Retrospectively Identifying Profit-Generating Transactions on the Algorand Blockchain

Gebele Jonas[a]

[a]*Technical University of Munich, Bavaria, Germany*

Abstract

This study systematically identifies arbitrage opportunities within Algorand's decentralized finance (DeFi) ecosystem, focusing on Automated Market Maker (AMM) platforms. Utilizing graph algorithms such as Bellman-Ford, alongside optimization techniques, we introduce a system designed for real-time and historical data arbitrage exploration. While this initial study focuses on select asset pairs and short time frames, it establishes a framework for future research. The work combines theoretical and practical insights, suggesting that further refinement and broader implementation could significantly increase arbitrage profitability.

Keywords: Decentralized Exchange, Arbitrage, Algorand, Automated Market Maker, Convex Optimization

1. Introduction

Arbitrage has always been a cornerstone of efficient markets, functioning as a self-regulating mechanism that inherently brings an asset's price to its equilibrium state across different platforms [30]. The rise of decentralized finance (DeFi), however, introduces new complexities into the arbitrage landscape, necessitating innovative analytical tools and execution protocols. While these characteristics add layers of efficiency and democratization to financial markets, they also present unique challenges. The identification and execution of arbitrage opportunities on blockchain networks involve navigating complex landscapes of liquidity pools, competition of other arbitrageurs, and network latency, among other factors. This paper aims to address these challenges by presenting a robust arbitrage system that operates in both real-time and batch modes.

We begin by providing an examination of arbitrage opportunity detection methodologies, focusing on the application of graph theory as well as optimal routing algorithms for identifying negative cycles across token networks. We then delve into optimizing transaction inputs, exploring methods ranging from binary search algorithms to more advanced techniques such as convex optimization. Last but not least we try to apply these methods to historical states to identify profit-generating transactions on Algorand.

2. Background

2.1. Algorand Blockchain

The Algorand blockchain was introduced in a 2017 white paper by Turing Award winner Silvio Micali, and its mainnet was subsequently launched in June 2019. The blockchain's native token, ALGO, serves a dual role—both as a medium for transaction fees and as an essential component in its consensus process [28].

Central to Algorand's design is its consensus algorithm, known formally as the Algorand Byzantine Fault Tolerance Protocol. Based on a Pure Proof-of-Stake (PPoS) model, the protocol is engineered to deliver immediate transaction finality and scalability. In each round, a committee of users is randomly selected to propose and validate the next block. The random selection process is securely managed through a Verifiable Random Function (VRF), offering both unpredictability and cryptographic

verifiability [20].

Algorand demonstrates capabilities that are particularly conducive to the needs of decentralized applications requiring high throughput and low latency. Whereas Ethereum manages an average throughput of approximately 30 transactions per second (tps) [21], Algorand is capable of handling up to 7,500 tps [20]. Moreover, unlike Ethereum, which primarily utilizes a fee-based transaction prioritization mechanism, Algorand typically operates on a First-Come-First-Serve (FCFS) basis [16], resorting to fee-based prioritization only under network congestion [16]. Each transaction costs a minimal fee of *0.001* ALGO, levied only upon successful execution [20]. With a block time averaging around 3-4 seconds, Algorand offers a level of speed advantageous for real-time applications[20]. These factors makes Algorand a compelling candidate for applications demanding rapid transaction processing and minimal latency but also introduces unique dynamics in the Maximal Extractable Value (MEV) extraction dynamics that influence MEV searchers' strategies [19][18][33].

Algorand supports various transaction types like payments for native token transfers (ALGO), Algorand Standard Asset (ASA) transfers for non-native tokens, and Algorand Smart Contract (ASC1) application calls. Additionally, Algorand provides atomic transaction capabilities, a feature allowing for grouped transactions. In these, a set of individual transactions are bundled into a group transaction, identified by a unique identifier, and submitted as a singular unit to the network. This guarantees that the set of transactions either all execute successfully or fail as a unit, offering a reliable mechanism for more complex transactional activities [20]. ASC1s serve as the backbone for decentralized applications on the Algorand blockchain. Written in the robust Transaction Execution Approval Language (TEAL) and executed within the Algorand Virtual Machine (AVM), each deployed ASC1 receives a unique ID upon deployment for streamlined blockchain interactions [20].

On Algorand we distinguishe between two types of Smart Contracts: Stateless and Stateful. Stateless Smart Contracts serve as enhanced validators, ensuring transactions meet specified conditions, whereas Stateful contracts handle local and global states, much like traditional smart contracts that build the foundation for decentralized and trustless financial services [20].

2.2. Decentralized Finance on Algorand

Financial applications built on various blockchains, commonly known as DeFi, are an evolving set of applications that replicate existing services from the traditional financial industry. Utilizing smart contracts, these applications on the Algorand blockchain cover a broad spectrum, from decentralized exchanges [5][13][11][7][4][3] and options markets [12][14] to lending protocols [8][5][15][9][6] and tokenized assets [10][2][1].

As of October 2023, the Total Value Locked (TVL) in Algorand's DeFi ecosystem was 43 million USD. The metric, assessed in the native token ALGO, has remained stable on a year-over-year basis while observing growth over a multi-year timeframe. Daily volume and daily active addresses are key metrics for blockchain activity. The daily volume for the last 12 months averaged at 4,050,000 ALGO, and the number of daily active addresses was approximately 33,000[1]. These metrics have been constant on a year-over-year basis. Within the Algorand DEX ecosystem, Tinyman and Pact each report a TVL of 8 million USD. In contrast, HumbleSwap manifests a lesser but still noteworthy TVL of 1.6 million USD. Regarding transactional volume for the month of September 2023, Tinyman (version 1 and 2) indicated an average daily volume of 170,000 USD[2]. Pact followed with 133,000 USD[3], while HumbleSwap recorded 10,000 USD[4]. AlgoFi, a notable DeFi protocol in the Algorand ecosystem, ceased operations in July 2023[5]. Prior to its closure, the protocol averaged a daily trading volume of 90,000 USD[6].

Focusing in the scope of this research specifically on decentralized exchanges within the Algorand DeFi landscape, Tinyman and Pact offer flashloan capabilities. Flashloans are a distinctive DeFi tool that permits users to borrow assets without collateral but within the confines of a single, atomic transaction. This opens the door for sophisticated financial maneuvers such as arbitrage, collateral shifting, and self-liquidation [29].

[1] https://defillama.com/chain/Algorand
[2] https://defillama.com/dexs/tinyman
[3] https://defillama.com/dexs/pact
[4] https://defillama.com/dexs/humble-defi
[5] https://finance.yahoo.com/news/
[6] https://defillama.com/dexs/algofi-swap

2.3. Decentralized Exchanges

Conventional centralized cryptocurrency exchanges manage a daily trade volume exceeding $20 billion[7]. These platforms typically employ a Continuous Limit Order Book architecture, wherein a centralized entity oversees the custody of customer assets and the settlement of trades. Within this exchange design, an enumerated list of all open orders is maintained, enabling real-time matching of buyers and sellers. Orders undergo processing in the sequence they are received, and the centralized exchange operator assumes the role of intermediary, continuously matching and recording orders on the books [27].

In contrast, Decentralized Exchanges (DEXs) leverage smart contracts to facilitate trades, supplanting a traditional order-book with an Automated Market Maker (AMM) model. In this paradigm, a smart contract manages a liquidity pool, which maintains reserves of various tokens and keeps them in a balanced ratio. The AMM enables peer-to-peer trading at predetermined rates, governed by a mathematical construct known as the swap invariant [27][25].

One widely-used variant of the swap invariant is the Constant-Product Market Maker (CPMM), which mathematically enforces that the product of the asset amounts within the liquidity pool remains invariant across trades. This enables the smart contract to perform automatic price discovery. Users interact with the AMM by submitting tokens, incurring a fee, and receiving the purchased tokens from the liquidity pool's reserves. In the current research context, focus is predominantly laid on DEXs implementing the CPMM model. However, it should be noted that there exist three primary types of swap invariants: constant-product, weighted constant-product, and bounded-liquidity constant-product. The arbitrage algorithms discussed in subsequent sections are applicable to all three types [27][25][22].

Price slippage pertains to the variance between the anticipated and actual execution prices in a trade. The term 'expected price-slippage' is used to denote the projected price change, contingent on trading volume and liquidity availability. Because this expected slippage is calculated from a historical blockchain state, intervening fluctuations between transaction submission and execution can result in unanticipated slippage. The cumulative impact of both expected and unexpected slippages determines the overall market impact of a trade [23].

2.4. Convex Optimization

Convex optimization is a specialized subset of mathematical optimization that focuses on the problem of minimizing a convex objective function over a convex domain, subject to convex inequality constraints. The formal mathematical expression of a general convex optimization problem is as follows [31]:

$$\begin{aligned} &\text{minimize} && f_0(\mathbf{x}) \\ &\text{subject to} && f_i(\mathbf{x}) \leq b_i, \quad i = 1, ..., m \end{aligned}$$

In this formulation, x denotes the vector of optimization variables, $f_0 : \mathbb{R}^n \to \mathbb{R}$ serves as the objective function to be minimized, and $f_i : \mathbb{R}^n \to \mathbb{R}$ are the constraint functions. The constants b_i specify the upper bounds for each constraint. An optimal vector x^* achieves the minimum value for the objective function while also satisfying all given constraints [31]. A function $f : \mathbb{R}^n \to \mathbb{R}$ is deemed convex if its domain, $\text{dom}(f)$, is a convex set, and it satisfies the following inequality for all $x, y \in \text{dom}(f)$ and for all $\alpha \in [0, 1]$:

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y)$$

This definition extends naturally to constraint functions f_i, which must also be convex for the problem to be a convex optimization problem [31].

Convex optimization problems can be solved reliably and efficiently. While there is generally no analytical formula for the solution, effective numerical methods exist [31]. For our research purposes, we utilize CVXPY[8], a domain-specific language (DSL) for convex optimization. This framework compiles high-level problem descriptions into low-level standard forms required by solvers and invokes these solvers to retrieve the optimal solution. The available solvers implement various techniques, such as interior-point methods, gradient descent, and simplex algorithms [31].

[7] https://cryptorank.io/exchanges

[8] https://www.cvxpy.org/

3. Related Work

In their seminal paper [27] Daian et al. kickstarted the MEV discourse by exploring value extraction strategies on Ethereum. The paper introduced Priority Gas Auctions (PGAs), where bots bid up fees for transaction priority, and highlighted this as a consensus security risk. In the paper [22] Liyi Zhou and colleagues introduce automated techniques for identifying profitable DeFi trades. One of the key methods employed is the Bellman-Ford algorithm, referred to as DEFIPOSER-ARB. This algorithm excels in detecting DeFi transactions that form cycles, making it particularly adept for arbitrage opportunities. Daniel Engel's work [25] introduces a mathematical framework for the composition of AMMs. Engel extends the concept of AMMs beyond isolated entities, exploring how they can be sequentially or parallelly composed. The paper formalizes composition operators for AMMs that manage multiple asset classes, providing a model for complex, multi-asset trading networks. Subsequent advancements in multi-asset trades and optimal routing on decentralized exchanges were contributed by Guillermo Angeris. His work concentrated on leveraging constrained optimization techniques to improve the efficiency of arbitrage opportunity identification, adding another layer of sophistication to the existing body of research. In his paper [24], published in July 2021, he presents an innovative solution by formulating various trade selection problems as convex optimization problems. This formulation enables reliable and efficient solving, expanding the range of trade possibilities within CFMMs. His subsequent work [17], released in December 2021, tackles the problem of optimally executing orders involving multiple crypto assets across a network of CFMMs. Interestingly, this paper also incorporates the problem of identifying arbitrage opportunities within a network of CFMMs as a special case, or alternatively, certifying the absence thereof.

4. Algorithmic Arbitrage Detection Strategies

In this section, we undertake a comprehensive investigation into computational strategies for arbitrage detection in DEXs. Arbitrage identification can be elegantly modeled using graph theory, where tokens are abstracted as vertices and their conversion rates act as directed edges between these vertices [32]. Our analysis begins with an exploration of traditional graph search algorithms that are widely employed in mainstream financial markets for detecting arbitrage opportunities within directed graphs [32]. However, these conventional methods encounter limitations when transposed to the context of DEXs, primarily due to the unique price invariants introduced by AMMs. To address these limitations, we introduce specialized approximation algorithms that leverage the distinct characteristics of asset-exchange graphs. We employ methodologies such as adjacency matrix transformations to effectively reduce the graph's complexity. This, in turn, narrows down the search space for identifying negative cycles - these cycles are indicative of potential arbitrage opportunities. Finally, we discuss the applicability of convex optimization techniques for optimal routing within bipartite asset-exchange graphs, providing a more efficient alternative for arbitrage detection. The primary focus of this section is to outline the theoretical frameworks and algorithmic methodologies that are critical to the domain of arbitrage detection in decentralized finance. Detailed discussion on the implementation specifics of these algorithms will be reserved for later sections of this paper.

4.1. Negative Cycle Detection Algorithms in Directed Graphs

To identify arbitrage opportunities DEXs on Algorand, we recast the problem within the framework of a directed graph $G = (N, E)$. The objective is to leverage this graph-theoretical representation for negative cycle detection, a classic algorithmic challenge that reveals arbitrage potential [22].

Nodes The set N represents the collection of tokens, where each asset is denoted by $c_i \in N$.

Edges The set E encapsulates all directed edges between nodes. An edge $e_{i,j}$ from asset c_i to asset c_j signifies the availability of a liquidity pool for the involved assets.

Prices For each directed edge $e_{i,j}$, a corresponding price $p_{i,j}$ is defined as the most favorable exchange rate for an infinitesimal quantity of asset c_i converting into asset c_j. This rate can be ascertained by dividing the reserves of asset c_j by those of asset c_j in the liquidity pool offering the highest exchange rate.

Path Paths in the graph outline a sequence of asset conversions. For instance, the composition of edges $e_{1,2}$ and $e_{2,3}$ with weights $p_{1,2}$ and $p_{2,3}$ formulates a path $c_1 \rightarrow c_2 \rightarrow c_3$, which signifies converting 1 unit of token c_1 into

$p_{1,2} * p_{2,3}$ units of token c_3.

Arbitrage An arbitrage opportunity is mathematically represented as a closed path

$$[c_1 \xrightarrow{a_1} c_2 \dots c_{k-1} \xrightarrow{a_{k-1}} c_k]$$

that originates and terminates at the same asset c_1, and where the product of the price-ratios along the path exceeds 1. This condition implies the possibility of initiating a transaction with one unit of asset c_1 and concluding with more than one unit of the same asset, thereby realizing a profit [32].

To formally align the arbitrage detection task with the problem of identifying negative cycles in edge-weighted digraphs, we introduce a logarithmic transformation of the edge weights, which we subsequently negate. This operation recasts the initial objective of identifying a cycle with a product of weights exceeding unity into the equivalent problem of locating a cycle whose sum of transformed weights is negative [32].

$$\begin{aligned} x_1 \cdot x_2 \cdot \ldots \cdot x_k &> 1 \\ \Leftrightarrow \frac{1}{x_1} \cdot \frac{1}{x_2} \cdot \ldots \cdot \frac{1}{x_k} &< 1 \\ \Leftrightarrow \ln\left(\frac{1}{x_1} \cdot \frac{1}{x_2} \cdot \ldots \cdot \frac{1}{x_k}\right) &< \ln(1) \\ \Leftrightarrow \ln\left(\frac{1}{x_1}\right) + \ln\left(\frac{1}{x_2}\right) + \ldots + \ln\left(\frac{1}{x_k}\right) &< 0 \end{aligned} \tag{1}$$

Let the original edge weights be $p_1, p_2, \ldots, p_k$; after applying the transformation, these weights become $\ln\left(\frac{1}{p_1}\right), \ln\left(\frac{1}{p_2}\right), \ldots, \ln\left(\frac{1}{p_k}\right)$. Thus, a cycle in the original graph with a product $p_1 \times p_2 \times \ldots \times p_k > 1$ corresponds to a cycle in the transformed graph where equation (1) holds [32].

In the context of a fully interconnected decentralized exchange network, the underlying directed graph manifests as a complete graph. Although no algorithm currently exists for efficiently identifying the optimal arbitrage opportunity, i.e., the most negative cycle in the directed graph, this paper will explore various approximation techniques aimed at resolving this issue. It is worth noting that the speed of an algorithm for identifying any arbitrage opportunity is of paramount importance. A trader equipped with the fastest algorithm has the potential to systematically exploit a multitude of arbitrage opportunities, thereby outperforming competitors relying on slower algorithms.

Given the critical role of algorithmic speed in identifying arbitrage opportunities, the evolving sparsity of the asset graph introduces another layer of complexity to the problem. As we selectively incorporate more assets into our network, the graph becomes increasingly sparse. This is because lesser-known tokens generally have trading pairs only with the base asset ALGO, resulting in a linear increase in the number of edges as the number of vertices grows, as illustrated by adjacency matrix of the Algorand asset-network in figure B.2 in the appendix. This sparsity not only influences the computational cost but also the efficacy of various algorithms designed for this specific arbitrage detection task. Understanding this relationship is crucial for choosing the most suitable algorithm, considering both its computational complexity and its ability to quickly pinpoint arbitrage opportunities.

The **Bellman-Ford Algorithm**, originally designed to find shortest paths in weighted graphs, can be modified to detect negative cycles with an computational complexity of $O(|V| \times |E|)$. In a sparse graph where the edge growth is nearly linear, the algorithm remains computationally viable even as the graph scales. It is adept at negative cycle detection, making it indispensable for arbitrage opportunity identification in sparsely connected DEX networks [32].

Johnson's Algorithm combines the Bellman-Ford algorithm and Dijkstra's algorithm to find shortest paths between all pairs of vertices. While generally efficient for sparse graphs, it may become less so in our specific context as the graph becomes sparser with the addition of more tokens. Therefore, Johnson's Algorithm can be a viable option for smaller subsets of tokens but may not scale efficiently for the entire, increasingly sparse, DEX network [32].

The **Floyd-Warshall Algorithm** is generally better suited for dense graphs due to its $O(|V|^3)$ time complexity. However, as the graph we are considering grows sparser with the addition of more selective assets, the Floyd-Warshall Algorithm becomes less ideal for real-time operations. Nonetheless, it could still be used for exhaustive offline analysis or when working with a highly selective and therefore smaller set of assets [32].

4.2. Exploring Graph Properties via Adjacency Matrix Transformations

Graph algorithms, especially those for path discovery, can often be challenging to parallelize due to inherent state dependencies during execution. For instance, the discovery of a path in the graph often depends on information about other paths, rendering naive parallelization strategies ineffective. Given the high-frequency and competitive nature of arbitrage in decentralized exchanges on Algorand [33], there is a necessity to focus on specific regions of the asset graph rather than exhaustively exploring the entire space. Prioritization can be made based on several criteria:

1. **Asset Liquidity**: Focusing on assets with higher liquidity can increase the probability of successfully executing arbitrage opportunities.
2. **Arbitrage Lenght**: Limiting the number of hops in a cycle can reduce computational complexity. It also is lowering the chance that other traders might interfere with the arbitrage opportunity.

To facilitate both efficient runtime and potential parallelizability, we utilize an Adjacency Matrix to represent the directed graph of asset pairs. The Adjacency Matrix offers a compact way to represent the directed graph, where each vertex signifies a specific asset and each edge represents a trade opportunity between two assets. It serves as an efficient data structure for the graph, particularly for sparse graphs like our network [32]. By employing an Adjacency Matrix, we can make efficient use of matrix operations to infer about the graph's properties and to detect arbitrage cycles.
By understanding and exploiting the properties of adjacency matrices, we can develop more efficient algorithms tailored to the specific characteristics of the arbitrage graph, leading to quicker and more effective identification of profitable opportunities.

4.2.1. Matrix Exponentiation-Based Path Analysis

Leveraging the properties of adjacency matrices and their exponentiation can provide rich framework for efficient arbitrage detection. However, it's important to note that conclusions about the graph's topological structure are not always straightforward. Some arbitrage cycles may be unattainable within the graph due to the necessity for a specific number of hops, while other cycles could pose computational difficulties due to their high degree of path variability, making them less suitable for certain algorithms.
In DEXs, we employ an adjacency matrix M to illustrate the interconnections between various tradable tokens (illustrated in figure B.2 in the appendix). The process of exponentiating this matrix to a power$n$, denoted as M^n, yields a new adjacency matrix. In this new matrix, the element $(M^n)[i][j]$ encapsulates the sum of the weights for all unique paths from vertex i to vertex j that involve exactly n edges. By leveraging matrix exponentiation at varying powers, one can unearth arbitrage cycles of different lengths within the asset network. If binary weights are used in the original adjacency matrix, the exponentiated matrix reveals the number of distinct paths between vertices i and j, rather than a sum of path weights (as shown in figure C.3 in the appendix for 3-hop arbitrages). This information can be invaluable for analyzing the intricacy of prospective arbitrage cycles within the asset network. This method also allows for strategic filtering of portions of the token graph that are irrelevant for specific arbitrage opportunities, particularly when a given number of hops is unfeasible for successful arbitrage. Furthermore, the knowledge of the number of potential cycles can aid in the efficient parallelization of computational tasks (as will be shown in later sections). The matrix exponentiation approach allows us to focus on efficient subsets of the graph, thereby omitting regions where arbitrage might be improbable due to hop constraints (white regions in figure C.3 in appendix). This granularity facilitates workload parallelization, particularly when balanced against the number of cycles revealed by the diagonal of the exponentiated matrix. This method's applicability extends beyond just cycles; it can also inform us about individual trading paths such as from USDT to USDC.

4.2.2. Trace-Based Discovery of Negative Cycles in Adjacency Sub-Matrices

The introduction of a trace-based discovery paradigm provides a sophisticated mechanism for isolating negative cycles, convertible into actionable arbitrage opportunities. Distinct from previous methods, this approach leverages the properties of adjacency matrices, allowing for parallelizable computations. Importantly, the exchange rates that constitute arbitrage cycles are located along the di-

agonal entries of the matrix. It should be emphasized that the principal diagonal of the original matrix encodes self-cycling of assets, which are not of relevance for arbitrage in our specific application. To overcome this case, a circular leftward shift on the matrix enables the extraction of each cycle's information along the diagonal. The trace of each iteratively transformed matrix serves as a criterion for identifying profitable arbitrage cycles. This methodology is amenable to generalization for cycles of arbitrary length n. Notably, for 2-hop cycles, the Hadamard product of the adjacency matrix A and its transpose A^T streamlines the identification process. Any entry in the resultant matrix that exceeds 1 signifies a potential arbitrage route. Currently, this elegant form of simplification is not extendable to cycles of greater length.
The value of this trace-based method lies in its computational efficiency, which becomes particularly essential for real-time arbitrage hunting in Decentralized Exchanges. This is especially beneficial given the sparsity characteristics of the asset trading graph, allowing for more focused and quicker analyses. Most critically, this technique can be dynamically applied to tokens for which the state of the respective liquidity pools change due to incoming transactions. This creates an immediate and direct way to compute possible arbitrages that might arise as a consequence of transactions which can be backrun, adding another layer of precision and reactivity to the arbitrage detection process.

4.3. Optimal Routing in Bipartite Graphs

Finding arbitrage opportunities may seem straightforward at first glance; however, identifying significantly profitable transactions across a diverse array of tokens and decentralized exchanges is a more intricate task. Traditional methods tend to rely on greedy search algorithms that focus solely on immediate conversion rates, often ignoring the liquidity constraints present in AMMs. In this section, we elevate the discussion to optimal arbitrage, particularly within the context of CFMMs. Supported by Guillermo Angeris' contributions, we apply convex optimization techniques. Importantly, these techniques do not merely identify arbitrage opportunities; they enable us to find the optimal set of trades that maximize overall arbitrage profit across a multitude of CFMMs [24][17].

4.3.1. Definition of the Search Space

The search space for our optimization problem is designed to address the complex task of optimally executing an order that involves multiple tokens across a network of CFMMs. Earlier graph representations were limited, capturing only the maximum exchange rates and losing other essential information. This necessitates a novel structural approach. We model our ecosystem consisting of m CFMMs, each indexed by $i = 1, \ldots, m$, interacting with a universe of n distinct tokens, indexed by $j = 1, \ldots, n$. The number of tokens traded by each CFMM i is represented by n_i, constrained by $2 \leq n_i \leq n$ [17]. This setup can be abstracted as a bipartite graph. On one side of the bipartition, we have m vertices representing the CFMMs. The other side contains n vertices representing the different tokens. An edge is drawn between a vertex from the CFMM group and a vertex from the token group if and only if that particular CFMM trades the associated token [17].

4.3.2. CFMM Trading Functions

In the case of a CFMM denoted as i, we use the terms Δ_i and Λ_i to refer to the tendered and received asset-vectors, respectively. These vectors are signifying the quantities of tokens transacted with the CFMM. For a trade (Δ_i, Λ_i) to be deemed valid by CFMM i, it must satisfy the equation

$$\phi_i(R_i + \gamma\Delta_i - \Lambda_i) = \phi_i(R_i),$$

where ϕ is the **trading function** of the CFMM. R_i are the current reserves of the CFMM, and $\gamma_i \in (0, 1]$ is the associated trading fee. An example of such a trade function is the geometric mean, applicable to all DEXs on Algorand [24][17].

$$\phi_i(R) = (R_1 \cdots R_{ni})^{1/ni}$$

The asset-vectors Δ_i and Λ_i of one DEX can be summed across all CFMMs, producing a **network trade vector**. The resulting n-vector Ψ encapsulates the net amount of each token transacted across the network. We define a **utility function** $U : \mathbb{R}^n \to \mathbb{R} \cup \{-\infty\}$ that quantifies the desirability of a trade Ψ to a searcher. The utility function can be specified as $U(z) = \pi^T z$, where $\pi \in \mathbb{R}$. In this context, π_i is interpreted as the market-price associated with each token i [17].

4.3.3. Stating the Constrained Optimal Routing Problem

The convex optimization problem we aim to solve is defined over a network of CFMMs. Each CFMM, denoted by i, has its own trading function ϕ_i, trading fees γ_i, and reserve amounts $R_i \in \mathbb{R}^+$. Additionally, we define A_i as the mappings of tokens to their corresponding CFMMs. Our primary objective is to maximize the utility function $U(\Psi)$, which quantifies the benefit gained from executing a series of trades across the network [17].
The constraints that must be satisfied for each trade are governed by each CFMM's swap invariant, trading fees, and reserve amounts. Our optimization problem together with these constraints and the objective function can be formally stated as follows [24][17]:

$$\begin{aligned} \text{maximize} \quad & U(\Psi) \\ \text{subject to} \quad & \Psi = \sum_{i=1}^{m} A_i(\Lambda_i - \Delta_i) \\ & \phi_i(R_i + \gamma_i\Delta_i - \Lambda_i) \geq \phi_i(R_i), \quad i = 1, \ldots, m \\ & \Delta_i \geq 0, \Lambda_i \geq 0, \quad i = 1, \ldots, m. \end{aligned}$$

We aim to identify a set of valid trades that would maximize the trader's utility, given the constraints. This forms the crux of what we refer to as the Optimal Routing Problem. Solving this optimization problem yields the n-vector Ψ, encapsulating the net amount of each token traded across the network. By sifting through the trade vectors of each CFMM, one can discern the specific trades that constitute this optimal network trade [24][17].
While a detailed proof of the convex nature of this problem is beyond the scope of this work, we refer readers to [17] for a comprehensive proof. Regardless of the overall complexity behind the optimization, it can be solved with domain-specific languages like CVXP, even for large networks [31].

4.4. Transaction Input Optimization for Arbitrage Execution

Identifying negative cycles through established graph algorithms suggests the existence of arbitrage opportunities within a graph. However, these opportunities become actionable only when the initial input amount for the trade in the arbitrage loop is optimized. Therefore, the overarching aim of input optimization is to maximize the profitability of the arbitrage opportunity at hand [22]. In the following, we explore various methodologies to achieve this granular level of input optimization.

- **Binary Search-Based Input Optimization** An initial method involves utilizing binary search algorithms, a technique previously explored in [22]. This approach leans on previously formulated equations for CPMMs [26]. The output for a given input in a arbitrage cycle can thus be calculated. The efficiency of this method can be further elevated by judiciously choosing the initial search boundaries.

- **Convex Optimization-Based Input Tuning** A more advanced method involves the application of convex optimization techniques. In this context, the search space of the convex optimization is tailored to contain only the CFMMs that are pertinent to the assets involved in the targeted arbitrage cycle. In contrast to heuristic methods, convex optimization is adept at isolating the exact optimal trade inputs by directly solving for the global optimum [17].

The convex optimization approach is generally favored for its ability to precisely find the global minimum and its superior runtime efficiency, making it the go-to method for optimal arbitrage execution.

5. Implementation & Infrastructure

The application project implementation accompanying this paper is engineered to function in two operational modes: real-time processing and batch processing of historical data. While the current implementation focuses solely on batch processing for historical states, this paper further explores how real-time processing could be efficiently implemented.

5.1. Infrastructure Set-Up

Utilizing third-party node APIs, such as those offered by Algonode.io, was deemed unsuitable for real-time analysis in our use-case. This necessitated the set-up of a dedicated participation node specifically for the application project. The Algorand Foundation recommends a hardware configuration comprising 8 vCPUs, 16 GB of RAM, a 100 GB NVMe SSD or its equivalent, and a 1 Gbps low-latency internet connection [20].

In order to minimize latency further, our infrastructure was configured to connect to a geographically nearby relay node hosted at Technical University Munich. Connection to these nodes is specified during the initiation of the "`goal node start -p ipaddress:port`" command. Multiple relay nodes can be managed by providing a semicolon-separated list of nodes [20].
For arbitrage detection, we employed a client listening mechanism connected directly to our node, designed to quickly react to detected opportunities, as outlined in the next section.

5.2. Implementation

This section elaborates how the theoretical concepts discussed earlier can be engineered into a functional arbitrage system. From real-time and batch data retrieval to algorithmic decision-making, this part of the paper provides an in-depth look into the workflow.

5.2.1. Data Retrieval

In the real-time setup, we leverage the API capabilities of our Algorand node. Our client continuously listens to our node by invoking the `/v2/status/wait-for-block-after/round` endpoint[9]. When the node receives this HTTP request, it sets up an internal wait channel that unblocks upon reaching the desired round, effectively notifying us of new blocks. This way, we are asynchronously alerted about new blocks, offering potential transactions to backrun. Multiple incoming requests are managed by an internal queue in the Algorand node. Further advancements could involve customizing the node's source code to directly notify us of incoming transactions. Such enhancements could bypass transaction validation and concentrate solely on parsing DEX application calls.
For batch data retrieval, we expect data to be available in a format where each liquidity pool's reserves are given for each block of a certain time period.

5.3. Implementation and Analysis of Arbitrage Strategies

Prior to the discussion on arbitrage algorithms, it is essential to clarify the initial setup regarding which tokens will be used for realizing profits and how their prices are sourced. For the purposes of this research, *ALGO, goBTC, PEPE*, and certain stablecoins such as *USDT, USDC* have been chosen as the tokens in which to take profits. As for the price data, we source real-time valuations for the tokens through the Binance API[10].
Our real-time arbitrage detection algorithm operates continuously, constantly monitoring incoming blocks for swap transactions. Upon identifying a swap, it updates the adjacency matrices representing the maximal exchange rates for each feasible trading cycle (both 2 and 3-hop). It then iteratively shifts and evaluates these matrices to look for negative cycles, which indicate profitable arbitrage opportunities. If such a cycle is found, the algorithm uses convex optimization to find the optimal inputs and then executes the arbitrage trade.

In a system where block times are commonly situated within a 3 to 4-second window, there exists a unique avenue to search for maximally profitable arbitrage cycles as opposed to greedily. This segment elucidates an al-

Algorithm 1 Real-time Arbitrage Detection for 2,3 hops

```
Initialize
set up adjacency matrices with maximal exchange rates
for feasible 2,3-hop cycles
while True do
    parse transactions of incoming block
    if swaps found then
        retrieve affected assets
        Update affected adjacency matrices
        for each affected adjacency matrix do
            for each column do
                perform circular left shift
                calculate trace of matrix
                if negative cycle then
                    find optimal inputs
                    execute arbitrage
                end if
            end for
        end for
    end if
end while
```

[9] https://github.com/algorand/go-algorand/algod/api/server/v2/

[10] https://www.binance.com/en/binance-api

gorithm tailored for batch processing, engineered to discover the highest-yielding arbitrage opportunities within these quasi-constant block-time intervals. This algorithm navigates the graph complexity of base token ALGO, by allowing parallelized operations that start from varying tokens. It identifies all possible n-hop cycles beginning and ending with a specific asset. Importantly, the actual arbitrage is not restrict to the starting point initially chosen. The rationale behind this is that arbitrage cycles are inherently cyclical; thus, the starting point is arbitrary, allowing for multiple interpretations of the cycle's beginning and end, as long as the product of exchange rates is higher than 1.

This algorithm is designed to quickly identify the most profitable arbitrage opportunities within the typical block-time range. It leverages parallel processing and reduced network graphs for efficient computation.

Algorithm 2 Arbitrage Detection for Batch Processing

```
for non-base tokens do
    get n-hop cycles for token
    get pools building the cycles
    for each profit-token in cycle do
        if Bellman-Ford then
            build directed graph out of pools
            find negative cycles
            find input using convex optimization
        else
            run optimization on pool subset
            parse trade-vectors to infer order
        end if
        if arbitrage found then
            execute arbitrage
        end if
    end for
end for
```

6. Analysis and Results

We successfully implemented the batch processing algorithm to serve as a proof of concept for our arbitrage system. Our data scraping modules targeted multiple DEXs on the Algorand blockchain, including Tinyman versions 1 and 2, Pact, and HumbleSwap yielding up to 62 tokens. It is noteworthy that due to the high latency of the software SDK for internal operations, we chose not to include HumbleSwap in our analysis. A comprehensive visualization of the network, delineating the relations between various assets and DEXs, is provided in the Appendix (see figure A.1 in appendix). This visual representation serves to elucidate the complexity and interconnectedness of the DeFi landscape on Algorand.

We conducted a trial run of our algorithm on a single day's worth of data from October 5, 2023. The algorithm identified a total of 524 events where asset swaps took place in liquidity pools, as captured by our DEX reserve scraper. The algorithm successfully detected arbitrage opportunities at the granularity of individual blocks. However, the profitability from these opportunities was generally marginal, mostly manifesting in microcents. This limitation likely stems from our decision to focus at this point in time solely on maximal exchange rates, thereby discriminating against liquidity.

Our preliminary results suggest several avenues for future research and improvement. For instance, our current algorithm restricts its search to cycles; expanding the scope to include paths would allow for a more thorough investigation of arbitrage opportunities involving equivalent assets like stablecoins or wrapped ALGO. Additionally, starting the search from less liquid, smaller coins appears to yield more arbitrage opportunities, suggesting that initiating with a flash loan could be a promising strategy.

7. Conclusion

This work serves as an investigation into the intricacies of arbitrage opportunities within the Algorand decentralized financial ecosystem. By presenting advanced methodologies and implementing them in a bespoke environment, it successfully identifies actionable arbitrage avenues for a select set of asset pairs. While the analysis is limited to short periods and specific token pairs, the findings demonstrate the potential for profitable arbitrage opportunities.

The examination of the techniques—ranging from graph-based algorithms for negative cycle detection to sophisticated input optimization strategies —establishes a framework for future research. The study's limitations in scope also point toward the expansive possibilities for future work. By broadening the asset pairs and time horizons

considered, alongside incorporating emerging algorithms and technologies, we anticipate a marked increase in arbitrage profitability.

In summary, this paper serves as a blueprint in the rapidly evolving landscape of Algorand-based decentralized finance, offering both theoretical insights and practical frameworks for further exploration and enhancement.

References

[1] Circle internet financial limited. *(https://www.circle.com/en/usdc)*, 2023.

[2] Tether limited. *(https://tether.to/en/)*, 2023.

[3] Alammex. *(https://www.alammex.com/)*, 2023.

[4] Algodex. *(https://algodex.com/)*, 2023.

[5] Algofi. *(https://www.algofi.org/)*, 2023.

[6] Algorai finance. *(https://algorai.finance/)*, 2023.

[7] Defly. *(https://defly.app/)*, 2023.

[8] Folks finance. *(https://folks.finance/)*, 2023.

[9] Gard protocol. *(https://algogard.com/)*, 2023.

[10] Algomint. *(https://algomint.io/)*, 2023.

[11] Pact finance. *(https://www.pact.fi/)*, 2023.

[12] Silo protocol. *(https://docs.silodefi.com/)*, 2023.

[13] Tinyman. *(https://tinyman.org/)*, 2023.

[14] Venue one. *(https://venue.one/)*, 2023.

[15] Yieldly finance. *(https://yieldly.finance/)*, 2023.

[16] Fabrice Benhamouda. Rransaction priority and finality. *Algorand Forum (https://forum.algorand.org/t/transaction-priority-and-finality/6383/2)*, 2022.

[17] Guillermo Angeris; Tarun Chitra; Alex Evans. Optimal routing for constant function market makers. *23rd ACM Conference on Economics and Computation*, 2022.

[18] Hu Facundo, Carrillo;Elaine. Mev in fixed gas price blockchains: Terra classic as a case ofstudy. *arXiv Preprint*, 2023.

[19] facuzeta. Frp-21: Mev in fixed gas price blockchains. *Falsbots Collective Forum*, 2023.

[20] Algorand Foundation. Algorand developer portal. *https://developer.algorand.org/docs/get-details/*, 2023.

[21] Ethereum Foundation. https://ethereum.org/en/l. *https://ethereum.org/en/*, 2023.

[22] Liyi Zhou; Kaihua Qin; Antoine Cully; Benjamin Livshits; Arthur Gervais. On the just-in-time discovery ofprofit-generating transactions in defi protocols. *IEEE Symposium on Security and Privacy*, 2021.

[23] Liyi Zhou; Kaihua Qin; Christof Ferreira Torres; Duc V Le; Arthur Gervais. High-frequency trading on decentralized on-chain exchanges. *IEEE Symposium on Security and Privacy*, 2021.

[24] Agrawal; Alex Evans Guillermo, Angeris; Akshay. Constant function market makers: Multi-asset tradesvia convex optimization. *Handbook on Blockchain, Springer*, July 2021.

[25] Daniel Engel; Maurice Herlihy. Composing networks of automated market makers. *3rd ACM Conference on Advances in Financial TechnologiesSeptember*, 2021.

[26] Gebele Jonas. Analysis of maximal extractable value on the algorand blockchain. 2023.

[27] Philip Daian; Steven Goldfeder; Tyler Kell; Yunqi Li; Xueyuan Zhao; Iddo Bentov; Lorenz Breidenbach; Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, minerextractable value, and consensus instability. *2020 IEEE Symposium on Security and Privacy (SP)*, 2020.

[28] Jing Chen; Silvio Micali. Algorand theoretical paper. *arXiv:1607.01341*, 2017.

[29] Dabao Wang;Siwei Wu;Ziling Lin;Lei Wu;Xingliang Yuan;Yajin Zhou;Haoyu Wang;Kui Ren. Towards a first step to understand flash loan and its applications in defi ecosystem. *SBC '21: Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing*, 2021.

[30] J.David Richardson. Some empirical evidence on commodity arbitrage and the law of one price. *Journal of International Economics, Volume 8, Issue 2*, 1978.

[31] Stephen Boyd; Lieven Vandenberghe. Convex optimization. *Cambridge University Press*, 2004.

[32] Robert Sedgewick; Kevin Wayne. Algorithms - forth edition. *Pearson Education, Inc.*, 2011.

[33] Burak Öz; Jonas Gebele; Filip Rezabek; Felix Hoops; Florian Matthes. A first study of mev on an up-and-coming blockchain: Algorand. *arXiv preprint arXiv:2308.06513*, 2023.

Appendix A. Graphical Representation of Algorand DEX Trade Network

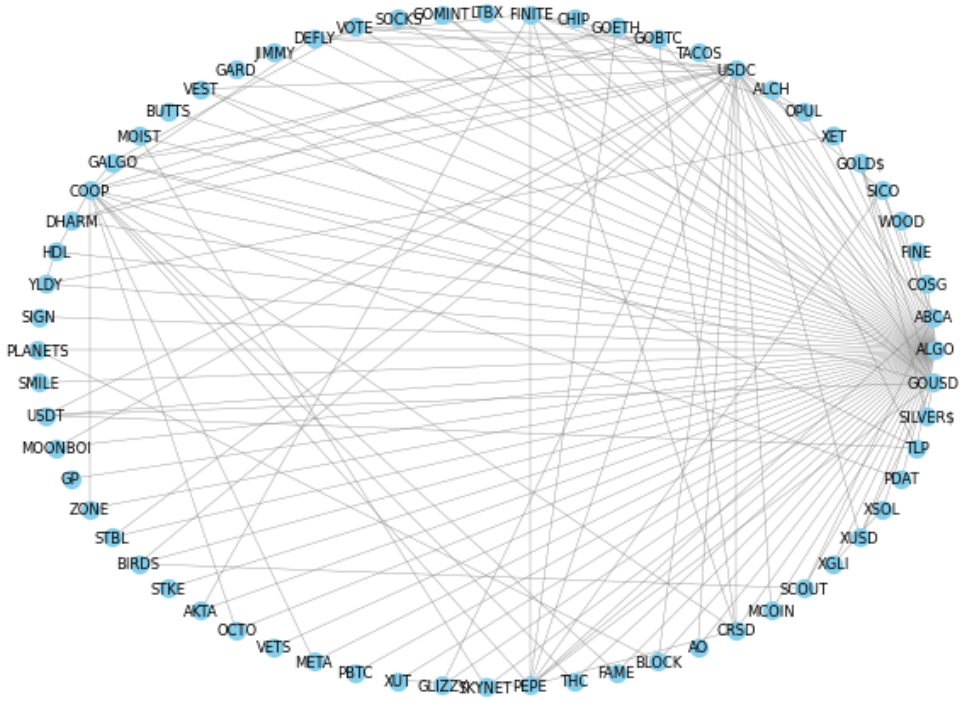

Figure A.1: Network Graph Depicting Trade Pathways Among Algorand DEXs

Appendix B. Matrix Representation of Algorand DEX Trade Network

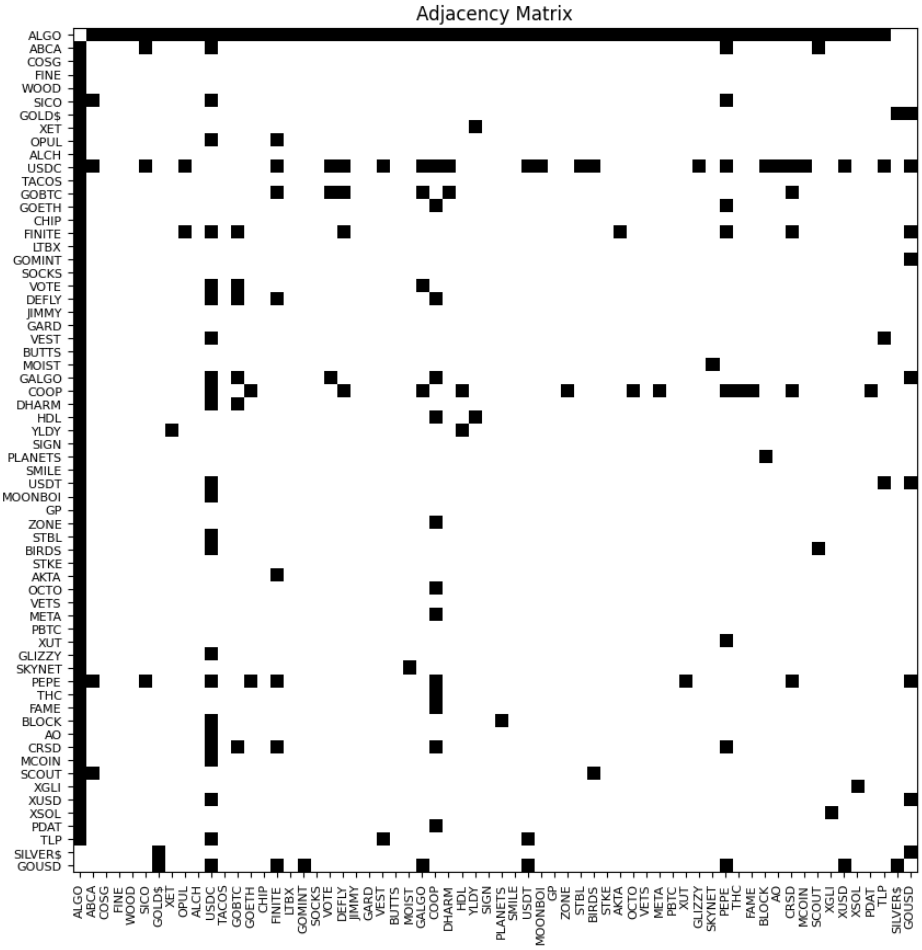

Figure B.2: Binary Adjacency Matrix Representation of Algorand DEX Trade Pathways

Appendix C. Quantitative Analysis of Trade Pathways using Matrix Exponentiation

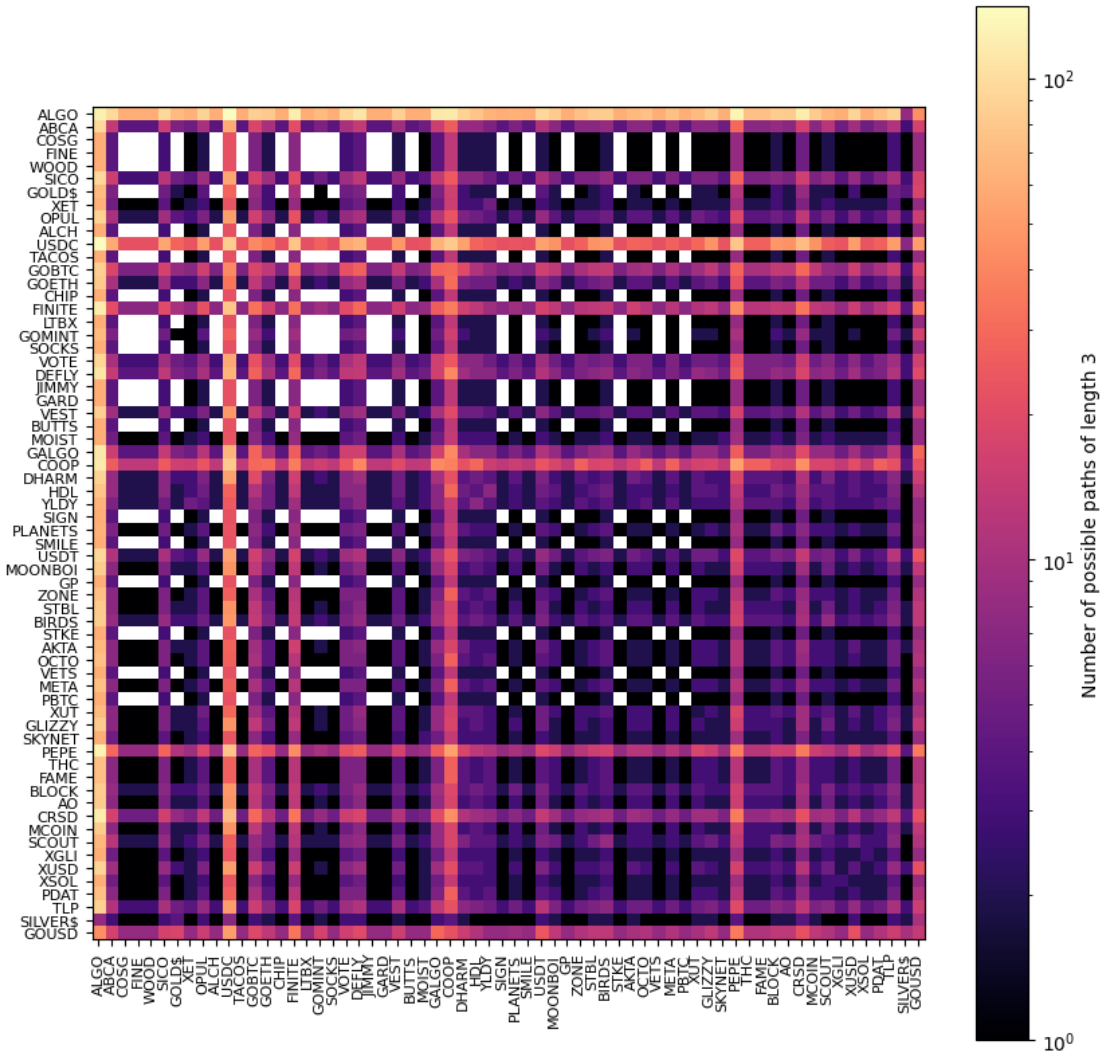

Figure C.3: Matrix Exponentiation Analysis of Unique Path Lengths between Algorand DEX Assets