.

by sm-stack & chris

.

Many thanks to Sang Kim & WonJ & Brynn & Sangwon Moon for discussions and reviews!

.

This research was inspired by the research “LVR-minimization in Uniswap V4” by@The-CTra1n, and the opinions expressed in this article are our own and not affiliated with the original author.

We conducted various simulations based on the research ‘LVR-minimization in Uniswap V4”

and the paper which utilizes Diamond Protocol to address LVR, and found that implementing LVR minimization through “Per-block conversion vs. Futures contract” rather than “Low Impact Re-add”, the solution presented in the paper, is more profitable in terms of profitability and slippage. In this research, we will describe the “Per-block conversion vs. Futures contracts” solution, share and analyze the simulation results, discuss what needs to be considered further, and finally describe the pseudo code for this solution.

High Level Overview: Diamond protocol

The Diamond Protocol refers to a liquidity pool that requires arbitrageur to pledge collateral to arbitrage against that liquidity pool, and reduces LVR, the liquidity provider’s loss, by sharing a portion of the arbitrage profits with the pool. This type of liquidity pool can be implemented using the Uniswap V4 Hook. Uniswap V4 Hooks run inside the initialize

, swap

, modifyPosition

donate

functions inside the PoolManager contract and allow you to execute specific logic before and after each action. Since LVR-minimization occurs when an arbitrageur makes a trade to gain arbitrage profits on the AMM, the main logic is executed in beforeSwap

and afterSwap

.

Before explaining the detailed logic, let’s see how it is implemented at a high level. The first swap opportunity in a given block is given to the arbitrageur who pledged collateral. Arbitrage beyond the value of the collateral pledged to the liquidity pool is not allowed, in order to maintain the price of the tokens in the liquidity pool, which is set by the first swap, even after subsequent swaps.

The first swap in a block gives arbitrageur an exclusive arbitrage opportunity, and the liquidity pool’s token price is arbitraged to equal the market price. (The pool’s token price remains at the market price until the end of the block.) At that point, arbitrageur will distribute a portion of the arbitrage profits to the pool, and the free tokens generated to bring the pool’s token price to the market price will be placed in the Vault. The second swap in a given block will be executed by the general public, and any change in the pool’s token price due to the swap will be matched using collateral held by arbitrageur.

[

Diamond-protocol-logic

1718×908 124 KB

](https://ethresear.ch/uploads/default/original/2X/3/310ea760c8325539bd4d7ee3f6e4c8ce7c18dd7c.png)

According to the above logic, as the block progresses, the vault will continue to accumulate tokens and the liquidity pool itself will become increasingly scarce. Therefore, unless you change the price of the liquidity pool’s tokens, you need to redistribute the vault’s tokens to the liquidity pool. Depending on the logic of this process, the return and slippage impact of the liquidity pool will vary. “LVR-mimization in Uniswap V4”

simulated and proposed pseudo code using the Low Impact Re-adding method. In simple terms, this method moves a certain percentage(1 - 5%) of the tokens in the vault to the liquidity pool for the first swap of every block. We use a different method, Per-block Conversion vs. Futures Contracts, presented in the paper “An Automated Market Maker Minimizing Loss-Versus-Rebalancing”

. The idea is to borrow tokens from arbitrageur using futures contracts, and then move as much of the tokens in the vault as possible into the liquidity pool.

Existing Solutions: Periodic Conversion, Low Impact Re-adding

Before we talk about the method we used, let's briefly explain the Periodic Conversion and Low Impact Re-adding method.

The first is the Periodic Conversion method. The Periodic Conversion method was presented in the paper "An Automated Market Maker Minimizing Loss-Versus-Rebalancing"

along with the Per-block Conversion vs. Futures Contracts method, where every {\tau}

block, {\eta}

worth of tokens, equal to half of the tokens in the vault, are auctioned to all users in the system, and the winner of the auction swaps the tokens in the vault into a liquidity pool.

The second is the Low Impact Re-adding method presented in the "LVR Minimization in Uniswap V4"

research, as described above, involves putting a certain percentage of the tokens in the vault directly into the pool without any conversion process. The percentage of tokens that go back into the pool in this case is called the Vault Re-deposit Ratio, and prior research("An Automated Market Maker Minimizing Loss-Versus-Rebalancing"

) has shown that a ratio of 1-5% is optimal.

We'll get a little more specific about the approach we used below.

The Solution: Per-block conversion vs. Futures contracts methodolgy of retained LVR into the liquidity pool

As mentioned earlier, there will be free tokens in the vault after arbitrageur executes an arbitrage trade. If we immediately put all the free tokens back into the liquidity pool, the price of the tokens in the liquidity pool will not match the market price. Therefore, at the market price {p_c}

, arbitrageur will swap half of the tokens in the vault and put all the tokens into the liquidity pool to match the market price to token ratio. Since this is a risky situation for arbitrageur to simply swap tokens at the market price {p_c}

, arbitrageur creates a futures contract to sell half of the swapped tokens at the same price. This has the same effect as if arbitrageur were lending tokens to the pool.

These futures contracts are executed every {\tau}

blocks, at which point the pool either loses or gains. If the pool price after {\tau}

blocks is {p_T}

and arbitrageur sends ({s_x, s_y}

) to the liquidity pool by executing a futures contract, the pool's {PnL}

will be:

{PnL = s_x + s_y*p_T}

Let's look at an example to illustrate.

If the vault currently contains 10X tokens and the token price is X = 10Y, arbitrageur will exchange 5X tokens for 50Y tokens and create a futures contract to sell the resulting 5X tokens at 50Y. It then re-adds all the tokens in the vault directly to the liquidity pool.

And let's say the token price at the time of the auction for the futures contract, which is "every {\tau}

block," is X = 20Y. People will flock to the futures contract because it's an opportunity to buy token X at a lower price, and the futures contract will be filled. The arbitrageur that created the original futures contract will then give 5X to the winning bidder and receive 50Y, and the liquidity pool will receive the difference, 50Y.

Conversely, suppose the price of the token drops to X = 5Y. Then the liquidity pool has to step in because no one will participate in the auction to buy the futures contract, or rather, the liquidity pool will have to sell the futures contract to the auction participants with additional tokens on top of it. In this case, we will receive a negative bid like the one below.

Person 1

Person 2

Person 3

Bid

-26Y

-27Y

-25Y

If Person 1 wins the auction, Person 1 will arbitrage a total gain of 1Y, since 5X = 25Y at the current external market price. However, in the example in the table above, Person 3 wins because he has the lowest absolute value bid.

The detailed algorithm can be found in the pseudo code provided at the bottom of this post.

Simulation Result (HODL / Periodic Conversion / Low Impact Re-adding / Per-block Conversion vs. Futures Contracts)

All simulation code can be found here.

[

sim-result-1

3200×2400 229 KB

](https://ethresear.ch/uploads/default/original/2X/b/b3e53ba021640b8d65d9acd5979df18f53b1e8aa.png)

In the graph above, the y-axis represents the return and the x-axis is the final price, with greater price volatility at the two extremes, and higher returns for Per-block Conversion vs. Futures Contracts than Periodic Conversion. This is because Per-block Conversion vs. Futures Contracts empties the vault every block, whereas Periodic Conversion empties the vault every {\tau}

blocks, so it is less exposed to price changes, resulting in a larger LVR.

Next, let's compare the simulation results of the Low Impact Re-adding and Per-block Conversion vs. Futures Contracts approaches.

[

sim-result-2

823×608 42.7 KB

](https://ethresear.ch/uploads/default/original/2X/4/424265505a80509f0e2e6263e6629903820dc727.png)

You can see that Low Impact Re-adding is more profitable when the price continues to rise or fall, and Per-block Conversions vs. Futures Contracts is more profitable when the price is moving sideways based on the market price. In the event of successive price increases or decreases, the Low Impact Re-adding method will only move a fraction of the tokens into the liquidity pool, resulting in a continuous accumulation of one type of token in the vault. If you HODL your tokens without placing them in a liquidity pool, the higher the final price, the higher the return. We can speculate that the tokens in the vault are more profitable because they are not used for swaps and have a similar effect to HODLs that are just held.

Finally, let's compare the slippage of Low Impact Re-adding and Per-block Conversion vs. Futures Contracts, where the graph is a token1 → token0

swap(price is token0/token1).

[

sim-result-3

776×590 32.8 KB

](https://ethresear.ch/uploads/default/original/2X/1/18999e822366c3a55b7e969dd89ab7ee2824451b.png)

You can see that slippage is significantly better with Per-block Conversion vs. Futures Contracts in situations with large price changes. Slippage is the difference between the theoretical number of tokens a user wants to swap and the actual number of tokens swapped; the more liquid the liquidity pool, the less slippage. Since Per-block Conversion vs. Futures Contracts puts all the tokens in the vault into the liquidity pool every block, it produces better results in terms of slippage than Low Impact Re-add, which only puts a certain percentage of tokens back into the liquidity pool every block.

Trade-off between "Conversion Frequency {\tau}

" and "Cost and Quality of auction"

The simulation results based on the Conversion Freqeuncy {\tau}

, a constant that can be adjusted in Per-block Conversion vs. Futures Contracts, are as follows.

[

sim-result-4

754×560 54.2 KB

](https://ethresear.ch/uploads/default/original/2X/e/e0a116394e525c27fc43bbdab3083f29ed465bbd.png)

[

sim-result-5

1750×1313 221 KB

](https://ethresear.ch/uploads/default/original/2X/d/dc69c492c41349c901cce601726f492103586bd1.png)

(From left to right, Conversion Frequency = 5, 10, 20)

In conclusion, we can see that the more frequently you hold auctions, the more profitable they are. We can see that this result is achieved because the shorter the auction cycle, the less exposed the token is to price volatility. One question might arise here. "Shouldn’t I just open an auction every block?”

However, opening an auction every block may reduce participation in the auction and result in contracts being settled at a price that is somewhat off the market price. The liquidity pool loses money and LPs lose revenue. In other words, there is a tradeoff between auction frequency and minimizing the impact of price volatility. This tradeoff is characterized by the importance of setting the appropriate duration of the auction by referring to several existing auctions.

For CoWSwap auctions, they use an auction matching system called Autopilot to manage the auction. This system does not specify the frequency of auctions, but rather determines the frequency of auctions based on the participation of bidders in the auction. This ensures the profitability of the protocol and the user experience. As you can see, auction matching systems are a very important area and there is a lot of prior research in this area.

We are currently implementing a Uniswap V4 Diamond Hook that implements Per-block Conversion vs. Futures Contracts based on the pseudo code below. We always welcome comments related to our work.

Pseudo code

[

pseudo-1

500×712 32.5 KB

](https://ethresear.ch/uploads/default/original/2X/e/ef146e39c2eca138c763a39e171b904aeb17d2ad.png)

[

pseudo-3

1038×1382 84.2 KB

](https://ethresear.ch/uploads/default/original/2X/d/d0a0a292b5d456779deeb449c70b5927aac5e435.png)