Internship report : Analysis of opportunities in the field of decentralized finance (BlockChain Ethereum)

Jean GRIMAL

October 2021 - March 2022

Contents

1 Introduction

The objective of this paper is to present the work I did during my internship at GebelGroup (a market making company for digital assets based in Paris). This internship was focused on DeFi issues, and especially on the Uniswap liqudity pools.

I will summarize each step: from my first steps in Defi to the implementation of an Princing algorithm for Uniswap liquidity pools, through the analysis of a large number of swaps from the biggest pools.

For confidentiality reasons, I will not give any results or code elements.

2 First steps

The first month of my internship was dedicated to the discovery of decentralized finance, and more specifically to the understanding of the functioning of Uniswap, the main decentralized exchange protocol (DEX), and then to get to know in detail the smart contracts of the V2 and V3 verions of Uniswap.

3 Swap analysis

The second part of the internship was to analyze the swaps linked to the main Uniswap pools (those with the most transactions, the largest liquidity reserves), in general these are the pools that associate WETH with other widely used tokens: USDC, USDT and DAI. We have therefore decided to focus on the WETH/USDC, WETH/USDT and WETH/DAI pools in V2, V3 at 0.3% fees and V3 at 0.05% fees: that makes 9 pools in total.

3.1 Data extraction

I obtained the swap data via Etherscan: I used python and the Etherscan API, and sent requests to extract the last 100,000 transactions related to each liquidity pool considered, which I exported as .csv files.

3.2 Swap filtration

The final objective being the implementation of an arbitrage strategy, we were only interested in "spot" operations (i.e. simple swaps), so we excluded all operations including additions and withdrawals of liquidity (mints and burns), or successions of swaps concerning the same Pool in the same block.

To do this, I used the Python Pandas library to process these large data tables and filter them in the desired way.

3.3 Analysys

Still on Pandas, now that I had the data I want, I first computed the net price associated to each swap thanks to the entry and exit amounts, remembering to subtract the amount of fees (for Liquidity Providers) and the gas fees (blockchain fees for each mined transaction, they can be computed for each swap thanks to the data extracted on Etherscan).

I was then able to calculate several **statistics** for each pool: the daily frequency of swaps, the amount of the biggest swap, the average amount, the quartiles and percentile 99, 95, 90 of the amounts, as well as the 5 addresses that trade the most on each pool, and the percentage of the volume handled by these addresses compared to the total volume.

This allowed us to compare the behaviors associated with V2 and V3 versions of Uniswap, to note the differences between the 0.3% and 0.05% pools, get an idea of the swap amounts, and identify the behavior of the addresses that trade the most.

I also **compared these price data to centralized market prices** (Coinbase and Binance) that I calculated with impact methods on centralized market orderbooks (archived by the company's data scientist) **with matching timestamps.**

4 Pricing

Now that we had an idea of the orders of magnitude of the trades that take place on the pools (in terms of frequency and size) as well as the opportunities present on Uniswap, the next step was to elaborate a python pricing script for these liquidity pools.

4.1 instantaneous data source

The first step was to get the data from the pools in real time: liquidity values (and tick values for V3), last swaps, mints and burns. I first used *theGraph* API. But because of latency issues, I finally decided to connect directly to the smart contracts of the pools via the web3.py library, and use the view functions and listen to the events emitted by the pools.

4.2 What does the function do?

This function implements the calculations made by the smart contracts of the Uniswap pools. The input arguments of my pricing function are the updated pool variables and a list of amounts (for one asset). The function gives as output the instantaneous buy and sell price of these amounts on this pool

It is possible to call this function simultaneously for several pools to identify arbitrage opportunities between these pools.

4.3 Validation

To validate the accuracy of my pricing function, I tested it on several swaps (I fetched the swaps data by listening to the swap events emitted by a pool): I applied the function to the input amount of the swap and to the variables of the pool upstream of the swap, and I compared the output amount given by my function to the actual output amount of the swap.

The observed differences are of the order of 10^9 **base points.**

This function allows us to identify inter-pool arbitrage opportunities or opportunities when the price of the pools is sufficiently out of line with that of the CEX.

5 Swap execution

Once the Pricing algorithm was validated, the new goal was to be able to perform a swap from our python script (when an arbitrage opportunity arises).

To do this, I went deeper into the web3.py library and the Uniswap smart contracts (especially the Router contract) but also into the ERC-20 contracts. I then learned (after multiple attempts on the testnets) how to handle the (simple and multiple) swap functions, but also the ERC-20 transfer function, and finally succeeded to execute swaps from my python pricing code.

This stage also allowed me to learn how to handle the **Gas fees**, to make sure that our swap is included within the next block.

6 Compound

In the same way as for the swap functions of the Uniswap smart contracts, I learned to handle the collateral deposit and borrowing functions of the Compound smart contracts (in case we are missing an asset to seize an opportunity).

7 Conclusion

To summarize, during this internship :

- I learned a lot about the decentralized finance ecosystem and about the functioning (in details) of Uniswap (V2 and V3).
- I was brought to do data manipulation and analysis (swaps), and to compare Uniswap prices with CEX prices.
- I implemented a very accurate Uniswap Pricing algorithm of Uniswap Pools.
- I mastered the communication with Smart contracts deployed on the Ethereum Blockchain (Uniswap and Compound in particular) via the Web3.py library.