Comparing Uniswap, a new liquidity algorithm for decentralized exchanges

BY he.d.d.shan@hotmail.com

Notice:

1. Some of the conclusions in this article are tested and some are derived from theory.
2. Readers of this article would be better off understanding Uniswap' s constant product theory.

Uniswap' s algorithm

Here is a brief review of Uniswap' s liquidity identity algorithm. For simplicity, UniswapV1V2 will be used for illustration.

For the trading pair TokenA=>TokenB, the corresponding liquidity is X and Y, then X*Y=K. When trading, X or Y will change, but the K value remains unchanged.

Assuming that the original liquidity is X0 and Y0, and the user pays dY amount of TokenB to purchase x amount of TokenA, the liquidity formula needs to be satisfied: X0 * Y0 = (X0 - x) * (Y0 + dY), which can be derived

x = X0 -(~0 * Y0)/ (Y0 + dY)- X0=(X0* dY)/ (Y0 + dY)

That is (*Formula 1*) $x = (X0 * dY) / (Y0 + dY)$

He's new algorithm (continuous price theory)

The core logic is: do not use x*y=k to deduce the transaction quantity, but use the price to deduce it.

After the purchase, the price has two forms of expression. The purchase price is: bPrice = x / dY; and the price of the remaining liquidity is: lPrice = (X0-x) / (Y0+dy). Of course, the two prices are the same (this is the core point), then x / dY = (X0-x) / (Y0+dy), which can be derived

x * (Y0+dy) = (X0-x) * dY

x * (Y0+2 * dy) = X0 * dY

That is (*Formula 2*) $x = (X0 * dY) / (Y0 + 2 * dY)$

Has anyone considered this algorithm? I haven't seen anyone propose this algorithm in the Chinese community. Other famous DEXs do not use this algorithm, such as PancakeSwap, dXdY, etc.

Expand He's algorithm (continuous price theory)

If there is a trading pool containing TokenA, TokenB, and TokenC, is it possible to trade any trading pair among these three Tokens? I am very sure of that.

There is a problem in this situation. When trading TokenA and TokenB,

the other two trading pairs TokenA=>TokenC and TokenB=>TokenC do not trade at all, but the prices change (one goes up and the other goes down). Because TokenA and TokenB in the corresponding pool have changed. So is this situation reasonable or unreasonable?

Whether it is reasonable or not depends on Uniswap:

In Uniswap, there are trading pairs TokenA=>TokenB, TokenB=>TokenC, TokenC=>TokenA. If a transaction occurs on the trading pair TokenC=>TokenA, the closed loop of the trading pair (TokenB=>TokenC, TokenC=>TokenA, TokenA =>TokenB) There are arbitrage opportunities. I won't go into details here. Friends who have done arbitrage will understand it at a glance. Friends who don't understand this can check the relevant information. The result of arbitrage is that the price comparison between TokenA and TokenC obtained through the path TokenA=>TokenB=>TokenC (three trading pairs) is the same as the price comparison of the trading pair TokenA=>TokenC (this one trading pair), which ultimately leads to The price of TokenB has changed relative to both TokenA and TokenB (one has gone up, the other has gone down). The handling fee is ignored here, but it does not affect the logic.

Therefore, it is reasonable to use the trading pool of the new algorithm to trade TokenA and TokenC, resulting in a change in the price of TokenB relative to TokenA and TokenB.

So, can a trading pool composed of four Tokens work? The answer is yes, five or six are fine.

In addition, compared to Uniswap, the definition of liquidity value has also changed. In this algorithm, it is more similar to the definition of UniswapV1V2, which is a weight.

Some meanings of He's algorithm (continuous price theory)

1. Can avoid internal arbitrage

The new algorithm circumvents internal arbitrage. This has been said above.

Uniswap's internal arbitrage is very serious, resulting in a large number of arbitrage transactions and increased gas fees. I personally feel that it has a certain negative impact on the Ethereum ecosystem.

Taken to the extreme, if there is only one decentralized exchange in Ethereum that uses this algorithm, and the trading pool of this exchange includes all Tokens, for example 200 Tokens, then there will be no arbitrage opportunities on Ethereum (of course cross-chain arbitrage opportunities are another matter).

2. Improve the utilization rate of liquidity funds

In a multi-Token trading pool, one Token corresponds to multiple

Token trading pairs, so the capital utilization rate of this Token is theoretically improved compared to Uniswap.

3. Reduce the impermanent losses of liquidity providers

Compared with Uniswap's Formula 1, Formula 2 of this algorithm accelerates the price decline, which is equivalent to reducing impermanent losses and increasing the profits of liquidity providers.
Of course, this is a factor that theoretically increases profits. The actual profit is related to many factors, which will not be discussed here.

my plan

1. Find a time to submit this algorithm to Uniswap and see what they say. It would be best if it could be adopted by Uniswap.
2. If necessary, I will write a set of code, including contracts and clients, to test and demonstrate this idea. My contract code may not be as perfect as Uniswap's, but it can ensure security and correct calculation, and can achieve a commercial level.

Possible improvements to the algorithm

1. Separate Token adds liquidity. If there are 30 types of Tokens in the pool, do users need to add 30 types of Tokens at once? This certainly doesn't make sense. It should be possible to add any kind of Token, including adding only one Token.
2. The calculation of liquidity, mainly the calculation of price, requires the number of Tokens corresponding to liquidity to be "blurred", "sampled" or "sharded", similar to UniswapV3's approach, which emphasizes liquidity in a certain range. Distribution, so that the corresponding relationship between the actual quantity and price of each Token can be separated, making it possible to add liquidity to a single Token.
3. It is easiest to unify the handling fees for each trading pair. But what about disunity? You can set a fee level, first class, second class. The transaction fee for first-class and first-class tokens is 3; the fee for first-class and second-class tokens, and second-class and second-class tokens is 5. There are many methods, I will just list this method here.