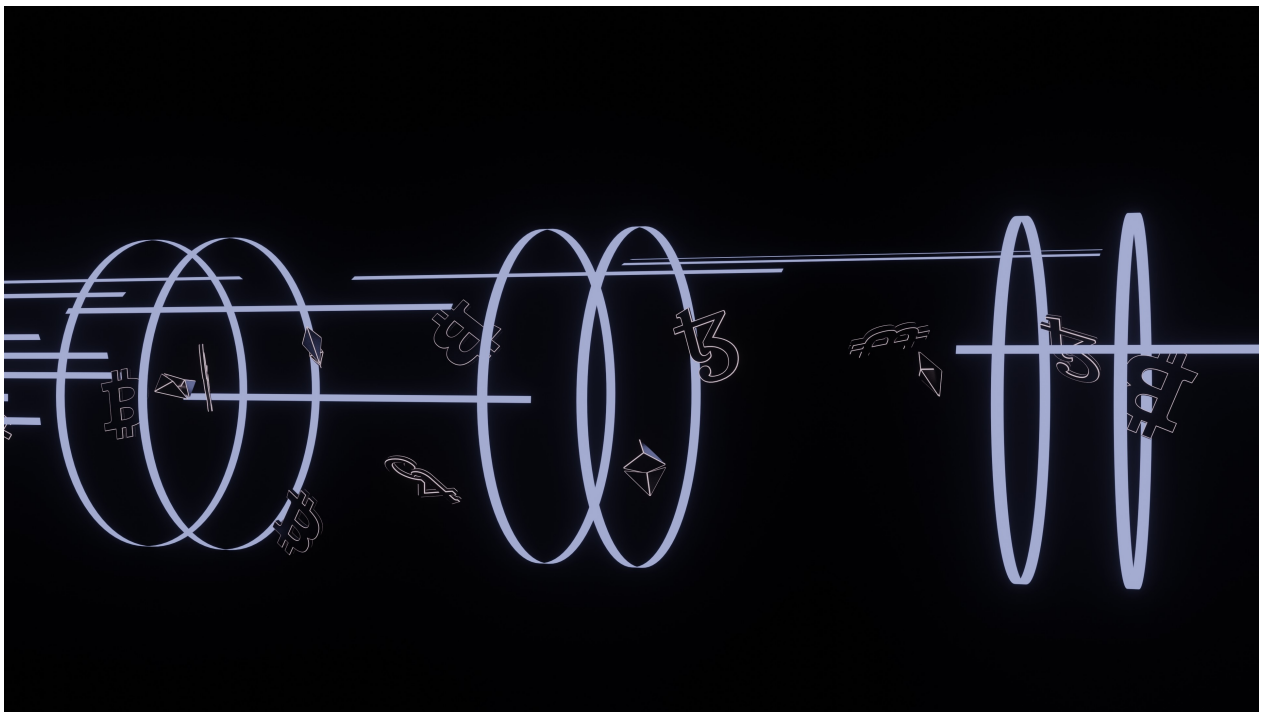
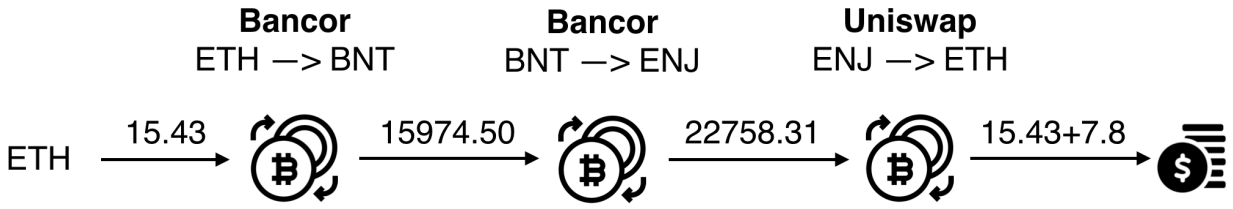
Unlocking DeFi Profits with Cyclical Arbitrage



Introduction

In the dynamic world of Decentralized Finance (DeFi), intricate and interlinked protocols have opened new possibility vectors for algorithmic trading. Cyclical arbitrage emerges as one such innovative strategy, leveraging the distinctive nature of DeFi. Rather than simply capitalizing on price discrepancies between two markets or tokens,

cyclical arbitrage employs a more complex approach. It examines and exploits potential profit opportunities found in intricate cycles of tokens and protocols.



Example of a cycle

Cyclical arbitrage shines in its ability to repeatedly utilize opportunities within uniquely characterized swap loops. This approach impressively expands the frontier of regular arbitrage strategies available in the conventional financial market.

This article shall delve into the inner workings of cyclical arbitrage - the algorithms, the data models and evaluation schemes that look for profit cycles in the DeFi protocols. We shall also examine the unique challenges, potential risks, and optimized mitigation strategies that come with this novel arbitrage strategy. Drawing upon practical examples and system designs, the aim is to provide a comprehensive walkthrough for anyone, whether you're a developer looking to design a DeFi trading bot or a DeFi enthusiast eager to understand the science behind these new strategies.

1. Theory

Cyclical arbitrage — or as it's sometimes called, network arbitrage — in DeFi extends from the principles of traditional arbitrage, taking advantage of the highly interconnected and permissionless nature of DeFi protocols. This increasingly complex land-

scape of protocols is a fertile ground for what is referred to as "Profitable Transaction Ordering Opportunities" (PTOOs) or "profit cycles".

The identification, evaluation and execution of these potential profit-yielding transactions are underpinned by the approach presented in the aforementioned study, segmented into three distinct, yet interconnected stages: Discovery, Analysis, and Execution.

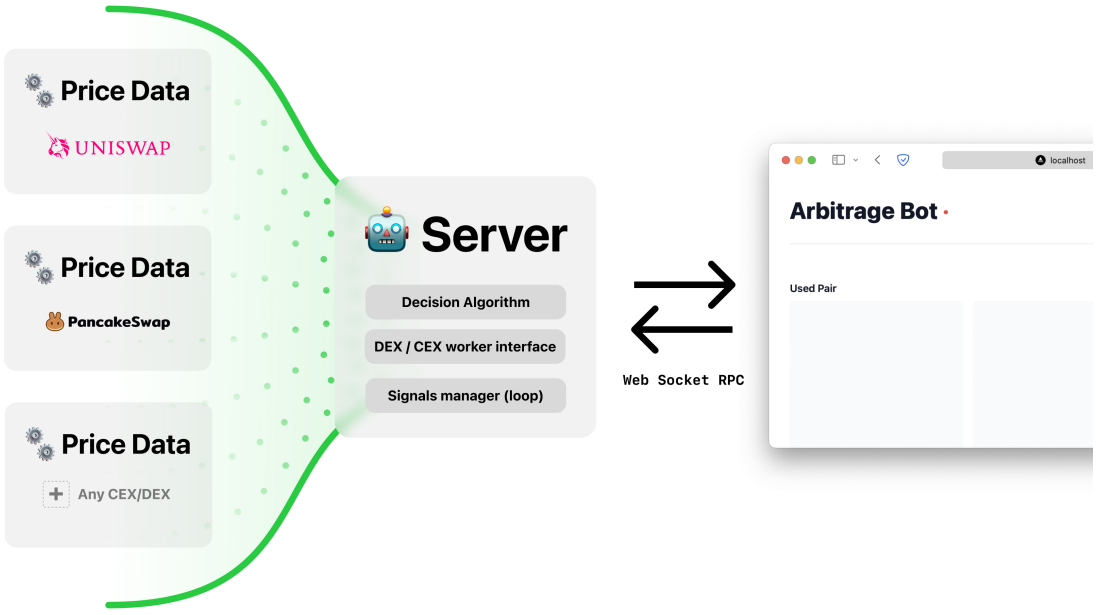
1. **Discovery**: The first stage in this cyclical arbitrage strategy is the discovery of profit opportunities within the DeFi protocols. This is achieved by representing a portion of the DeFi network as a strongly connected directed graph, where nodes represent smart contracts on the protocols and edges represent function calls between smart contracts. It is then about identifying the cycles within this graph which correspond to PTOOs, mostly through algorithmic ways such as Bellman Ford's algorithm.
2. **Analysis**: Once these cycles are identified, the next step is to evaluate them for profitability. This is a more complex task as it involves taking into account for various factors, including slippage, gas costs, and transaction fees. One major aspect of this step, is to determine the size of the transaction. As certain protocols, such as Uniswap, allow borrowing from their pool, we can use a mecanism named "flash-swap" to maximise our profit while providing no liquidity at all!
3. **Execution**: Lastly, if a cycle is deemed profitable after the analysis, it comes down to executing it. It is worth noting that the execution requires a precise sequence of function calls through smart contracts to accurately traverse the identified cycle. This sequence of execution, if done correctly,

results in profitable transactions, leading to cyclical arbitrage opportunities. Everything must be executed in a single block.

As we progress with the theory, it is important to note that the speed of discovery, analysis, and execution plays an imperative role. With advancing technology and growing competition, the quicker the process, the larger the arbitrage profit potential in the swiftly evolving DeFi space. While speed may not be as important in DeFi as it is with centralised exchanges, we can still greatly benefit from it by being above in the mempool.

This theoretical framework, drawn from established research 1, serves as the foundation of our exploration into the profitable yet intricate world of cyclical arbitrage in DeFi.

2. Data gathering



Acquiring data for the just-in-time discovery process is a multifaceted operation and forms the backbone of cyclical arbitrage strategy. Performance in data gathering is crucial, as variations in data could mean the difference between identifying a profitable cycle and overlooking an opportunity.

The crucial steps we undertook to build a robust data gathering process involve:

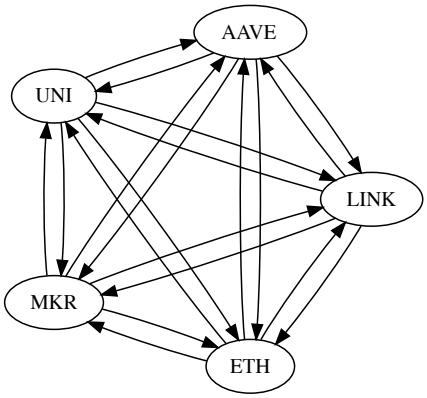
1. Liquidity Pool Identification: Our primary task was identifying liquidity pools on decentralized exchanges, with a primary focus on Uniswap V2 given its market relevance and significant trading volumes.

2. Identifying Primary Source of Truth: The essence of data gathering lies in accuracy, and ensuring the same involved identifying reliable sources. For Uniswap V2, this was primarily reserves. Basing our data on reserves allowed us to tap into accurate

and immediate pricing information, which we could then process locally for identifying arbitrage opportunities.

3. Querying Blockchain: Building an efficient data gathering process meant frequently interacting with the blockchain. We queried the blockchain at every block, subsequently updating our internal in-memory database. By doing so, we maintained an up-to-the-second influx of crucial data, helping identify arbitrage opportunities in real-time.

4. Building The Graph: The last step involved synthesizing our gathered data into a meaningful framework - a strongly connected directed graph. Nodes represented smart contracts, and edges symbolized function calls between them, effectively identifying potential cycles for PTOO.



One of the principal challenges encountered during this process was efficiency. Our primary issue was the latency present in our initial stack that leveraged TypeScript and Ethers.js. Constraints around a single connection and serial response processing, even with wiser strategies like `Promise.all` or worker threads, it still severely limited our data gathering speed and efficiency, impeding the arbitrage strategy.

We shifted our stack to Swift, and the difference was night and day. Swift, while being mostly known for mobile development, offered better throughput, was compati-

ble with lower-level languages like C/C++, and afforded a mature ecosystem, leading to improved productivity. Most notably, it natively supported concurrency, which allowed us to simultaneously process responses and perform calculations. This shift expedited our data processing capabilities considerably - facilitating us to fetch and process about a hundred pairs in nearly 50ms on the mainnet.

In retrospect, Swift was the perfect language, because of its performance, modularity, easy to read syntax and concurrency APIs. Swift aims to maximize clarity of code, and thus it fights to reduce boilerplate. The top-end goal of Swift is to optimize the time it takes to write and maintain the project, which includes debugging time and other things that go beyond just pounding out the code.

3. Finding opportunities

Once we had an efficient data gathering and storage process in place, the next hurdle was identifying opportune moments within the large and complex graph—instances that could potentially yield a profitable transaction cycle or PTOO. Our innovative strategy to spot these profit groups hinges on a classic graph theory algorithm – The Bellman-Ford algorithm.

Here's a step-by-step breakdown of the procedure:

1. Bi-directional Price Generation: Starting from our Swift data store, we generate a list of bi-directional prices. The aim was to capture the optimal price, factoring in possibilities of multiple exchanges for the same pair, where optimal rates in one direction might not always translate into the optimal rates in the reverse direction.

2. Front-End Trader Interface: On generating this list, we provide it to our framework's front-end – the part of the program directly accessible to the user, that's why

we call it front-end, don't get confused with the web – written in C, enabling better performance. This front-end processes the matrices of the fixed amount-in prices; the necessity of fixed amount-in prices arises from the fact that we cannot update the price as variable in the Bellman-Ford algorithm.

3. Negative Cycle Identification: In this step, the front-end primarily looks for negative cycles. We leverage machine-accelerated algorithms when available and opt for the default implementation as a fallback – offering potential speedups on platforms like macOS, with technologies like Accelerate & Metal.

4. Backend Evaluation: Subsequent to the cycle identification, our solution's backend steps into the fray, carries out a comprehensive review and optimization of the amount-in by way of gradient descent. It's during this phase that this system spots the best opportunities, if existing.

In the backend, one of the significant tools we utilized for mathematical assessment was Euler, a custom-built library, a subset of Euclid, a calculator app for macOS I built.

The Bellman-Ford algorithm, coupled with our robust data gathering process, accelerated front-end, and analytical backend, comes together to provide the backbone of our solution– identifying profitable transaction cycles promptly, efficiently and reliably.

4. Let's make some cash!

With the potential opportunities identified, our next aim was to facilitate on-chain trades adhering to the identified paths with the specified amounts. This sequence of

steps converting potential opportunities into tangible monetary gains is critical to the cyclical arbitrage process.

Actuated by a meticulously designed framework, we convert all pertinent information into byte data representing Solidity language structures. This data is then dispatched to a purpose-built smart contract competent in invoking the appropriate interfaces to effectuate the intermediary transactions on various exchanges.

Understanding the importance of gas optimization in DeFi, we have put in place intermediary contracts for all supported exchanges, structured to be reusable and generic. This approach not only minimizes gas usage but also enables our strategy to be resilient to potential adversarial actions like sandwich attacks. Our transactions are harder to detect and target due to this design.

Moreover, unlike fixed slippage strategies that are common in many DeFi setups, our system recomputes slippage tolerance dynamically. If a transaction does not yield profit, it simply aborts, and the only loss incurred is the gas fee, not the trading capital itself. This ensures that all transactions performed in a single block either generate profit or cost only the minimum transaction fee.

Upon a transaction's successful validation, the resulting ETH or other base tokens are sent back to the user. The program continues this process, looking towards the next potential cycle of profit.

By utilizing this system, we edge closer to making cylical arbitrage an easily accessible and potentially profitable activity in DeFi.

Demo Time

At this point, theory and process have manifested into a potent mechanism for realizing consistent returns. To truly encapsulate its capability, we need to put this system to the test. Our testnet demo provides a real-time execution of the bot, displaying how the entire strategy comes to life.

In one such testnet demo, we successfully showcased the potential the bot holds. Enacting the cyclical arbitrage strategy, the bot was capable of doubling the entered amount in a single trade! That's the dynamism of DeFi and the power of cyclical arbitrage materialized.

Although the current bot is a proof-of-concept demonstration, its success could well pave the way for a sophisticated investment strategy or product within Pyratz. The high-risk, high-yield nature of cyclical arbitrage makes it an enticing proposition for investors keen on maximizing profitability.

Beyond investment products, there are other potential applications of this bot as well. One of the significant use-cases could be to assist Automated Market Makers (AMMs) achieve more accurate pricing on non-liquid pairs. This could benefit the entire DeFi ecosystem by improving price effiency and market liquidity.

A note about me

As I sit down to write this article, I am concluding an enriching four-month journey as a blockchain engineer intern at PyratzLabs, where I had the privilege of working on intriguing projects such as the one I just shared with you.

Just before immersing myself in the world of PyratzLabs, I was studying at Florida Polytechnic University. For two and a half enriching years, I was deeply engaged in software engineering studies, a time span during which I found myself working on a myriad of intriguing projects in application development. I was focused on the conception and development of Elva, an innovative social network designed to connect groups of friends and streamline the organization of their meetings. I'm excited to resume work on Elva, armed with newfound insights and experiences.

Right before, I also worked as a freelancer, which allows me to amalgamate my expertise into real-world applications, absorb new skills, and adapt to dynamic work environments that constantly keep me on my toes. Driven by an innate curiosity and a keenness for understanding the myriad applications of technology, I found myself captivated by the transformative potential of blockchain. This fascination led me on a journey to delve into the intersection of blockchain and finance.

My internship at PyratzLabs has served as an incredible learning curve. I'd like to extend my gratitude towards Bilal El Alamy for providing me with this opportunity. I also wish to thank Mohamed Friaht, Anthony Hayot, Thomas Binetruy, Nathan Benamou, Clement Roure, David Vandenbussche and the entire PyratzLabs team for their invaluable guidance. Working with such a talented group of individuals has been a pleasure and has greatly enriched my internship experience.