Received: 3 September 2023 | Revised: 25 October 2023 | Accepted: 2 January 2024

DOI: 10.1002/nem.2261

SPECIAL ISSUE PAPER

WILEY

DEEPER: A shared liquidity decentralized exchange design for low trading volume tokens to enhance average liquidity

Srisht Fateh Singh[1] | **Panagiotis Michalopoulos**[1] | **Andreas Veneris**[1,2]

[1]Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada

[2]Department of Computer Science, University of Toronto, Toronto, ON, Canada

Correspondence
Srisht Fateh Singh, Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada.
Email: srishtfateh.singh@mail.utoronto.ca

Abstract

This paper presents DEEPER, a design for a decentralized exchange that enhances liquidity via reserve sharing. By doing this, it addresses the problem of shallow liquidity in low trading volume token pairs. Shallow liquidity impairs the functioning of on-chain markets by creating room for unwanted phenomena such as high slippage and sandwich attacks. DEEPER solves this by allowing liquidity providers of multiple trading pairs against a common token to share liquidity. This is achieved by creating a common reserve pool for the shared token that is accessible by each trading pair. Independent from the shared liquidity, providers are free to add liquidity to individual token pairs without any restriction. The trading between one token pair does not affect the price of other token pairs even though the reserve of the shared token changes. The proposed design is an extension of concentrated liquidity automated market maker DEXs that is simple enough to be implemented on smart contracts. This is demonstrated by providing a template for a hook-based smart contract that adds our custom functionality to UNISWAP V4. Experiments on historical prices show that for a batch consisting of eight trading pairs, DEEPER enhances liquidity by over 2.6–5.9×. The enhancement in liquidity can be increased further by increasing the participating tokens in the shared pool. While providing shared liquidity, liquidity providers should be cautious of certain risks and pitfalls, which are described. Overall, DEEPER enables the creation of fair markets for low trading volume token pairs.

1 | INTRODUCTION

Blockchains are distributed ledger technologies where modifying the ledger state requires consensus among participants, also known as miners, who maintain the chain. Bitcoin[1] was the first public and permissionless blockchain to be resistant against adversarial miners provided the majority of participants are honest. This introduced the concept of trustless payments where a transfer is guaranteed regardless of the sender, receiver, or amount. Although Bitcoin was only limited to payments, Ethereum[2] extended blockchain functionalities to arbitrary Turing-complete contracts known

as *smart contracts*. The correct execution of these contracts is based on a trustless execution model. This led to the onset of decentralized finance (DeFi)[3] that removed the need for centralized intermediaries, thus eliminating central points of failure.

This trend of bringing decentralization and introducing trustlessness continues further in several other traditional technology sectors. This includes revolutionizing research platforms with decentralized science,[4] gaming industry with game finance,[5] web services with Web 3.0,[6] supply chains,[7] and social engagement platforms.[8] One key aspect in designing a decentralized system, including blockchains, is economic incentives.[9] These incentives align a distributed and unfamiliar group of users towards a common goal. Such incentives are usually implemented as cryptocurrency tokens with each application having its own token.

Apart from just providing incentives, cryptocurrency tokens can also serve many other diverse functionalities. Some of them include providing utility in web applications, such as access to data storage or computing power,[10] governance rights in decentralized autonomous organizations (DAOs) via voting,[11] reputation and identity score,[12] tokenization of real-world assets,[13] and so forth.[14] Since each of these tokens has either a limited or programmed supply, they obtain real-life economic value, and thus marketplaces, both centralized and decentralized, have been developed for trading them. Tokens are either traded on a centralized exchange (CEX) that adopts the traditional central limit order book (CLOB) mechanism or a decentralized exchange (DEX) running automated market making (AMM) algorithms. In contrast with centralized exchanges, decentralized exchanges allow customers to have self-custody of their funds, eliminate insider activities, distribute protocol fees to market participants, and ensure the solvency of the exchange.[15] That being said, certain challenges must be addressed in the realm of decentralized exchanges to fully harness their potential benefits and ensure their widespread adoption.

In AMMs, liquidity providers (LPs) create a *liquidity pool* by depositing a pair of assets that other traders can swap from. For every swap, the trader pays a fee, proportional to the swap amount, that goes to the LPs. Traders prefer an asset pair with significant liquidity, that is, higher asset reserves because otherwise, it can lead to unconventional price movements also known as "*slippage*." Evidently, higher slippage results in worse execution prices for large trades, and also, the trading pair's price becomes vulnerable to manipulation as discussed in Section 3.1. Simultaneously, LPs are not incentivized to provide significant asset reserves when the trading volume of a pair is very low since fewer fees will be distributed to them. Therefore, a new design of a marketplace is needed where tokens with low trading volume also enjoy significant liquidity without incurring additional costs to the LPs to acquire more token reserves.

Tackling the above issue, this paper presents DEEPER, a design for a decentralized exchange that allows LPs of multiple tokens against a common currency to assemble and share their liquidity for that common currency. This allows LPs to achieve the objective of providing higher liquidity without acquiring additional token reserves. DEEPER extends a concentrated liquidity AMM[16] by adding the functionality of shared reserve for the common currency in a batch of multiple trading pairs. Instead of supplying individual currency liquidity for each token pair, LPs can create a single, deep pool of the currency reserve that can be potentially accessed by any token pair. However, the first pair whose price reduces gets to access the shared pool and the corresponding currency reserves are allocated to that pair. This is based on an algorithmic access mechanism that ensures that the currency reserves in the shared pool never go negative. At the same time, sharing currency reserves is optional for LPs and does not prevent them from providing concentrated liquidity for individual trading pairs. Unlike other multi-token pool platforms such as BALANCER,[17] DEEPER remains a sovereign AMM DEX. It does not depend on arbitrageurs to adjust the price of a token when a trade is made between other tokens in the pool. Lastly, the design of DEEPER is kept simple enough making it practical to implement on EVM-based smart contracts. To demonstrate this, we provide a template for a *hook contract*, which is a smart contract design methodology to extend execution functionalities in UNISWAP V4.[18] The template hook contract provisions shared liquidity on top of a concentrated liquidity AMM DEX. This paper is an extension of our previous work.[19]

To evaluate the added benefits and potential limitations of our design, we perform experiments on historic price movements of low trading volume tokens. The results show that DEEPER can enhance the active liquidity for a trading pair, on an average, by a factor of up to 5.9× when using a batch of eight assets. In doing so, LPs do not need to provide any additional reserves of tokens compared to the contemporary AMM design. At the same time, our model only requires between 16.8% and 24.4% of the amount of currency used in the traditional design to achieve similar levels of liquidity. However, the liquidity enhancement reduces when the liquidity position is not altered for extended periods of time. We also study the relationship between the average price gain of a trading pair and its liquidity enhancement. Results show that token pairs with large price drops can consume a significant portion of currency

reserves, thus leaving small room for liquidity enhancement in the other trading pairs within the batch. Although the increase in liquidity can be enhanced further by increasing the number of pooled tokens in a batch, LPs need to be cautious while picking the tokens for a batch. These pitfalls give room for further developments and improvements in our design.

The paper is organized as follows: Section 2 gives preliminaries on AMMs and drawbacks of low liquidity, Section 3 describes problem statement and gives an overview of the solution, Section 4 presents the design of the DEEPER DEX including a hook template and comments on divergence loss for LPs, Section 5 describes the experimental setup and summary of results, Section 6 elaborates on the applications and risks of our design, Section 7 discusses related work in this field, and lastly, the paper is concluded in Section 8.

2 | BACKGROUND

2.1 | Constant product automated market makers

Constant product automated market maker (CPAMM) was the first successful algorithmic market maker introduced by the UNISWAP DEX,[20] which is governed by the *constant product formula*. In a nutshell, consider a trading pair consisting of tokens $\mathcal{T}_a$ and $\mathcal{T}_b$ such that the price of $\mathcal{T}_a$ with respect to $\mathcal{T}_b$ is p. Then, the liquidity pool of the above pair has active token reserves as a function of price comprising $r_a(p)$ and $r_b(p)$ units of $\mathcal{T}_a$ and $\mathcal{T}_b$, respectively, such that $r_a(p)r_b(p) = L^2$, where L does not depend on p. The constant L is called the *liquidity* of the pool. The marginal price p, that is, the price for an infinitesimally small trade, can be derived as

$$p = -\frac{\mathrm{d}r_b}{\mathrm{d}r_a} = -\frac{\mathrm{d}}{\mathrm{d}r_a}\left(\frac{L^2}{r_a}\right) = \frac{L^2}{r_a^2} = \frac{r_b}{r_a} \tag{1}$$

This can be interpreted as the equivalent amount of $\mathcal{T}_b$ per unit amount of $\mathcal{T}_a$ in the reserves. The expression for token reserves can therefore be derived as follows:

$$\begin{aligned} r_a(p) &= \sqrt{r_a(p)r_b(p)}\sqrt{\frac{r_a(p)}{r_b(p)}} = \frac{L}{\sqrt{p}} \\ r_b(p) &= \sqrt{r_a(p)r_b(p)}\sqrt{\frac{r_b(p)}{r_a(p)}} = L\sqrt{p} \end{aligned} \tag{2}$$

When a trader swaps Δr_b units of $\mathcal{T}_b$ at price p, then $(1-\mu)\Delta r_b$ is collected as *liquidity fees* for LPs, where $\mu \in [0,1]$ and is set close to 1. In return, the trader receives Δr_a units of $\mathcal{T}_a$ following the constant product rule, that is,

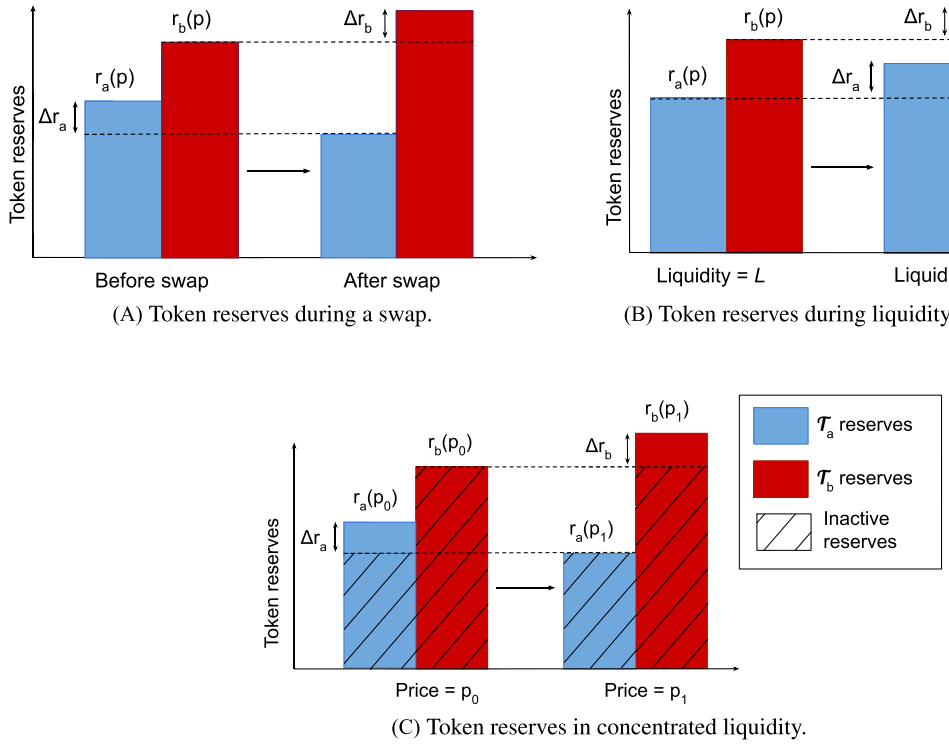
$$(r_b(p) + \mu\Delta r_b)(r_a(p) - \Delta r_a) = L^2 \tag{3}$$

An illustration of token reserves before and after the swap for $\mu = 1$ is shown in Figure 1A.

Furthermore, LPs can alter liquidity by adding or removing token reserves. If an LP provides Δr_a and Δr_b of $\mathcal{T}_a$ and $\mathcal{T}_b$ respectively at price p, then the following needs to hold:

$$\frac{\Delta r_a}{\Delta r_b} = \frac{r_a(p)}{r_b(p)} \tag{4}$$

Here, Δr_a and Δr_b should either be both positive (deposit) or negative (withdraw). The new liquidity L' can now be calculated as $(r_a(p) + \Delta r_a)(r_b(p) + \Delta r_b) = (L')^2$. Figure 1B gives an illustration of token reserves when liquidity increases in the pool. Although in CPAMM liquidity does not change with the price of the tokens, this is not the case in concentrated liquidity AMM as presented next.



(A) Token reserves during a swap.

(B) Token reserves during liquidity provision.

(C) Token reserves in concentrated liquidity.

FIGURE 1 An illustration of token reserves in automated market makers during swap and liquidity provision.

2.2 | Concentrated liquidity AMM

Consider Figure 1C where the price of $\mathcal{T}_a$ increases from p_0 to p_1, $\mathcal{T}_a$ reserves reduce by Δr_a while $\mathcal{T}_b$ reserves increase by Δr_b. In this price interval, only Δr_a of $\mathcal{T}_a$ and Δr_b of $\mathcal{T}_b$ are actively swapped while the rest of the reserves (marked with dashes) remain inactive. This is the key idea behind concentrated liquidity AMMs (CLAMMs) where LPs can provide liquidity in a price interval $[p_0, p_1]$ by only supplying Δr_a units of $\mathcal{T}_a$ and Δr_b units of $\mathcal{T}_b$. Meanwhile, the liquidity in CLAMM is the same as CPAMM, that is, $L^2 = r_a(p) r_b(p)$. When the price is outside the above price interval, the liquidity becomes inactive. Hence in a CLAMM, $r_a(p), r_b(p)$ are called *virtual reserves*. The real reserves of $\mathcal{T}_a, \mathcal{T}_b$ at price $p \in [p_0, p_1]$, denoted by $r'_a(p), r'_b(p)$, can therefore be calculated in terms of virtual reserves as follows:

$$\begin{aligned} r'_a(p) &= r_a(p) - r_a(p_1) = L\left(\frac{1}{\sqrt{p}} - \frac{1}{\sqrt{p_1}}\right) \\ r'_b(p) &= r_b(p) - r_b(p_0) = L\left(\sqrt{p} - \sqrt{p_0}\right) \end{aligned} \tag{5}$$

We used the result from Equation (2) to derive the expressions for virtual reserves in terms of liquidity and price. Observe that at the boundaries of the interval, that is, $p \in \{p_0, p_1\}$, the real reserves consist of either only $\mathcal{T}_a$ or $\mathcal{T}_b$. Alternatively, one can also write the constant product equation in terms of real reserves $r'_a(p), r'_b(p)$ as follows:

$$(r'_a(p) + r_a(p_1))(r'_b(p) + r_b(p_0)) = L^2 \tag{6}$$

As illustrated in Figure 2, Equation (6) (shown in solid orange) represents a translation of the original constant product curve (shown in dashed green).

L remains constant within a price interval but can vary across intervals. Concentrated liquidity, therefore, allows LPs to add arbitrary liquidity in different price intervals. Figure 3 shows an example liquidity profile by an LP across price intervals. The liquidity at price p is denoted by $L(p)$. Figure 4 presents the corresponding real reserves provided for each price interval. The active price interval is marked with circled ticks and consists of both $\mathcal{T}_a$ and $\mathcal{T}_b$ reserves. The inactive price intervals consist of only $\mathcal{T}_a$ or $\mathcal{T}_b$ for prices higher or lower than the current price.

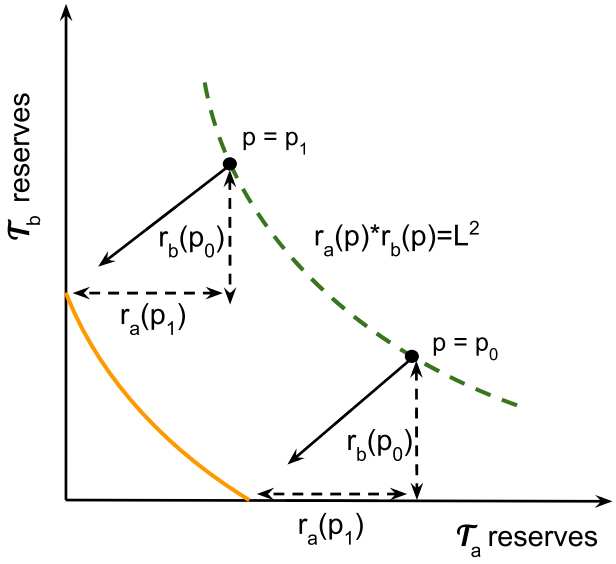


FIGURE 2 An illustration of the constant product curve in a price interval of a concentrated liquidity AMM.

FIGURE 3 An example liquidity distribution during a given interval.

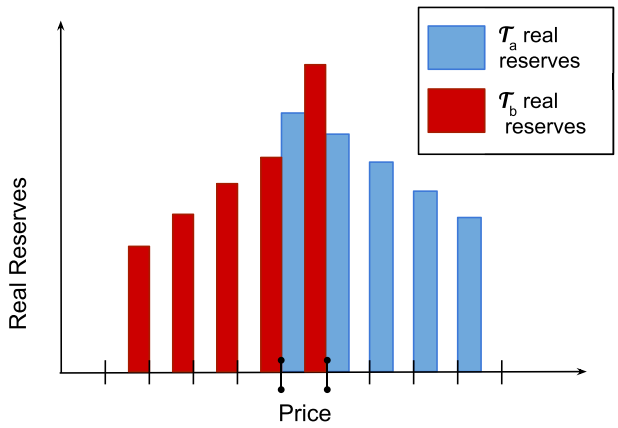


FIGURE 4 Illustration of real reserves distribution of an LP in concentrated liquidity DEX.

2.3 | Slippage

Slippage in an AMM is defined as the deviation between the realized price of a trade and the marginal price. As before, if a trader swaps Δr_b units of $\mathcal{T}_b$ in exchange for Δr_a units of $\mathcal{T}_a$ at the initial exchange price of p, then the slippage of this trade is defined as

$$\mathcal{S}(\Delta r_b, r_a(p), r_b(p)) = \frac{\Delta r_b/\Delta r_a}{p} - 1 \tag{7}$$

For a CLAMM, ignoring trading fees and assuming that the price does not cross its tick during the swap, if the new marginal price becomes p', then $\frac{\Delta r_b}{\Delta r_a}$, which is the execution price of the trade turns out to be $\sqrt{pp'}$. Thus, the above expression for slippage can be written as

$$\frac{\sqrt{pp'}}{p} - 1 = \frac{\sqrt{p'} - \sqrt{p}}{\sqrt{p}} = \frac{L\left(\sqrt{p'} - \sqrt{p}\right)}{L\sqrt{p}} = \frac{\Delta r_b}{L\sqrt{p}} \tag{8}$$

Thus, given a trade size Δr_b and an initial AMM price p, the slippage is lower for a higher value of virtual liquidity L and vice-versa. In other words, a pool with lower liquidity requires a smaller trade amount to cause the same amount of slippage.

3 | WORK MOTIVATION

In this section, we explain the undesirable consequences caused by low liquidity and low volume of a trading pair. We then give an overview of our solution and how it alleviates the problem of low liquidity.

3.1 | Potential consequences of low liquidity: sandwich attacks

Low liquidity and high slippage pave the way for *sandwich attacks* in which an attacker manipulates the AMM price due to slippage and ends up extracting value. On the other hand, the victim experiences a worse execution price. The attack procedure, as given in Figure 5, occurs as follows: Suppose $User_1$ wants to purchase $\Delta r_{a,1}$ units of $\mathcal{T}_a$ (transaction $k+1$ in the figure) whose initial marginal price is p with respect to $\mathcal{T}_b$ on an AMM. $User_2$ observes the transaction waiting in the public mempool. $User_2$ then sends two transactions right before and after the $User_1$'s transaction (transactions $k, k+2$, respectively). The former transaction buys $\Delta r_{a,2}$ units of $\mathcal{T}_a$ while the latter transaction sells $\Delta r_{a,2}$ units of $\mathcal{T}_a$. Let the execution price for the first and second transaction be p_2, p_1 respectively, while the new marginal price is p', then $p' > p_1 > p_2 > p$ due to slippage. Let the execution price of $User_2$ in the third transaction be p'_2, then $p' > p'_2 > p_2$ due to slippage. Thus, $User_2$ (the attacker) is profitable if the profit from the first and third transactions exceeds their gas cost while $User_1$ (the victim) gets a worse execution price compared to a situation with no sandwich transactions.

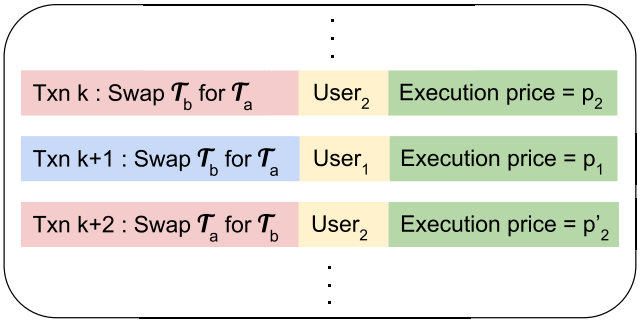


FIGURE 5 Sequence of transactions during sandwich attacks.

As discussed before, since the capital required to cause slippage is low in pools with lower liquidity, lower liquidity pools are more vulnerable to such sandwich attacks. Thus, fair marketplaces should ideally have high liquidity for trading pairs.

3.2 | Capital efficiency and market fairness during low trading volume

As mentioned in Equation (3), the revenue of an LP comes from the trading fee that is charged as a constant fraction of the tokens swapped. Problems arise when the trading volume of a token pair $\mathcal{T}/\mathcal{T}_c$ ($\mathcal{T}_c$ is a highly liquid currency) is orders of magnitude lower than other trading pairs. This can be caused due to the following unavoidable reasons:

- $\mathcal{T}$ is a newly launched token and its underlying utility has not gained traction among users;
- $\mathcal{T}$ has a seasonal utility (e.g., a DAO token used for voting in a protocol,[21] an access token for a real-world event,[8] or a football club fan token);
- the overall market has low liquidity due to the high cost of acquiring capital, that is, high borrowing rates;
- $\mathcal{T}$ targets a small niche of users (e.g., $\mathcal{T}$ represents an LP token of a UNISWAP V2 pool).

One or more of the above conditions leads to a lower trading volume of the $\mathcal{T}/\mathcal{T}_c$ pair that causes the following cascading consequences:

1. As lower trading volume leads to lower LP fees, this reduces the incentive for an LP to participate. Moreover, this exposes them to the risk of an overall loss if their impermanent loss (as discussed later in Section 4.4) dominates the trading fees. If the trading fee is set high enough to increase the LP's incentives, it discourages the token users and traders including arbitrageurs from trading $\mathcal{T}$.
2. If the $\mathcal{T}/\mathcal{T}_c$ pair ends up with low liquidity, then it is subject to unfavorable economic events of high volatility and high slippage. As derived earlier, lower liquidity makes a trading pair vulnerable to price manipulations since low capital is now required to manipulate reserve ratios and hence the prices. This opens room for DeFi attacks including sandwich attacks as discussed in Section 3.1. These consequences can seriously damage the integrity of a platform whose operation relies on the fairness of the token price, for example, when $\mathcal{T}$ represents a DAO or a voting token.
3. Despite their exposure to impermanent loss, if an LP provides deep liquidity to a low-volume trading pair, this approach is not capital efficient. This is because the liquidity capital stays idle for the majority of the time and suffers opportunity costs.

Expanding on the last point, with the explosion of DeFi and other ecosystems in blockchain and the reduced trust in CEXs, it has become crucial for on-chain DEXs to sustain a fair market for thousands of token pairs.[22] Even if a token observes low trading volume, this does not imply a lower significance of the token itself. Thus, it is important for its market price to remain fair. This requires a mechanism that enhances the liquidity profile while consuming low input capital, that is, real reserves so that it does not hurt the LPs economically. Furthermore, the solution design should be easily implementable on common blockchain environments such as the Ethereum virtual machine or EVM.

3.3 | Evaluation metric

In this paper, we consider N trading pairs of tokens $\mathcal{T}_0, \mathcal{T}_1 ... \mathcal{T}_{N-1}$ against a digital currency $\mathcal{T}_c$ with high liquidity (e.g., ETH). We profile the liquidity on an AMM and the price of these trading pairs in a constant time interval I. Such a time interval gives a finite range of prices that are processed. The price of the token at time t is denoted by $p(t)$. The price of each $\mathcal{T}_i$ is divided into intervals of uniform width. We assume that during I, no LP mints or withdraws their liquidity.

Let $L_i(p(t))$ denote the liquidity profile for $\mathcal{T}_i$ at price $p(t)$ at a given instance t. At $t = 0$, the total real reserves of $\mathcal{T}_i, \mathcal{T}_c$ for the pair i are denoted by $\mathbf{r'_i}, \mathbf{r'_{i,c}}$, respectively. Suppose there are two LPs with their proposed liquidity profiles $L_i(p)$ and $L'_i(p)$ with respect to the price for each trading pair. Then for each pair, we compare the two profiles by calculating the metric z_i as follows:

$$I \cdot z_i = \int_0^I \frac{L_i(p(t))}{L_i'(p(t))} dt \tag{9}$$

The above metric informs, on average, the enhancement in experienced liquidity when the liquidity profile is $L(p)$ in comparison to $L'(p)$ for a trading pair i. The total initial real reserves for $\mathcal{T}_c$ provided collectively by the LPs equals:

$$\mathbf{r'_{total,c}} = \sum_{\mathbf{i=0}}^{\mathbf{N-1}} \mathbf{r'_{i,c}} \tag{10}$$

Formally, our objective is to maximize the average increase in liquidity z_i for each trading pair without changing the total initial currency reserves $\mathbf{r'_{total,c}}$ and token reserves $\mathbf{r'_i}$ for $\mathcal{T}_i$.

3.4 | DEEPER overview

DEEPER is a DEX design that allows LPs of different trading pairs against a common currency to gather and pool their currencies to enhance the available liquidity of each of the trading pairs. Given a batch of N trading pairs $\mathcal{T}_i/\mathcal{T}_c$ on DEEPER, each pair gets real $\mathcal{T}_c$ reserve for liquidity provision in one of the two ways:

1. Individual reserves for concentrated liquidity in each price range (same as in CLAMM).
2. Shared reserves where an LP provides initial liquidity profiles for the inactive price intervals of each pair and deposits a lump sum $\mathcal{T}_c$ in the shared reserves pool accessible by all pairs.

Let R denote the amount of $\mathcal{T}_c$ in the shared pool at a given instance. During a given time interval, if the price of $\mathcal{T}_i$ increases activating a thus far inactive higher price interval that consists of only $\mathcal{T}_i$, then the accumulated $\mathcal{T}_c$ from the recently inactivated interval is added to the shared reserves pool and R increases. Likewise, if the price of $\mathcal{T}_i$ decreases such that an inactive lower price interval requiring only $\mathcal{T}_c$ becomes active, then those tokens are withdrawn from R and allocated to the newly activated interval. The reserves allocation from the shared pool is designed such that the shared reserve pool never goes negative. In the unlikely event of reducing $\mathcal{T}_i$ prices leading to a dried-up shared reserves pool, the individual concentrated liquidity provision per token pair starts to dominate. This serves as a fail-safe mechanism when the price crash of one trading pair consumes significant shared reserves.

4 | PROTOCOL DESIGN

DEEPER is a CLAMM-based DEX design that allows LPs of several trading pairs with a common currency to come together and share their currency reserves. This, however, does not prevent an LP from providing individual liquidity to just one pair. Thus, for each asset pair $\mathcal{T}_i/\mathcal{T}_c$, we define two kinds of liquidity provisions: *shared* and *individual*. The shared liquidity provision is explained below.

4.1 | Shared liquidity provision

The total available shared reserves of $\mathcal{T}_c$ represented by R is split between *busy* reserves or R^b and *available* reserves or R^a so that $R = R^a + R^b$.

The virtual liquidity profile, for shared liquidity provision, of a trading pair i is divided into intervals of prices of uniform width. Each interval is mapped to an integer tick such that if an interval $[p_0, p_1)$ has tick k, then $p[k] = p_0$. Similarly, $L_i[k]$ and $r'_{i,c}[k]$ represent the liquidity and the corresponding $\mathcal{T}_c$ reserves (either active or inactive) in tick k at a given instance. Let K_i be the tick of the currently activated interval with $\mathbf{K_i}$ being its value at the beginning, that is, $t = 0$ for each trading pair $\mathcal{T}_i/\mathcal{T}_c$. An illustration of the ticks and token notation is presented in Figure 6.

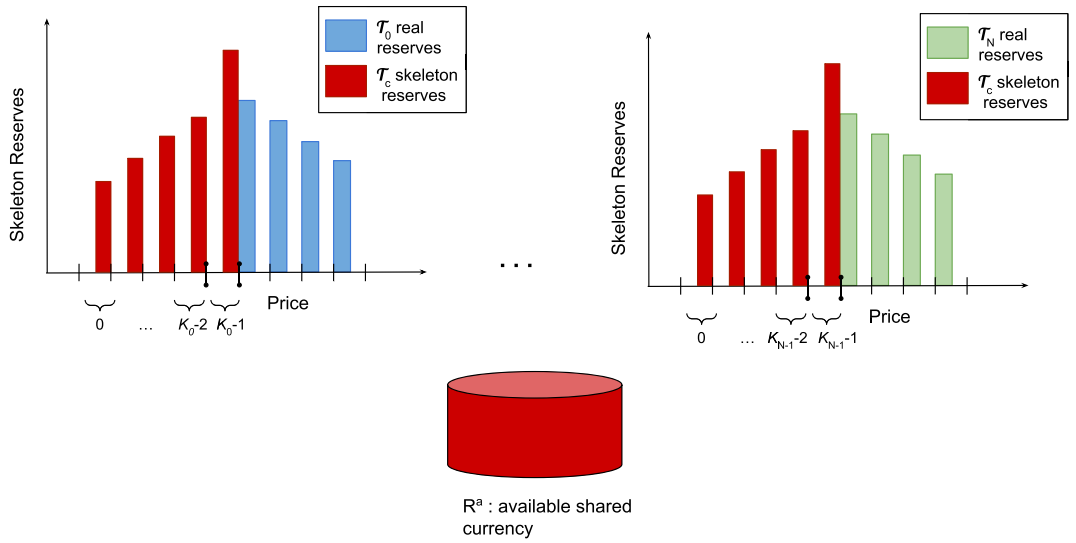


FIGURE 6 Illustration of skeleton liquidity, notation for tokens and ticks, and available reserves in the DEEPER protocol.

Liquidity provision using shared $\mathcal{T}_c$ reserves consists of the following specifications:

1. To provide shared $\mathcal{T}_c$ at $t=0$, LPs need to (i) provide liquidity profile for each trading pair i, that is, $L_i[k]\ \forall k \le \mathbf{K_i}$, and (ii) deposit $\mathbf{r'_{i,c}}$ units of $\mathcal{T}_c$ reserves in accordance with the above liquidity profile for each i. For shared provision, we call L_i the *skeleton* liquidity profile because it will be used to calculate the actual liquidity of a price interval. Similarly, $\mathbf{r'_{i,c}}$ are called the skeleton $\mathcal{T}_c$ reserves. Since sharing LPs can only provide $\mathcal{T}_c$ reserves, the skeleton liquidity is positive in price ticks that are less than the activated ticks and zero elsewhere. The skeleton liquidity in each interval remains constant unless some LP mints or burns their shared liquidity.
2. The actual liquidity of pair i in the active price interval is denoted by $\mathcal{L}_i$, and it remains constant when the price of $\mathcal{T}_i$ lies within the active price interval. Swaps that do not change the price interval are executed based on CLAMM with $\mathcal{L}_i$ as the virtual liquidity of the active interval.
3. When the price of $\mathcal{T}_i$ decreases such that the tick transitions from K_i to K_i-1 with the new interval having $\mathbf{r'_{i,c}}[K_i-1]$ skeleton $\mathcal{T}_c$ reserves, the following events occur:
   - The new active interval secures $r'_{i,c}[K_i-1]$ units of $\mathcal{T}_c$ from the available reserves pool such that

$$r'_{i,c}[K_i-1] = R^a\left(\frac{\mathbf{r'_{i,c}}[K_i-1]}{\sum_{j=0}^{K_i-1}\mathbf{r'_{i,c}}[j]}\right) \tag{11}$$

   This is secured by reducing the available reserves pool and increasing the busy reserves pool. Since $r'_{i,c}[K_i-1]$ is always a fraction of the available reserves R^a, it never goes negative after the operation. The actual liquidity can be derived from real reserves using the relation in Equation (5).
   - The token $\mathcal{T}_i$ in the tick K_i becomes inactive with its amount stored in memory and it reactivates in the future when the tick transitions from K_i-1 to K_i as discussed next.
4. When the price of $\mathcal{T}_i$ increases and the tick crosses from K_i to K_i+1, then the accumulated $\mathcal{T}_c$ in the newly inactive tick K_i is transferred from busy reserves to available reserves and any $\mathcal{T}_i$ reserves stored in the memory of tick K_i+1 are released in the tick.

Lastly, if the price of a pair $\mathcal{T}_i$ goes all the way down to tick 0, then this pair consumes all available shared $\mathcal{T}_c$. In such a case, individual reserves become dominant. The shared reserves, however, are restored when the price of this pair starts to increase. Therefore, the available shared $\mathcal{T}_c$ can be potentially secured by any trading pair, and the first one to access it secures $\mathcal{T}_c$ while the quota per price interval of every other pair reduces.

4.2 | Shared liquidity withdrawal and LP fees

LPs can withdraw their shared $\mathcal{T}_c$ reserves at any point. In doing so, they receive the proportion of available reserves and any inactive $\mathcal{T}_i$ that belongs to them, and the skeleton liquidity provided by them is removed for each pair. Further, any trading fee that is accrued in a price interval is distributed among the LPs in proportion to their skeleton liquidity.

4.3 | Individual liquidity

An LP is free to provide individual liquidity in any price range by providing $\mathcal{T}_i$ or $\mathcal{T}_c$ or both depending on the active state of the interval of interest. Since they cannot provide shared reserves for $\mathcal{T}_i$ for liquidity, individual liquidity is the only way to serve this purpose.

4.4 | Divergence loss for shared liquidity providers

Divergence loss is defined as the opportunity cost for an LP to provide token reserves as liquidity compared to just holding them. In CPAMM and CLAMM, given an initial liquidity profile (e.g., Figure 3), the divergence loss is a function of the token price.[23] Since an LP can recover any accrued losses when $\mathcal{T}_i$ trades back at the initial price (when liquidity was provided), divergence loss is not permanent. Therefore, it is also referred to as *impermanent loss*. In the case of DEEPER, however, the divergence loss is not a function of just the token price, because the total $\mathcal{T}_c$ reserves owned by an LP depend on the relative order of securing $\mathcal{T}_c$ from the shared reserves pool by the trading pairs. In other words, divergence loss is price path dependent.

However, the divergence loss *still* remains impermanent for an LP. This is described in the lemma below:

Lemma 1. The divergence loss of LPs providing shared reserves for $\mathcal{T}_c$ at price p_i for a token $\mathcal{T}_i$ with respect to $\mathcal{T}_c$ at $t = 0$ and subsequently withdrawing their liquidity at the same initial price for each token is zero and independent of any intermediary price movements.

Proof. Suppose LPs provide a total R reserve of shared $\mathcal{T}_c$ and a skeleton liquidity profile L_i at an initial price $\mathbf{p}_i$ for each pair i. Then, this profile is defined for the ticks less than the current active tick K_i for each pair. We prove that the shared reserve for $\mathcal{T}_c$ equals R when the price of $\mathcal{T}_i$ becomes $\mathbf{p}_i$ for all i.

When the current tick transitions from k to $k-1$ and secures $\mathcal{T}_c$ from the shared pool, the amount of shared reserves secured by tick $k-1$ is stored in its memory which serves as its history. Eventually, when the tick transitions back from $k-1$ to k, the shared $\mathcal{T}_c$ that was withdrawn earlier is added back to the shared pool.

Therefore, when the final prices become the same as the initial prices, the initial and final active ticks become the same for all the pairs. Therefore, any borrowed $\mathcal{T}_c$ from the shared reserve pool is released back. The LPs can now withdraw exactly the initially supplied $\mathcal{T}_c$ from the shared pool. Any $\mathcal{T}_c$ or $\mathcal{T}_i$ that was supplied as part of individual liquidity remains the same since individual liquidity reserves are a function of token prices. Since the LP can withdraw exactly the same amount of reserves that they supplied initially and at the same initial price, the total divergence loss suffered is zero. □ ■

4.5 | Smart contract friendliness and hook template

A smart contract[24] is a Turing-complete program that is deployed on a blockchain. A blockchain account can asynchronously trigger functions in these programs by paying gas fees that are charged per contract operation during a call. Therefore, an ideal smart contract design performs the minimal amount of updates in the state (i.e., variables) of the contract during a function call.

Since DEEPER is an extension of a concentrated liquidity DEX, the protocol can be implemented as an augmentation to the UNISWAP DEX using their *hook* design, which is provisioned in UNISWAP V4.[18] A hook is a third-party smart

contract with developer-defined logic that is called by the core DEX contract during its call execution cycle. Developers can create custom functions inside the hook contract to add functionalities in the call of the original contract. We give a template below for a hook design that modifies the `beforeSwap()` and `afterSwap()` callback functions. These hook functions are called right before and after the `swap()` function is executed as illustrated in Figure 7. A hook callback function returns back to the core contract after completing its execution. For a batch of N trading pairs, a common hook contract is used for each pool $\mathcal{T}_i/\mathcal{T}_c$ as explained below:

- Core contract: This is the concentrated liquidity contract in UNISWAP V4 that maintains pools for the trading pairs $\mathcal{T}_i/\mathcal{T}_c$ and performs swaps in each of them. `swap()` and `modifyPosition()` are external functions of this contract with the former performing token swap and the latter allowing LPs to add or remove their liquidity positions. The hook callback functions `beforeSwap()` and `afterSwap()` are executed before and after the `swap()` function and return back to the core contract after completing execution.
- DEEPER hook contract: This contract manages the shared reserves for a batch and owns the shared reserves in the inactive ticks, that is, $r'_i[j]\ \forall j > K_i$ and $r'_{i,c}[j]\ \forall j < K_i\ \forall i \in [0, N-1]$ at all times. At the same time, the contract provisions the assets for the active tick inside the core DEX contract and is the sole owner of this shared LP position. The hook contract keeps track of the cumulative skeleton $\mathcal{T}_c$ below the active tick for each i and total available $\mathcal{T}_c$ or R^a. This allows it to calculate the $r'_{i,c}$ for a range using Equation (11) when the price decreases and a tick is crossed. The actual liquidity of the current tick is calculated using the linear relationship of L and $r'_{i,c}$ at the interval edge as shown in Equation (5). The amount of $\mathcal{T}_i$ is stored for the ticks above the active tick.
- `addSharedLiquidity()`: This is an external function of the hook contract that is called by an LP to provide shared liquidity. In doing so, an LP provides a skeleton liquidity profile and deposits the corresponding token reserves in the hook contract for each pool in the batch. If the skeleton liquidity includes the active tick, the hook contract deposits the token reserves in the core contract using the `modifyPosition()` function.
- `removeSharedLiquidity()`: This is an external function of the hook contract that is called by an LP to exit their shared liquidity. In doing so, the hook contract removes the caller's position in the active tick from the core contract using the `modifyPosition()` function. Then, the caller receives their token reserves from both active and inactive (owned by the hook contract) intervals along with the accrued fees, if any.
- `beforeSwap()`: This callback function is a part of the hook which is called before a swap is executed. If the swap is exchanging $\mathcal{T}_i$ for $\mathcal{T}_c$, it first determines the number of ticks decreased and adds liquidity using the `modifyPosition()` external function. Similarly, if the swap is exchanging $\mathcal{T}_c$ for $\mathcal{T}_i$, it first determines the number of ticks increased and adds any previously accumulated $\mathcal{T}_i$ reserves in the higher ticks.
- `afterSwap()`: If the swap increases(decreases) the active tick, this callback function removes $\mathcal{T}_c(\mathcal{T}_i)$ as part of the inactive liquidity using `modifyPosition()` and updates R^a. It also performs fee accounting to manage fee distribution among the sharing LPs and $\mathcal{T}_i$ accounting to update the assets in the inactive ticks.
- Individual liquidity: LPs willing to provide individual liquidity can do so by using the `modifyPosition()`, that is, providing liquidity directly to the core contract instead of providing via the hook contract.

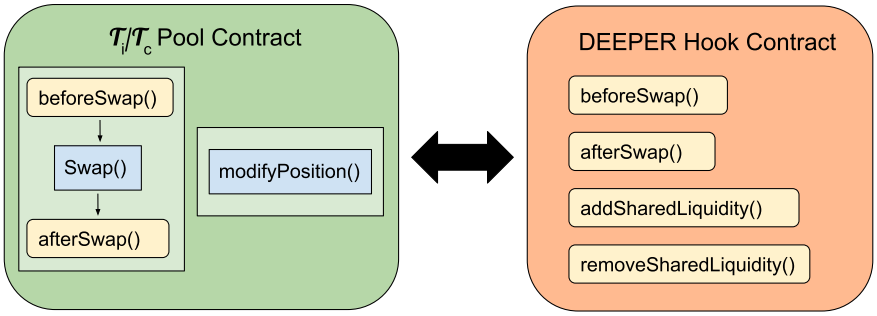


FIGURE 7 An illustration of the hook template for shared liquidity.

5 | PROTOCOL EVALUATION

In this section, we evaluate the benefits of the DEEPER DEX and study the parameters that optimize them. Our experiments attempt to answer the following questions:

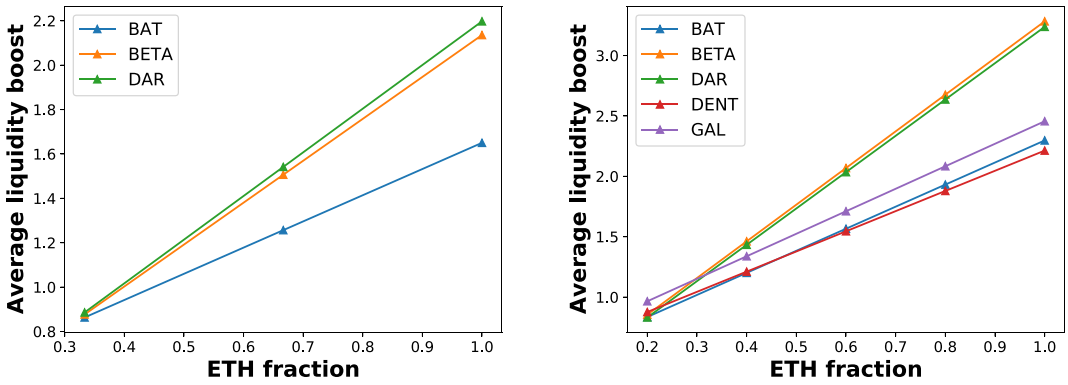
1. What is the average increase in the experienced liquidity of our shared liquidity model compared to the individual liquidity model?
2. What is the relationship between the shared liquidity boost and the number of trading pairs in a shared batch?
3. How can LPs optimize their enhanced liquidity by varying I?
4. What is the relationship between the price drop of a trading pair and the amount of shared reserves it consumes?

5.1 | Methodology

For the purposes of this evaluation, we use the historical price data from the month of December 2022 of trading pairs from the UNISWAP V3 DEX for the following tokens: Basic Attention Token (BAT),[25] Beta Finance (BETA),[26] Mines of Dalarnia (DAR),[27] Dent Wireless (DENT),[28] Galxe (GAL),[29] Holo Chain (HOT),[30] OMG Network (OMG),[31] and Smooth Love Potion (SLP).[32] Each of the trading pairs exhibits the following two properties: (i) They are traded against Wrapped ETH and (ii) had an average daily volume between \$0–50 k. Such a trading pair with a trading fee of 0.3% generates a total revenue of \$0–150 per day for all the LPs collectively. We create three batches each containing three, five, and eight trading pairs. We use three time periods of 1 day, 7 days, and 14 days for I during which the skeleton liquidity profiles remain constant. For a given token batch and time period I, we simulate the liquidity environment by initializing liquidity between the minimum and maximum price during that month. For the shared ETH, we input the skeleton liquidity profile and the corresponding ETH reserves for each pair. The skeleton liquidity peaks at the start price and decays slightly from there. For prices above the start price, we initialize individual liquidity by providing the asset token in each price interval. This liquidity decreases as price increases similar to ETH. We also create, for comparison, a model where ETH is provided via individual liquidity with equal ETH allocation to each pair.

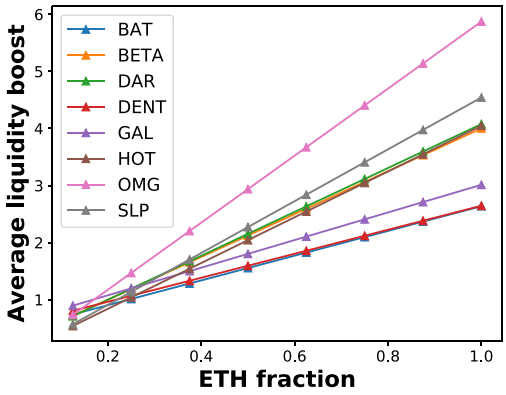
5.2 | Summary of results

1. **Liquidity boost:** Figure 8A shows the average increase in liquidity (z in Equation (9)) compared to the individual liquidity provision for a batch consisting of three trading pairs and I set to 1 day. The x-axis represents the total amount of initial ETH deposited as a fraction of the ETH used in the individual liquidity provision. The graph is plotted for three values of ETH fractions: $\frac{1}{3}$, $\frac{2}{3}$, 1. We can observe that the shared liquidity is more than 80% of its individual counterpart while costing only a third of ETH reserves. Moreover, the amount of initial ETH required reduces by 55.1%, 60.2%, and 60.9% for the three trading pairs, respectively, to achieve the same average liquidity as in the individual model. When the initial ETH reserves are increased so as to consume the same ETH as the individual model, the experienced liquidity increases by 1.6×, 2.1×, and 2.2× for the three trading pairs, respectively. Therefore, DEEPER DEX significantly increases liquidity without consuming any surplus asset reserves.
2. **Batch size:** Figure 8B,C illustrates the liquidity increase with respect to the ETH reserves used for a batch of size 5 and 8 assets, respectively. For these batches, the cost of initial ETH reduces by 70.0%–78.2% and 75.6%–83.2%, respectively, to achieve the same liquidity as the contemporary model. The corresponding average liquidity increase is between 2.2–3.3 and 2.6–5.9, respectively. This shows that as more trading pairs pool their ETH, the liquidity per pair increases while the cost of ETH to achieve similar levels of liquidity decreases.
3. **Variation with I**: Figure 9 shows the liquidity boost averaged over all the trading pairs in a batch of five token pairs for different values of I, that is,1 day, 7 days, and 14 days, respectively. The key observation here is that the liquidity boost decreases over longer values for I. This is because when a trading pair consumes the available ETH, there is less ETH left in the shared pool for other pairs. This becomes dominant in longer time intervals in a market scenario with decreasing prices. Therefore, LPs should either update their skeleton liquidity profiles more frequently or provide the skeleton liquidity over a longer price range to observe high shared liquidity enhancement.
4. **Liquidity boost versus price drop:** For this experiment, we simulate shared liquidity for multiple batches of two trading pairs where each of the eight trading pairs is paired with the remaining seven and I is set to 14 days to



(A) Batch of 3 assets.

(B) Batch of 5 assets.



(C) Batch of 8 assets.

FIGURE 8 Average liquidity boost for multiple batches. The x-axis represents the fraction of ETH compared to the individual liquidity provision.

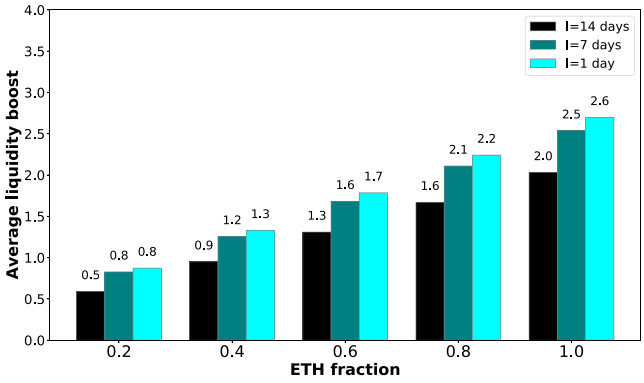


FIGURE 9 Liquidity increase for multiple values of I

represent a longer time period. Figure 10 shows the liquidity boost in the second pair plotted against the price drop in the first pair. On the other hand, Figure 11 shows the liquidity boost in the first pair plotted against the price drop in the first pair. To calculate the price drop, we take the logarithm of the ratio calculated by dividing the price at the start of the month and the mean price over 14 days. Figure 10 shows that as the price drop in the first pair increases, the liquidity boost in the second pair generally decreases. However, Figure 11 shows that a higher price drop in the first pair increases the same pair's liquidity boost. This aligns with the design of DEEPER where the trading pair with a falling price consumes available reserves. However, this can be detrimental in scenarios where the price of one pair reduces drastically in comparison to other pairs in the batch. For instance, pairs batched with BETA/WETH, which

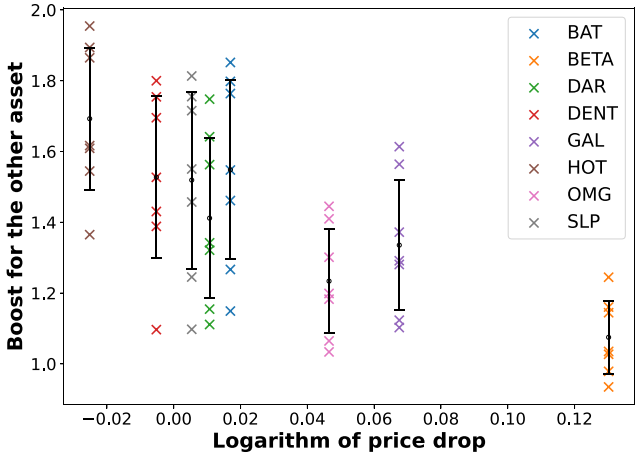


FIGURE 10 Second pair's liquidity increase versus price drop in the first pair.

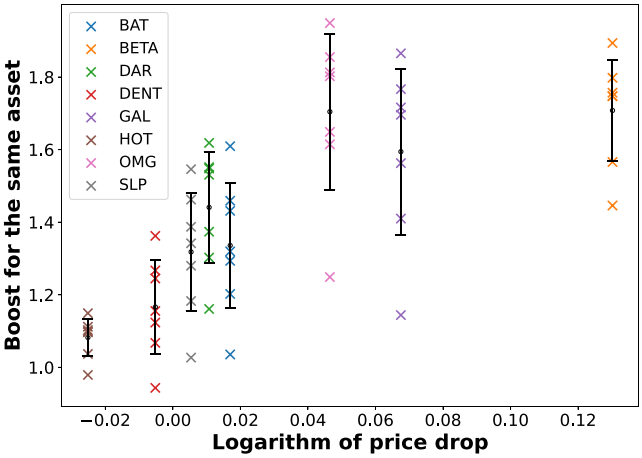


FIGURE 11 First pair's liquidity increase versus price drop in the first pair.

observed the highest logarithm price drop of 0.13, got a boost between 0.93 and 1.24 with a mean boost of 1.08. On the other hand, the BETA/WETH itself observed a boost between 1.44 and 1.89 with a mean boost of 1.71. Thus, trading pairs with large price drops can consume most of the available reserves while preventing a higher boost to other pairs in the batch.

6 | DISCUSSION

Since DEEPER uses the sharing of currency reserves to enhance the average active liquidity, it does not require additional capital from the LPs. Although, on one hand, the liquidity boost increases with the number of tokens in the batch, it comes at an added risk. As shown in Figures 10 and 11, a significant mean price drop in one token pair can lead to a skewed allocation of available reserves. The security of a token batch is thus limited by the most vulnerable token in the batch since a significant price drop in one token can consume most of the available reserves. The significant price drop can be due to numerous reasons including hacks or failure of the underlying token protocol. Thus, LPs should only include tokens in a batch that they believe are resilient and should generally avoid extremely large batch sizes or high-risk tokens.

One particular case where DEEPER adds significant value is when all the tokens in a batch stem from a single protocol while simultaneously having a distinct face value for each one of them. For example, platforms such as

CHILIZ[33] enable the creation of numerous football club fan tokens. Although each token uses the same underlying technology of the CHILIZ platform, they represent separate teams. Therefore, a batch of such tokens shares a similar level of security which allows the creation of larger batch sizes. Since there are many football clubs that are not popular and hence have low trading volume, DEEPER is an efficient way to create fair marketplaces for them.

Another benefit of our design is the reduction in vulnerability of low-volume trading pairs to high slippage and attacks leveraging it as explained in Section 3.1. Because DEEPER increases the average active liquidity of each trading pair by a significant factor, it increases the cost of executing sandwich attacks as well as reduces slippage for a trade of similar size compared to individual liquidity provision. Therefore, the benefit of uniting multiple low-volume tokens is reaped by each one of them in the batch. Further, it should be noted that the applications of DEEPER are two-fold. First, it serves as a platform to provide significant liquidity to token pairs with low trading volume. This is demonstrated by the high liquidity boost in Figure 8. Second, for token pairs with a high trading volume, it significantly reduces the amount of currency required to observe similar levels of liquidity. This is demonstrated by the reduction in the initial ETH required to observe similar levels of liquidity. In our results for a batch of three trading pairs, the initial ETH reduction was at least 55% for each trading pair. Therefore, this property can be leveraged by LPs to increase their capital efficiency for high-volume pairs. Lastly, although our experiments include ETH, which is a highly liquid token, as the common currency, DEEPER can also be used by having low-liquidity tokens as the common currency since the underlying algorithm does not assume the liquidity of the common currency.

7 | RELATED WORK

DEEPER extends the design of concentrated liquidity AMMs, especially UNISWAP V3. Although, to our knowledge, there are no platforms that enable liquidity sharing for currency tokens, certain platforms, as discussed below, have an implicit sharing mechanism for token liquidity. Lastly, our design is inspired by *Just in Time* liquidity provision, which is an adversarial attack to extract value from liquidity providers. In contrast, our design helps liquidity providers by increasing their capital efficiency.

7.1 | Multi-token automated market makers

Platforms such as BALANCER[34] and CURVE FINANCE[35] utilize a generalized invariant as opposed to the UNISWAP invariant which only works for a pair of tokens. This extends the AMM functionality by allowing the creation of multi-token pools with more than two tokens. A trader can then swap one token in exchange for another within the pool. Such a pool provides shared liquidity provision, for example, a pool consisting of $\mathcal{T}_a, \mathcal{T}_b$ and ETH can be seen as $\mathcal{T}_a$/ ETH, $\mathcal{T}_b$/ ETH with shared ETH liquidity provision. However, in such platforms, a transaction swapping one token impacts the price of other tokens. In the previous example, if a trader buys $\mathcal{T}_a$ in exchange for ETH, such a transaction also increases the price of $\mathcal{T}_b$ with respect to ETH. As a result, these platforms depend on arbitragers to maintain the price of the tokens after it changes during swaps. DEEPER, on the other hand, isolates the prices of the trading pairs in a batch, while at the same time allows sharing the ETH liquidity. Since DEEPER does not depend on arbitragers and other marketplaces to function, this makes it a sovereign platform.

7.2 | Just in time liquidity

CLAMM platforms such as UNISWAP V3 allow LPs to provide liquidity in a narrow range of a single price tick. This can be used by LPs to provide liquidity "just in time" (JIT) by front-running a swap transaction and then exiting their liquidity right after the transaction.[36] Such a provision is capital efficient for the JIT LPs since they are only supplying liquidity in the active price ranges while it is detrimental for passive LPs since they receive a lesser portion of fees.[37] DEEPER borrows from this scheme since liquidity from the available reserves is provisioned to a token pair that requires it (after its price reduces). This, however, is implemented as an algorithm and poses no cost in the form of transaction fees to LPs as opposed to JIT liquidity provision where LPs pay transaction fees.

8 | CONCLUSION AND FUTURE WORK

Tokens are a vital component of a well-functioning decentralized system. Therefore, it is important to establish efficient markets for such tokens. With the proliferation of innovation in the applications of blockchain and distributed ledger technology, the number of different kinds of tokens has increased substantially. One significant challenge to sustain fair markets for a variety of tokens is to address low liquidity provisions for trading pairs on decentralized exchanges. Lower liquidity is observed especially in token pairs with low trading volume resulting in unwanted price movements and manipulation attacks. On the other hand, low trading volume can be caused by various unavoidable factors and does not necessarily indicate lower significance for the token. This paper delves into this issue and presents DEEPER, a novel solution to enhance the average liquidity for a batch of low-volume trading pairs against a common currency. This is done via a reserve-sharing mechanism for the common currency, which does not incur additional costs to the liquidity providers of the trading pairs. The experimental results done on historic price data of low volume tokens using DEEPER show that reserve sharing significantly enhances average liquidity. Subsequently, we highlight the precautions that LPs should be aware of before providing liquidity in DEEPER. In conclusion, the shared reserve allocation mechanism and simple design make DEEPER a practical solution to the problem of shallow liquidity provision in low trading volume trading pairs.

Potential future work includes accounting for LP earnings from trading fees and impermanent loss while calculating the common currency allocation to a trading pair's pool.

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

ORCID


Srisht Fateh Singh https://orcid.org/0009-0008-7121-6171

REFERENCES

1. Nakamoto S. Bitcoin whitepaper. URL: https://bitcoin.org/bitcoin.pdf (17.07.2019); 2008.
2. Buterin V. Ethereum: a next-generation smart contract and decentralized application platform; 2014.
3. Werner SM, Perez D, Gudgeon L, Klages-Mundt A, Harz D, Knottenbelt WJ. SoK: Decentralized finance (DeFi). https://doi.org/10.48550/ARXIV.2101.08778; 2021.
4. Ding W, Hou J, Li J, et al. DeSci based on web3 and DAO: a comprehensive overview and reference model. *IEEE Trans Comput Social Syst*. 2022;9(5):1563-1573. https://doi.org/10.1109/tcss.2022.3204745
5. Proelss J, Sevigny S, Schweizer D. GameFi—the perfect symbiosis of blockchain, tokens, DeFi, and NFTs? *SSRN Electron J*. 2023. https://doi.org/10.2139/ssrn.4316073
6. Liu Z, Xiang Y, Shi J, et al. Make web3.0 connected. *IEEE Trans Dependable Secure Comput*. 2022;19(5):2965-2981. https://doi.org/10.1109/tdsc.2021.3079315
7. Dutta P, Choi T-M, Somani S, Butala R. Blockchain technology in supply chain operations: applications, challenges and research opportunities. *Transp Res Part E: Log Transp Rev*. 2020;142:102067. https://doi.org/10.1016/j.tre.2020.102067
8. Friends with benefits. https://www.fwb.help/events; 2023.
9. Ballandies MC. To incentivize or not: impact of blockchain-based cryptoeconomic tokens on human information sharing behavior. *IEEE Access*. 2022;10:74111-74130. https://doi.org/10.1109/access.2022.3189774
10. Williams S, Diordiiev V, Berman L, Uemlianin I. Arweave: a protocol for economically sustainable information permanence. arweave.org, Tech. Rep; 2019.
11. Chen Y. Blockchain tokens and the potential democratization of entrepreneurship and innovation. *Bus Horiz*. 2018;61(4):567-575.
12. Bellini E, Iraqi Y, Damiani E. Blockchain-based distributed trust and reputation management systems: a survey. *IEEE Access*. 2020;8:21127-21151.
13. Gupta A, Rathod J, Patel D, Bothra J, Shanbhag S, Bhalerao T. Tokenization of real estate using blockchain technology. *Applied Cryptography and Network Security Workshops: ACNS 2020 Satellite Workshops, AIBlock, AIHWS, AIoTS, Cloud S&P, SCI, SecMT, and SiMLA, Rome, Italy, October 19–22, 2020, Proceedings 18*: Springer; 2020:77-90.
14. Oliveira L, Zavolokina L, Bauer I, Schwabe G. To token or not to token: tools for understanding blockchain tokens. In: 39th International Conference on Information Systems; 2018.
15. Barbon A, Ranaldo A. On the quality of cryptocurrency markets: centralized versus decentralized exchanges. arXiv preprint arXiv:2112.07386; 2021.
16. Adams H, Zinsmeister N, Robinson D. Uniswap v2 core. https://uniswap.org/whitepaper-v3.pdf; 2020.
17. Ottina M, Steffensen PJ, Kristensen J. Balancer. *Automated Market Makers*: Apress; 2023:69-116.
18. Adams H, Salem M, Zinsmeister N, et al. Uniswap v4 core [draft]; 2023.

19. Singh SF, Michalopoulos P, Veneris A. Deeper: enhancing liquidity in concentrated liquidity AMM DEX via sharing. In: 2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC) IEEE; 2023:1-7.
20. Adams H, Zinsmeister N, Salem M, Keefer R, Robinson D. Uniswap v3 core. https://uniswap.org/whitepaper.pdf; 2020.
21. Liu L, Zhou S, Huang H, Zheng Z. From technology to society: an overview of blockchain-based DAO. *IEEE Open J Comput Soc*. 2021;2: 204-215. https://doi.org/10.1109/ojcs.2021.3072661
22. Morales AJ, Somin S, Altshuler Y, Pentland AS. User behavior and token adoption on ERC20. https://doi.org/10.48550/ARXIV.2005.12218; 2020.
23. Loesch S, Hindman N, Richardson MB, Welch N. Impermanent loss in uniswap v3; 2021.
24. Zheng Z, Xie S, Dai H-N, et al. An overview on smart contracts: challenges, advances and platforms. *Futur Gener Comput Syst*. 2020;105: 475-491. https://doi.org/10.1016/j.future.2019.12.019
25. Basic Attention Token. https://basicattentiontoken.org/; 2023.
26. Beta Finance. https://www.betafinance.org/; 2023.
27. Mines of Dalarnia. https://www.minesofdalarnia.com/; 2023.
28. Dent Wireless. https://www.dentwireless.com/; 2023.
29. Galxe. https://galxe.com/; 2023.
30. Holo Chain. https://www.holochain.org/; 2023.
31. OMG Network. https://omg.network/; 2023.
32. Smooth Love Potion. https://axieinfinity.com/; 2023.
33. Chiliz. Chiliz: a digital currency for sports and esports, entertainment platforms, adoptable universally across other industries; 2018.
34. Martinelli F, Mushegian N. A non-custodial portfolio manager, liquidity provider, and price sensor; 2019.
35. Egorov M. Stableswap—efficient mechanism for stablecoin liquidity; 2019.
36. Wan X, Adams A. Just-in-time liquidity on the uniswap protocol. *SSRN Electronic J*. 2022. https://doi.org/10.2139/ssrn.438230
37. Xiong X, Wang Z, Knottenbelt W, Huth M. Demystifying just-in-time (JIT) liquidity attacks on uniswap v3. Cryptology ePrint Archive; 2023.

How to cite this article: Singh SF, Michalopoulos P, Veneris A. DEEPER: A shared liquidity decentralized exchange design for low trading volume tokens to enhance average liquidity. *Int J Network Mgmt*. 2024;e2261. doi:10.1002/nem.2261.