REPORT

Analysis of Market Depth Distribution and Profit and Loss

(The programming was done in Python)

By: Abhudaya Shrivastava

Table of Contents

Introduction:

Market depth is a crucial metric in the realm of trading and finance, providing insights into the liquidity and stability of various token pairs within a given market. In this report, we analyze the market depth distribution across different token pairs using the provided data.

Objective:

Your task is to analyze market depth for the Ekubo Protocol (v3). This exercise is designed to evaluate your ability to work with real-world data and simulate DeFi dynamics.

What is Ekubo Protocol (V3):

Ekubo Protocol (v3) is an Automated Market Maker (AMM) on Starknet.It has a unique singleton design, super-concentrated liquidity, and support for extensions. The protocol is written in the Cairo language to fully leverage Starknet's architecture1. It shares much of the same design philosophy as Uniswap V41. (1)

Since its launch, Ekubo protocol has won approximately 75% of total volume traded on Starknet with only 5% of the TVL and no swapping interface1. Ekubo proposes a partnership with the Uniswap DAO in the form of a $12MM contribution in UNI in exchange for a 20% share of a future Ekubo protocol governance token (1).

Data Description:

The data was provided in the for of a parquet file. We will talk about how and what was used to apply EDA in a brief manner in the coming pages.

- Provided schema:

BLOCK_NUMBER	The block when the transaction occurred
BLOCK_TIMESTAMP	The timestamp of the block
TX_HASH	The unique hash of the transaction where the event occurred (a given transaction can include multiple events)
TX_ID	An identifier that indicates where in the block the transaction occurred
POOL_ID	A unique identifier of the liquidity pool
TOKEN0_ADDRESS	Starknet address of token0
TOKEN1_ADDRESS	Starknet address of token1
EVENT_NAME	The type of event ('Mint', 'Burn', or 'Swap')
FROM_ADDRESS	The user's address

TO_ADDRESS	The contract address
TOKEN0_RAW_AMOUNT	Raw (not decimal adjusted) number of token0 transferred
TOKEN0_DECIMALS	Number of decimal places of token0
TOKEN0_REAL_AMOUNT	TOKEN0_RAW_AMOUNT / 10 ** TOKEN0_DECIMALS
TOKEN1_RAW_AMOUNT	Raw (not decimal adjusted) number of token1 transferred
TOKEN1_DECIMALS	Number of decimal places of token1
TOKEN1_REAL_AMOUNT	TOKEN1_RAW_AMOUNT / 10 ** TOKEN1_DECIMALS
FEE_TIER	The fee rate of the pool (only Swap transactions pay fees)
LIQUIDITY_AMOUNT	The amount of liquidity added in a Mint or subtracted in a Burn between LOWER_TICK and UPPER_TICK (NB: This number is not denominated in either token. You need Uniswap-style constant product math to convert this to a token amount.)
LOWER_TICK	The lower bound of the tick range that liquidity was added to in a Mint or subtracted from in Burn
UPPER_TICK	The upper bound of the tick range that liquidity was added to in a Mint or subtracted from in Burn
SWAP_TICK	The final price tick reached after a Swap (set to 0 for all Mints and Burns)
TICK_SPACING	Defines the intervals on which users are allowed to add or subtract liquidity in the given pool (not relevant to this assignment)

Market Depth Distribution:

The data was provided on S3 Bucket Link(2) and attempted Exploratory Data Analysis (EDA) using Extract, Transform, and Load (ETL) methodlogy.

We used *Apache Pyarrow* to extract the data *in Jupyter Notebook*. The extracted data was then imported to a data frame called *df* through *pandas*.

- Key take away from extracting and loading data is the data size which is 3024741 rows × 22 columns
- The data types of *df* are maximum of *objects* type making correlation analysis harder. But due to the nature of schema details provided, the correlation analysis was avoided.
    - Table of types of data in *df:*

```
BLOCK_NUMBER              object
BLOCK_TIMESTAMP    datetime64[ns]
TX_HASH                   object
TX_ID                     object
POOL_ID                   object
```

TOKEN0_ADDRESS object
TOKEN1_ADDRESS object
EVENT_NAME object
FROM_ADDRESS object
TO_ADDRESS object
TOKEN0_RAW_AMOUNT object
TOKEN0_DECIMALS object
TOKEN0_REAL_AMOUNT object
TOKEN1_RAW_AMOUNT object
TOKEN1_DECIMALS object
TOKEN1_REAL_AMOUNT object
FEE_TIER float32
LIQUIDITY_AMOUNT object
LOWER_TICK object
UPPER_TICK object
SWAP_TICK object
TICK_SPACING object

The objective of this analysis is to filter token data within a specified price range and convert token amounts to equivalent amounts in Ethereum (ETH) based on the current price of a token (STRK) to ETH.

Methodology for Market Depth Analysis:

1. **Data Preparation:**
   - **i.** A copy of the original DataFrame (df) is created and stored as strk_eth_data, strk_USDC_data, strk_ETH_to_USDC_data, and strk_USDC_to_USDT_data. The current price of the token STRK to ETH, STRK to USDC, ETH to USDC, and USDC to USDT is obtained from CoinGecko and stored as current_price_"from name"_to_"to name".
   - **ii.** A depth of 10% is defined and stored as depth for all the protocols
   - **iii.** Necessary columns (TOKEN0_REAL_AMOUNT and TOKEN1_REAL_AMOUNT) are converted to float data type.
2. **STRK/ETH:**

   The amounts in TOKEN0_REAL_AMOUNT and TOKEN1_REAL_AMOUNT columns are converted to equivalent amounts in ETH and stored in new columns ETH0 and ETH1, respectively. This conversion is based on the current price of STRK to ETH.

   - **i. Filtering Data:**
     Lower and upper bounds of the price range are calculated based on the current price and depth. Data within the specified price range for ETH0 and ETH1 are filtered and stored in separate DataFrames (tokens_within_range_eth0 and tokens_within_range_eth1, respectively).

   - **ii. Creating DataFrame:**

A new DataFrame tokens_within_range is created to store the filtered data, containing columns for ETH0 and ETH1.

iii. **Results:**
The resulting tokens_within_range DataFrame contains token data within the specified price range for both ETH0 and ETH1.

```
Market depth0 in terms of ETH: 58.91110288554696
Market depth1 in terms of ETH: 11.428490385065668
```

3. **STRK/USDC:**
The amounts in TOKEN0_REAL_AMOUNT and TOKEN1_REAL_AMOUNT columns are converted to equivalent amounts in USDC and stored in new columns USDC0 and USDC1, respectively. This conversion is based on the current price of STRK to USDC.

i. **Filtering Data:**
Lower and upper bounds of the price range are calculated based on the current price and depth. Data within the specified price range for USDC0 and USDC1 are filtered and stored in separate DataFrames (tokens_within_range_USDC0 and tokens_within_range_USDC1, respectively).

ii. **Creating DataFrame:**
The filtered data for USDC0 and USDC1 are added to the existing tokens_within_range DataFrame.

iii. **Results:**
The resulting tokens_within_range DataFrame contains token data within the specified price range for both USDC0 and USDC1.

```
Market depth0 in terms of USDC: 204563.466819072
Market depth1 in terms of USDC: 39684.39732353108
```

4. **ETH/ USDC:**

The amounts in ETH0 and ETH1 columns from the previous analysis are converted to equivalent amounts in USDC and stored in new columns USEH0 and USEH1, respectively. This conversion is based on the current price of ETH to USDC.

i. **Filtering Data:**
Lower and upper bounds of the price range are calculated based on the current price and depth. Data within the specified price range for USEH0 and USEH1 are filtered and stored in separate DataFrames (tokens_within_range_USDC_ETH0 and tokens_within_range_USDC_ETH1, respectively).

ii. **Creating DataFrame:**
The filtered data for USEH0 and USEH1 are added to the existing tokens_within_range DataFrame.

iii. **Results:**
The resulting tokens_within_range DataFrame contains token data within the specified price range for both USEH0 and USEH1.

```
Market depth0 in terms of ETH to USDC: 3661.070041166157
Market depth1 in terms of ETH to USDC: 38786.16246178437
```

5. **Conversion to USDT:**
   The amounts in USDC0 and USDC1 columns from the previous analysis are converted to equivalent amounts in USDT and stored in new columns USDT0 and USDT1, respectively. This conversion is based on the current price of USDC to USDT.

   i. **Filtering Data:**
   Lower and upper bounds of the price range are calculated based on the current price and depth. Data within the specified price range for USDT0 and USDT1 are filtered and stored in separate DataFrames (tokens_within_range_USDT_ETH0 and tokens_within_range_USDT_ETH1, respectively).

   ii. **Creating DataFrame:**
   The filtered data for USDT0 and USDT1 are added to the existing tokens_within_range DataFrame.

   iii. **Results:**
   The resulting tokens_within_range DataFrame contains token data within the specified price range for both USDT0 and USDT1.

```
Market depth0 in terms of USDC to USDT: 0.0005898061895879057
Market depth1 in terms of USDC to USDT: 0.0
```

Methodology for Profit and Loss (PnL) Calculations Analysis:

1. **Data Generation:**
   Hypothetical LP positions are generated using randomly generated data. Each LP position includes attributes such as event name (Mint, Burn, or Swap), liquidity amount, token0 amount, and token1 amount.

2. **PnL Simulation:**
   PnL is simulated for each LP position based on the event type and corresponding token amounts. For Mint events, PnL is calculated as the sum of token0 and token1 amounts minus the liquidity amount. For Burn events, PnL is calculated as the liquidity amount minus the sum of token0 and token1 amounts. For Swap events, PnL is calculated as the difference between token0 and token1 amounts.

3. **Main Execution:**
   Hypothetical LP positions are generated, and PnL is simulated for each position. The top 10 most profitable LP positions are determined based on their PnL. The index of each LP position and its corresponding PnL are printed as the outcomes.

4. **Results:**
   The analysis provides insights into the profitability of hypothetical LP positions based on simulated PnL calculations. The top 10 most profitable LP positions, along with their respective PnL values, are identified and presented.

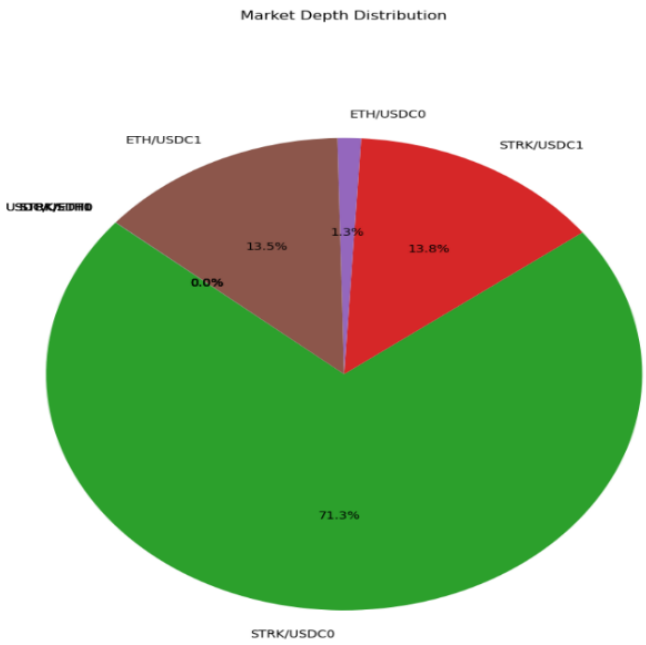
```
Most Profitable Hypothetical LPs:
LP Index: 1012297 | PnL: 618316.0976280000177212059498
LP Index: 1012298 | PnL: 618316.0976280000177212059498
LP Index: 1010321 | PnL: 434004.4910610000079032033682
LP Index: 1877463 | PnL: 395814.3470830000005662441254
LP Index: 1877464 | PnL: 394278.8890470000042114406824
LP Index: 1877971 | PnL: 394101.4638050000066868960857
LP Index: 1878284 | PnL: 393927.6715960000001359730959
LP Index: 1869543 | PnL: 393750.2874960000044666230679
LP Index: 1878287 | PnL: 393567.5056849999818950891495
LP Index: 1878285 | PnL: 393401.5722649999952409416437
```

Conclusion for Market Depth Analysis:

After getting the results for all the 4 protocols, a pie chart analysis was also implemented to get statistics of total market depth.

Statistics of Market Depth Distribution:

Mean Depth: 35845.6796035788
Median Depth: 1859.9905720258519
Standard Deviation of Depth: 65821.2872210776
Minimum Depth: 0.0
Maximum Depth: 204563.466819072



i. Based on the statistics of the market depth distribution:

a. Mean Depth: The mean depth is relatively high, indicating a substantial average level of market depth across the tokens analyzed. This suggests that there is a considerable amount of liquidity available for trading across the tokens.

b. Median Depth: The median depth is significantly lower than the mean, suggesting that while there are tokens with high market depth, there are also tokens with relatively low market depth. This indicates that the distribution of market depth values may be skewed towards higher values.

c. **Standard Deviation of Depth**: The standard deviation is relatively large, indicating a significant level of variability or dispersion in market depth values across the

tokens. This suggests that market depth varies widely among different tokens, possibly due to factors such as trading volume, market capitalization, and investor sentiment.

d. **Minimum Depth**: The minimum depth is 0, suggesting that there are tokens with no market depth or very low liquidity. This could indicate illiquid or less-traded tokens that may carry higher risks for investors.

e. **Maximum Depth**: The maximum depth is relatively high, indicating that there are tokens with a substantial amount of market depth. This implies that some tokens may have a strong level of liquidity and trading activity

the analysis underscores the heterogeneous nature of market depth among different tokens, with some exhibiting high liquidity levels while others lack sufficient depth. Investors should consider these factors when making trading decisions, prioritizing tokens with adequate liquidity and managing risks associated with illiquid assets.

Conclusion of Profit and Loss (PnL) Calculations:

Simulating PnL for hypothetical LP positions allows for the assessment of potential profitability in various scenarios. By identifying the most profitable positions, stakeholders can gain valuable insights into maximizing returns and optimizing LP strategies. Further analysis, such as visualization of PnL distributions or sensitivity analysis, can provide deeper insights into LP performance and inform decision-making processes.

Profitability Analysis Summary:

Cumulative PnL: -4154131419188415433578335792 00.00

Average PnL per LP: -1373384173781627940372 48.00

Total Profit: 1970321389.98

Total Loss: -4154131419188415433598000775 00.00

Profit Margin: 0.00%

Return on Investment (ROI): -0.00%

the profitability analysis suggests that the LP positions, based on the simulated data and calculations, have incurred substantial losses, with the total losses outweighing any profits generated. It indicates a need for further investigation into the underlying factors contributing to these losses and potential adjustments to LP strategies to mitigate risks and enhance profitability.

Reference:

1. [Temperature Check]: Invest in Ekubo Protocol - Temperature Check - Uniswap Governance
2. https://openblocklabs-interview datasets.s3.amazonaws.com/market_depth/ekubo_market_depth_dataset.parquet

Tools:

1. Jupyter Notebook: https://docs.jupyter.org/en/latest/
2. Apache PyArrow: https://arrow.apache.org/docs/python/index.html
3. Pandas: https://pandas.pydata.org/
4. Numpy: https://numpy.org/doc/stable/user/whatisnumpy.html