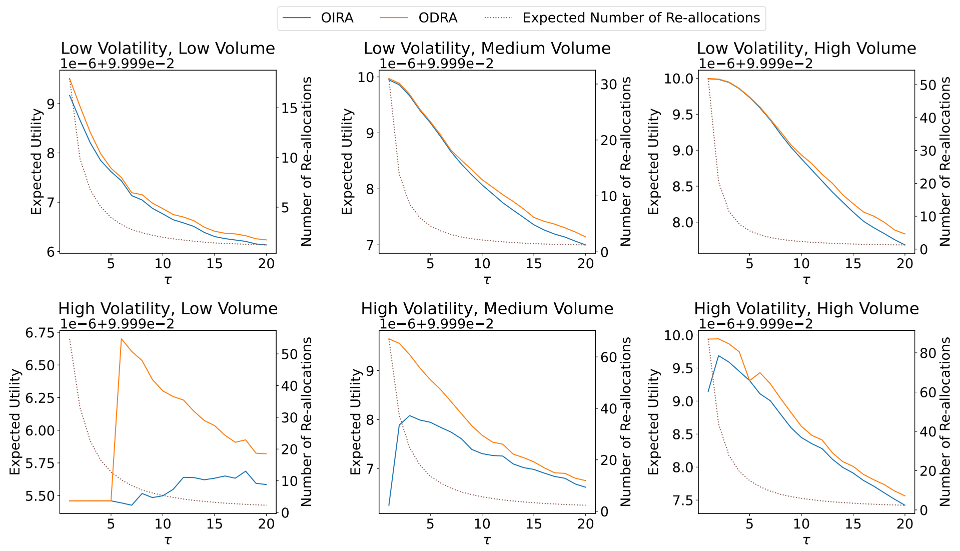


Figure 10 The performance of OIRA and ODRA strategies as we modulate $\mathcal{P}_M$ and $\boldsymbol{\lambda}$. For all plots, we use $a = 10$ for risk aversion and we modulate the $(\bar{\lambda}, \alpha) \in \{(0.000025, 0.000025), (0.00005, 0.00005), (0.000075, 0.000075)\}$ in columns from left to right. The top row plots low volatility $\mathcal{P}_M$ and the bottom row plots high volatility $\mathcal{P}_M$. When scientific notation is used for the y-axis values in certain subplots, it is denoted by a number above the respective y-axis.

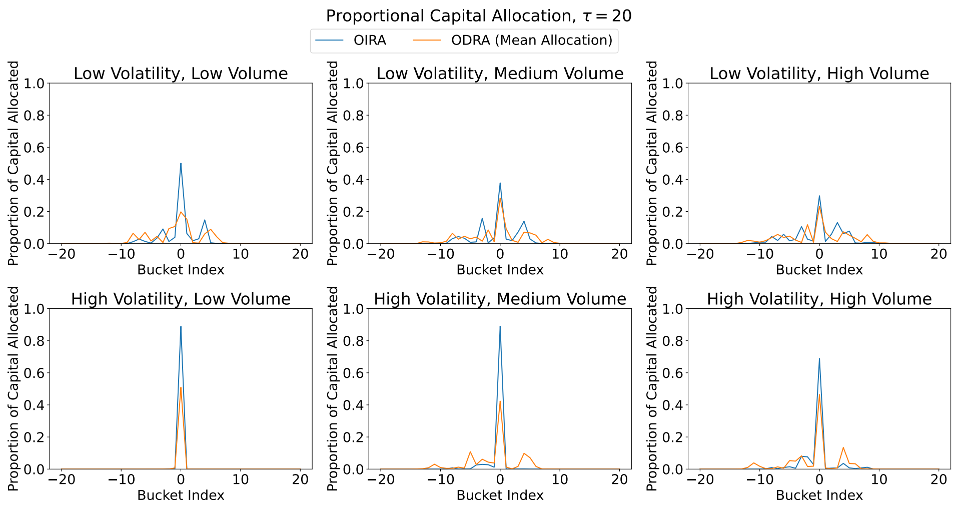


Figure 11 OIRA allocation and ODRA average allocation as we modulate $\mathcal{P}_M$ and $\boldsymbol{\lambda}$. For all plots, we use $a = 10$ for risk aversion and we modulate the $(\bar{\lambda}, \alpha) \in \{(0.000025, 0.000025), (0.00005, 0.00005), (0.000075, 0.000075)\}$ in columns from left to right. The top row plots low volatility $\mathcal{P}_M$ and the bottom row plots high volatility $\mathcal{P}_M$.

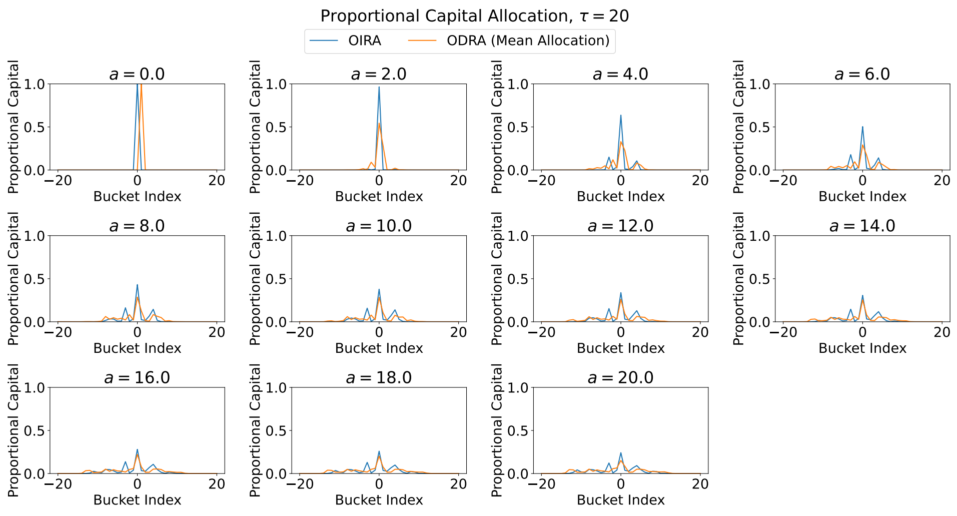


Figure 12 OIRA allocation and ODRA average allocation for $\tau = 20$ for low volatility $\mathcal{P}_M$ as we modulate risk-aversion from $a = 0$ to $a = 20$.

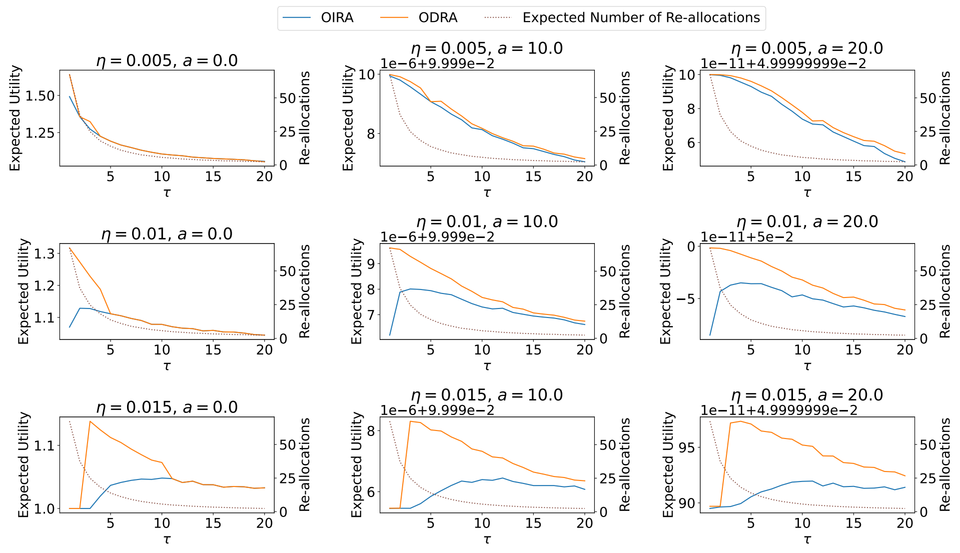


Figure 13 The performance of OIRA and ODRA strategies as we modulate η and risk-aversion. For all plots, we use high volatility $\mathcal{P}_M$. We modulate a in $\{0, 10, 20\}$ in columns from left to right and $\eta \in \{0.005, 0.01, 0.015\}$ in rows from top to bottom. When scientific notation is used for the y-axis values in certain subplots, it is denoted by a number above the respective y-axis.

6.4 The Impact of Reallocation Costs

In Figure 13 we modulate the cost of reallocation, η. We see that higher η values lead to higher optimal τ values for both OIRA and ODRA strategies. This is to be expected, for although low τ values might lead to higher gains in fees, this also leads to more frequent resets which in turn come with a higher cost.

7 Conclusion

This paper fills existing gaps in the literature regarding strategic liquidity provision strategies for LPs in Uniswap v3. Whereas earlier important work has either optimized for complex liquidity positions in static environments, or simple positions with dynamic reallocations, our work simultaneously provides complex, context-dependent liquidity allocations that dynamically reallocate as prices evolve in v3 contracts. Our results show that such liquidity provision strategies provide large gains for LPs in multiple economic environments for decentralized exchanges. Natural directions of future work include: incorporating LVR [20] into the objective definition to optimize the strategy of a delta-hedged LP, incorporating a game-theoretic framework to liquidity provision which is more apt for large LPs and modelling competition between different pools such as v2 and v3 pools for same token pairs.

References

1 Hayden Adams. Uniswap whitepaper, 2018. URL: `https://hackmd.io/@HaydenAdams/HJ9jLsfTz`.

2 Hayden Adams, Noah Zinsmeister, and Dan Robinson. Uniswap v2 core, 2020. URL: `https://uniswap.org/whitepaper.pdf`.

3 Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. Uniswap v3 core, 2021. URL: `https://uniswap.org/whitepaper-v3.pdf`.

4 Guillermo Angeris and Tarun Chitra. Improved price oracles: Constant function market makers. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 80–91, 2020.

5 Guillermo Angeris, Alex Evans, and Tarun Chitra. When does the tail wag the dog? Curvature and market making. *arXiv preprint arXiv:2012.08040*, 2020.

6 Guillermo Angeris, Hsien-Tang Kao, Rei Chiang, Charlie Noyes, and Tarun Chitra. An analysis of Uniswap markets. *arXiv preprint arXiv:1911.03380*, 2019.

7 Jun Aoyagi. Lazy liquidity in automated market making. *Available at SSRN 3674178*, 2020.

8 Kenneth Joseph Arrow. *Aspects of the theory of risk-bearing.* Helsinki, 1965.

9 Yogev Bar-On and Yishay Mansour. Uniswap liquidity provision: An online learning approach. *arXiv preprint arXiv:2302.00610*, 2023.

10 Agostino Capponi and Ruizhe Jia. The adoption of blockchain-based decentralized exchanges. *arXiv preprint arXiv:2103.08842*, 2021.

11 Álvaro Cartea, Fayçal Drissi, and Marcello Monga. Decentralised finance and automated market making: Predictable loss and optimal liquidity provision. *Available at SSRN 4273989*, 2022.

12 Alex Evans. Liquidity provider returns in geometric mean markets. *arXiv preprint arXiv:2006.08806*, 2020.

13 Alex Evans, Guillermo Angeris, and Tarun Chitra. Optimal fees for geometric mean market makers. *arXiv preprint arXiv:2104.00446*, 2021.

14 Zhou Fan, Francisco J. Marmolejo Cossío, Ben Altschuler, He Sun, Xintong Wang, and David C. Parkes. Differential liquidity provision in Uniswap v3 and implications for contract design. In *3rd ACM International Conference on AI in Finance, ICAIF*, pages 9–17, 2022.

15 Rafael Frongillo, Maneesha Papireddygari, and Bo Waggoner. An axiomatic characterization of CFMMs and equivalence to prediction markets. *arXiv preprint arXiv:2302.00196*, 2023.

16 Mohak Goyal, Geoffrey Ramseyer, Ashish Goel, and David Mazières. Finding the right curve: Optimal design of constant function market makers. *arXiv preprint arXiv:2212.03340*, 2022.

17 Lioba Heimbach, Eric Schertenleib, and Roger Wattenhofer. Risks and returns of Uniswap v3 liquidity providers. *arXiv preprint arXiv:2205.08904*, 2022.

18 Max. Introducing Alpha Vaults–an LP strategy for Uniswap V3, 2021. URL: `https://medium.com/charmfinance/introducing-alpha-vaults-an-lp-strategy-for-uniswap-v3-ebf500b67796`.

19 Jason Milionis, Ciamac C Moallemi, and Tim Roughgarden. A Myersonian framework for optimal liquidity provision in automated market makers. *arXiv preprint arXiv:2303.00208*, 2023.

20 Jason Milionis, Ciamac C Moallemi, Tim Roughgarden, and Anthony Lee Zhang. Automated market making and loss-versus-rebalancing. *arXiv preprint arXiv:2208.06046*, 2022.

21 Michael Neuder, Rithvik Rao, Daniel J Moroz, and David C Parkes. Strategic liquidity provision in uniswap v3. *arXiv preprint arXiv:2106.12033*, 2021.

22 John W Pratt. Risk aversion in the small and in the large. *Econometrica*, 32:122–136, 1964.

23 Jan Christoph Schlegel, Mateusz Kwaśnicki, and Akaki Mamageishvili. Axioms for constant function market makers. *Available at SSRN*, 2022.

24 Martin Tassy and David White. Growth rate of a liquidity provider's wealth in $xy = c$ automated market makers, 2020.

25 Dave White, Martin Tassy, Charlie Noyes, and Dan Robinson. Uniswap's financial alchemy, 2020. URL: `https://research.paradigm.xyz/uniswaps-alchemy`.

26 Wenqi Zhao, Hui Li, and Yuming Yuan. Understand volatility of Algorithmic Stablecoin: Modeling, verification and empirical analysis. In *Financial Cryptography and Data Security*, volume 12676 of *Lecture Notes in Computer Science*, pages 97–108. Springer, 2021.