universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Assessing wash trades in large-scale transaction networks

verfasst von | submitted by

Markus Zimmball

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 921
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Informatik
Betreut von | Supervisor:	Ass.-Prof. Dott.ssa Dott.ssa.mag. Yllka Velaj PhD

Acknowledgements

I would like to thank Professor Yllka Velaj, who sparked my interest in dynamic graph networks, and would like to express my genuine appreciation for her exceptional guidance and support, as well as the freedom she gave me to choose and explore a topic that truly captivates my interest.

Additionally, I am deeply grateful to Tim, Matthias, and Tobin, who generously shared their time to critique my work and engage in thoughtful, constructive conversations.

Abstract

Wash trading is a form of market manipulation where assets circulate through controlled accounts to create artificial trading volume. This practice poses significant challenges to financial market integrity, particularly in emerging digital asset ecosystems. This thesis presents DSCENT (Dynamic Streaming Cycle Enumeration with Temporal constraints), a novel algorithm designed to detect temporal cycles in continuous transaction streams, enabling real-time identification of potential wash trading patterns. Unlike existing batch-processing approaches such as 2SCENT, DSCENT processes transactions as they arrive using a streaming architecture with runtime generation, parallel execution, and adaptive memory management. The algorithm was evaluated on two real-world datasets: the Meebits NFT collection and 1.8 billion Ethereum blockchain transactions. Analysis of the Meebits ecosystem revealed extensive cyclic trading patterns coinciding with the LooksRare token reward period (January–March 2022), when the NFT marketplace incentivized trading volume through token distributions, with approximately 50% of detected components representing trivial two-party exchanges. Performance testing on Ethereum data demonstrated the algorithm's scalability, processing the entire dataset in under two days while maintaining memory consumption below 87 GB. DSCENT achieved order-of-magnitude improvements over 2SCENT in both runtime (15x speedup) and memory efficiency (10x reduction). The identification of optimal parameters balancing detection capability with computational efficiency provides practical guidance for deployment. While limitations exist in detecting sophisticated manipulation schemes and validating ground truth, this work contributes essential tools for maintaining market integrity in decentralized financial systems where traditional surveillance mechanisms are inadequate.

Kurzfassung

Beim Wash Trading handelt es sich um eine Form der Marktmanipulation, bei der Vermögenswerte gezielt durch kontrollierte Konten zirkulieren, um künstliches Handelsvolumen zu erzeugen. Diese Praktik gefährdet die Integrität von Finanzmärkten erheblich, insbesondere in neu entstehenden digitalen Vermögenswert-Ökosystemen. Die vorliegende Arbeit präsentiert DSCENT (Dynamic Streaming Cycle Enumeration with Temporal constraints), einen neuartigen Algorithmus zur Erkennung temporaler Zyklen in kontinuierlichen Transaktionsströmen, der potenzielle Wash-Trading-Muster in Echtzeit identifiziert. Anders als bestehende Ansätze wie 2SCENT, die Transaktionen in Chargen verarbeiten, verarbeitet DSCENT eingehende Transaktionen unmittelbar mittels einer Streaming-Architektur, die auf dynamischer Generierung, paralleler Ausführung und adaptivem Speichermanagement basiert. Die Evaluierung des Algorithmus erfolgte anhand zweier realer Datensätze: der Meebits NFT-Kollektion sowie 1,8 Milliarden Ethereum-Blockchain-Transaktionen. Die Analyse des Meebits-Ökosystems offenbarte umfangreiche zyklische Handelsmuster, die zeitlich mit der LooksRare-Token-Belohnungsperiode (Januar-März 2022) zusammenfielen — in dieser Phase regte der NFT-Marktplatz durch Token-Ausschüttungen das Handelsvolumen an. Dabei stellten etwa 50% der erkannten Komponenten triviale Zwei-Parteien-Austausche dar. Die Leistungsbewertung anhand der Ethereum-Daten belegt die Skalierbarkeit des Algorithmus: Der gesamte Datensatz wurde in unter zwei Tagen verarbeitet, wobei der Speicherverbrauch 87 GB nicht überstieg. Gegenüber 2SCENT erzielte DSCENT erhebliche Verbesserungen — sowohl hinsichtlich der Laufzeit (15-facher Geschwindigkeitsgewinn) als auch der Speichereffizienz (10-fache Reduktion). Die Identifikation optimaler Parameter, welche Erkennungsfähigkeit und Recheneffizienz ausbalancieren, liefert praktische Leitlinien für den Einsatz in realen Systemen. Trotz bestehender Einschränkungen bei der Erkennung hochentwickelter Manipulationsschemata und der Validierung eindeutiger Referenzdaten leistet diese Arbeit einen wesentlichen Beitrag zur Wahrung der Marktintegrität in dezentralisierten Finanzsystemen, wo traditionelle Überwachungsmechanismen an ihre Grenzen stoßen.

Contents

List of Tables

List of Figures

List of Algorithms

1 Introduction and Motivation

The detection of fraudulent activities in financial transaction networks, where nodes represent entities and edges capture timestamped and weighted transactions, has become increasingly critical as modern financial systems generate high-volume, high-velocity transactions that present unprecedented challenges for maintaining market integrity [McK24, ARO+22, ZKX23]. Fraudulent schemes have evolved alongside technological advancement, and their effective control requires advanced detection mechanisms to identify and prevent illicit activities that undermine fair market operations [ARO+22, ZKX23, BIA24].

Among various forms of market manipulation, wash trading poses significant threats across diverse asset classes. The UK Financial Conduct Authority (FCA) defines wash trades as "a sale or purchase of a qualifying investment where there is no change in beneficial interest or market risk, or where the transfer of beneficial interest or market risk is only between parties acting in concert or collusion, other than for legitimate reasons" [Fin13]. This deceptive practice manifests across multiple domains: in housing markets through property flipping schemes where properties are traded in circular patterns among colluding parties to artificially inflate appraisal values [Fed25, ABDY15], and in financial asset markets where actors create false impressions of trading volumes to mislead investors about asset liquidity and demand [IT18, VW21, MMMN23, CLTY23]. The challenge extends to emerging digital asset ecosystems, particularly non-fungible tokens (NFTs) and cryptocurrency networks. These decentralized environments present unique detection challenges due to their relative lack of regulation, ease of account creation, and pseudonymous transaction capabilities that can obscure coordinated fraudulent activities across multiple accounts [VW19, Ant22].

The identification of fraudulent activities through cycle detection in transaction networks has emerged as a fundamental approach in financial forensics. Cycle detection is fundamental because legitimate market activity rarely produces systematic circular patterns, making the presence of cycles a powerful signal of deliberate coordination and manipulation [CLC+16, QCQ+18]. In wash trading schemes, assets frequently circulate through networks of controlled accounts, creating detectable cyclical patterns that distinguish fraudulent activity from legitimate market behavior. These cycles serve as fingerprints of manipulation, providing investigators with concrete evidence of coordinated artificial trading [CLC+16, BAI23]. In digital asset markets, cycle detection algorithms can identify networks where the same NFT or cryptocurrency moves through multiple accounts controlled by the same actor, creating false impressions of market activity. These patterns often involve assets returning to their original holders through complex transaction paths, generating artificial volume metrics that deceive potential investors about genuine market interest and liquidity [TVH25, WWY+23].

Detecting these patterns in modern financial networks necessitates automated algorithmic approaches for effective fraud detection. Traditional manual investigation methods cannot analyze the massive transaction datasets generated by contemporary financial systems where millions of transactions occur daily [McK24, KG24, DSE25]. Real-time pattern recognition systems have become essential tools for maintaining market integrity [BIA24, CZWL24]. The development of effective cycle detection algorithms represents a critical research frontier requiring integration of insights from graph theory, machine learning, and financial economics. These systems must balance computational efficiency with detection accuracy, identifying genuine fraudulent cycles while minimizing false positives that could disrupt legitimate trading activities. The challenge is compounded by adaptive fraudulent actors who continuously evolve their strategies to evade detection [CZWL24, PGM+21]. Furthermore, effective fraud detection systems must provide interpretable results that can support regulatory enforcement actions and legal proceedings, highlighting the importance of developing algorithms that are both technically robust and practically applicable in real-world financial forensics contexts [CZWL24].

This thesis presents DSCENT (Dynamic Streaming Cycle Enumeration with Temporal constraints), a novel streaming solution for cycle detection. While existing approaches require complete datasets and batch processing, DSCENT processes transactions as they arrive, enabling immediate detection of cyclic and potential fraudulent patterns as they emerge. The algorithm introduces five key innovations:

- **Enhanced block transaction processing**: DSCENT employs techniques for handling simultaneous edges within a block, enabling more efficient processing of temporal dependencies that occur at the same timestamp.
- **Streaming architecture**: Transactions are processed as they arrive, maintaining only the necessary temporal window of data rather than requiring the entire graph to be available upfront.
- **Dynamic generation**: Cycle candidates are dynamically generated and determined to be ready for processing based on temporal windows, allowing the algorithm to yield results continuously as the stream progresses.
- **Parallel execution model**: Finalized seeds are offloaded to a process pool for parallel execution, exploiting the independence of seed processing to achieve significant speedup.
- **Adaptive pruning strategy**: The algorithm employs a pruning mechanism that maintains memory efficiency while ensuring no cycles are missed.

By extending the theoretical foundations of cycle detection to the streaming domain, DSCENT maintains rigorous correctness guarantees while achieving the performance characteristics demanded by modern financial markets. Through comprehensive evaluation on two real-world datasets, we demonstrate that DSCENT not only matches the detection accuracy of existing methods but fundamentally transforms when and how fraudulent cycles can be identified — shifting from post-hoc analysis to real-time prevention.

2 Related Work

Temporal graphs, where edges are associated with timestamps representing the time of interaction, provide a natural model for dynamic systems across numerous domains. Research on temporal graphs encompasses diverse computational problems including reachability analysis, path finding, connectivity, exploration, and pattern detection [Mic16]. These problems arise in applications ranging from communication networks and transportation systems to social network analysis and financial transaction monitoring. Among these computational challenges, cycle detection has emerged as particularly relevant for identifying fraudulent patterns in financial networks, where circular flows of assets often indicate manipulative trading behavior.

2.1 Classical Cycle Detection Algorithms

The theoretical foundation for cycle detection in directed graphs traces back to the 1970s, with several seminal contributions establishing the groundwork for modern approaches. The early methods built upon basic depth-first search (DFS) traversal, progressively incorporating advanced pruning strategies to improve computational efficiency.

The Johnson algorithm represents a pivotal advancement in cycle enumeration for directed graphs, efficiently identifying all simple cycles through a sophisticated "block and unblock" mechanism that prevents node revisitation while managing feasible cycle completion pathways [Joh75]. This algorithm employs depth-first search specifically tailored for directed graph structures, establishing a foundation for subsequent algorithmic developments.

Developed in parallel to Johnson's work, the Read-Tarjan algorithm implements a backtracking technique that relies on vertex ordering rather than the blocking/unblocking mechanism [RT75]. While functionally similar to Johnson's approach in utilizing depth-first search principles, the Read-Tarjan algorithm's simpler backtracking strategy of unmarking vertices and removing them from a stack achieves cycle enumeration through its ordering constraints rather than Johnson's dynamic blocking mechanism. While these classical algorithms establish fundamental principles for cycle enumeration, they operate on static graphs and lack temporal awareness.

2.2 Temporal and Constrained Cycle Detection

The detection of fraudulent cycles in financial networks requires algorithms that can identify patterns within specific temporal windows, as manipulative trading schemes typically manifest as coordinated transactions occurring within constrained time periods.

Peng *et al.* developed the BC-DFS (Barrier-Constrained depth-first search) algorithm, which incorporates sliding window techniques with hop-constrained cycle enumeration [PZL+19]. The barrier constraint imposes a maximum timestamp threshold that limits the temporal span of enumerated cycles. This approach constrains cycle length to maintain computational feasibility, as exhaustive cycle enumeration exhibits exponential complexity in dense financial networks.

Building upon hop constraints, Kumar and Calders introduced the 2SCENT algorithm, which targets cycles that form within constrained time windows in dynamic graph structures [KC18]. This approach operates through a two-stage process: the first stage collects reverse-reachability summaries that track which nodes can reach a given target node at different times, and the second stage utilizes these summaries in a temporally constrained depth-first search. The algorithm reduces memory consumption by bundling temporal paths, enabling efficient processing of time-series transaction data while maintaining detection accuracy for temporal fraud patterns. Like BC-DFS, 2SCENT operates in batch mode and requires the complete dataset upfront. While the sliding window technique provides some temporal locality, the algorithm cannot process transactions as they arrive in a true streaming fashion.

2.3 Performance and Scalability Optimization

As transaction networks continue to expand in scale and complexity, research focus has shifted toward developing efficient query mechanisms that can support real-time cycle detection. Qiu *et al.* implemented a comprehensive index structure within Alibaba Group's infrastructure, introducing a streaming algorithm specifically designed to minimize computationally intensive exploration paths [QCQ+18]. Their Hot Point Index (GraphS) provides efficient storage and retrieval capabilities, enabling practical deployment of cycle detection algorithms in production environments. However, their approach relies on identifying and indexing high-degree hub nodes to accelerate cycle detection, assuming relatively stable network topology. This index-based strategy is not resilient to malicious actors who frequently create new accounts, abandon suspicious addresses, and shift trading patterns to evade detection.

The parallelization of cycle detection algorithms has become increasingly important for handling large-scale financial networks. Blanuša *et al.* conducted extensive benchmarking and review of existing parallelization techniques for various cycle enumeration algorithms in both coarse-grained and fine-grained parallel contexts [BAI23]. Coarse-grained parallelization distributes large, independent units of work (such as entire connected components) across processors with minimal synchronization overhead, while fine-grained approaches parallelize smaller operations, potentially requiring more frequent coordination between threads. Their investigation includes innovative strategies such as copy-on-steal work distribution, where processors duplicate necessary data structures when acquiring tasks from other workers' queues rather than sharing them directly. This approach eliminates synchronization contention at the cost of increased memory consumption.

DSCENT employs a straightforward fine-grained parallelization strategy where inde-

pendent candidate sets are distributed across a process pool for parallel exploration. While this naive approach provides substantial speedup for the workloads examined in this thesis, the parallelization techniques reviewed by Blanuša *et al.* could be integrated into DSCENT to further optimize performance for specific network characteristics.

2.4 Applications in Digital Asset Markets

Digital asset trading operates on blockchain infrastructure, which fundamentally shapes the environment for cycle detection algorithms. The immutable and transparent nature of blockchain ledgers provides publicly accessible datasets of real economic activity, offering researchers complete transaction histories with genuine financial incentives at stake [Nak08]. Ethereum's architecture particularly influences cycle detection requirements in digital asset markets. Introduced in 2015, Ethereum extended blockchain functionality through smart contracts — self-executing programs that enable automated trading without intermediaries [Woo14]. This programmability has enabled complex trading ecosystems, particularly in non-fungible token (NFT) markets, where decentralized exchanges facilitate rapid, programmatic trading that can be exploited for wash trading schemes [LC25]. The Merge, which was implemented as a hard fork by designating a consensus block that deliberately lacked a valid nonce (the brute-force solution to the cryptographic puzzle required in mining), completed on September 15, 2022 and transitioned Ethereum from Proof-of-Work (where miners compete to solve computationally intensive problems to validate blocks) to Proof-of-Stake (where validators are selected to propose blocks based on their staked ETH holdings) consensus, establishing highly consistent block times of approximately 12 seconds [KM23]. The decentralized nature of blockchain networks presents specific challenges for cycle detection algorithms: the ease of creating new addresses enables malicious actors to rapidly establish complex trading networks, transaction pseudo-anonymity complicates entity resolution, and the absence of regulatory gatekeepers means transactions execute immediately without approval or the possibility of reversal, allowing fraudulent trading patterns to complete in spite detection. These technical characteristics combined with market-specific factors and the lack of regulatory oversight in NFT markets exemplify these digital trading environments, where unique digital assets are traded with highly subjective valuations and minimal barriers to entry [VW19].

In response to these challenges, Wen *et al.* developed NFTDisk, a specialized visual analytics tool designed for expert-driven qualitative analysis of NFT transaction networks [WWY+23]. Their implementation enables domain experts to interactively explore transaction patterns and revealed significant wash trading schemes through manual investigation, including alternating cycle groups that artificially inflate market volume and asset valuations. However, while NFTDisk's qualitative approach contributes valuable domain knowledge for characterizing wash trading behaviors, it provides a visualization interface for human interpretation rather than automated algorithmic detection. It requires manual examination to identify suspicious patterns and cannot systematically enumerate all temporal cycles or operate in real-time on continuous transaction streams.

3 Theoretical Foundation

This chapter establishes the theoretical foundation for analyzing market manipulation in blockchain-based digital asset markets. We begin by defining the mathematical notation of transaction flows in temporal graphs. Further, we briefly trace blockchain's evolution from Bitcoin's introduction as a decentralized ledger to Ethereum's emergence as a programmable platform, which enabled creators to publish NFT collections through smart contracts. Building on this foundation, we formalize wash trading detection as a cycle identification problem and incorporate two key concepts from existing literature: the burstiness index from Goh *et al.* [GB08] for quantifying temporal transaction clustering, and the economic profit function from La Morgia *et al.* [MMMN23] that captures the incentive structures driving manipulative trading behavior. These elements provide the theoretical scaffolding for the empirical analyses that follow.

3.1 Preliminary Definitions and Notation

Definition 3.1 (Transaction Network)**.** A *transaction network*, *transaction graph* or *dynamic network* $G = (V, \mathcal{E}, \varphi)$ is a weighted, directed graph consisting of a finite vertex set V, a set $\mathcal{E}$ of edges, and a weight function $\varphi : \mathcal{E} \to \mathbb{R}^+$. Each edge $e = (u, v, t) \in \mathcal{E}$ is a triple consisting of two distinct vertices $u, v \in V$, a timestamp $t \in \mathbb{R}^+$.

In this structure, the set of vertices V represents the set of unique actors (e.g., wallet addresses) who participate in the marketplace. The set of edges $\mathcal{E}$ represents the transactions between these actors. The edge $e = (u, v, t)$ is directed from u to v with $t \in \mathbb{R}^+$ being the timestamp indicating when the transaction was executed, and $\varphi(e) \in \mathbb{R}^+$ represents the value or amount associated with the transaction (e.g., transaction amount, volume, or monetary value).

Definition 3.2 (Trading Volume)**.** For a subgraph $G' = (V', \mathcal{E}', \varphi')$ where $V' \subseteq V$ and $\mathcal{E}' \subseteq \mathcal{E}$, the *trading volume* is defined as:

$$\mathrm{vol}(G') := \sum_{e \in \mathcal{E}'} \varphi(e).$$

Trading volume (or simply *volume*) represents the total monetary value of all transactions within a given subgraph. In financial markets, volume serves as a key indicator of market activity and liquidity, with higher volumes typically signaling greater participant interest and market health.

Definition 3.3 (Temporal Walks)**.** Let $k \in \mathbb{N}$ and $v_0, v_1, \ldots, v_k \in V$ be a sequence of (not necessarily distinct) vertices. A *temporal walk* w from v_0 to v_k is a sequence of edges

$$w = \langle (v_0, v_1, t_1), (v_1, v_2, t_2), \ldots, (v_{k-1}, v_k, t_k) \rangle,$$

such that $t_1 < t_2 < \ldots < t_k$. Throughout this thesis, we use the shorthand notation

$$w : v_0 \xrightarrow{t_1} v_1 \xrightarrow{t_2} v_2 \cdots \xrightarrow{t_k} v_k.$$

We now extend the concept of a temporal walk by allowing multiple timestamps between consecutive vertices.

Definition 3.4 (Bundled Walks)**.** Let, $v_0, v_1, \ldots, v_k \in V$ be a sequence of (not necessarily distinct) vertices. A *bundled walk* W from v_0 to v_k is a sequence of triplets

$$W = \langle (v_0, v_1, T_1), (v_1, v_2, T_2), \ldots, (v_{k-1}, v_k, T_k) \rangle,$$

where for each $i \in \{1, \ldots, k\}$ $T_i = \{t_i^1, t_i^2, \ldots, t_i^{m_i}\}$ is a non-empty and finite set of timestamps such that $(v_{i-1}, v_i, t_i^m) \in \mathcal{E}$ for all $1 \leq m \leq m_i$. Furthermore, for each $i \in \{1, \ldots, k-1\}$, we require the following *temporal ordering constraint* to hold:

$$\min T_i < \min T_{i+1} \wedge \max T_i < \max T_{i+1}.$$

We use the shorthand notation

$$W : v_0 \xrightarrow{T_1} v_1 \xrightarrow{T_2} v_2 \cdots \xrightarrow{T_k} v_k.$$

A temporal walk can be viewed as a special case of a bundled walk where each timestamp set contains exactly one element, i.e., $|T_i| = 1 \mid \forall i \in \{1, 2, \ldots, k\}$.
For a bundled walk W, the set of edges involved in the walk is denoted by

$$\mathcal{E}_W = \bigcup_{i=1}^{k} \{(v_{i-1}, v_i, t) : t \in T_i\}.$$

Similarly, the set of vertices involved in the walk is denoted by

$$V_W = \bigcup_{i=0}^{k} \{v_i\}.$$

Let $\mathcal{A}$ denote the universe of all assets. For each edge $e = (u, v, t) \in \mathcal{E}$, we define the asset mapping function $\alpha : \mathcal{E} \to \mathcal{A}$ that returns the asset transferred in transaction e from vertex u to vertex v at time t.

Given a bundled walk W, the set of assets traded along the walk is:

$$\mathcal{A}_W = \{\alpha(e) \mid e \in \mathcal{E}_W\} \subseteq \mathcal{A}$$

This represents all unique assets from the universe $\mathcal{A}$ that are transferred in the transactions comprising walk W.

Definition 3.5 (Walk Duration)**.** The *duration* of a temporal walk w is defined as

$$\mathrm{dur}(w) := t_k - t_1.$$

The *duration* of a bundled walk W is defined as

$$\mathrm{dur}(W) := \max(T_k) - \min(T_1).$$

Definition 3.6 (Temporal Path)**.** A bundled — or simply temporal — walk

$$W : v_0 \xrightarrow{T_1} v_1 \xrightarrow{T_2} v_2 \cdots \xrightarrow{T_k} v_k$$

is called a *temporal path* if no vertex is visited more than once, i.e. $v_i \neq v_j$ for $i \neq j$.

Definition 3.7 (Temporal Cycle)**.** A bundled — or simply temporal — walk

$$p : v_0 \xrightarrow{T_1} v_1 \xrightarrow{T_2} v_2 \cdots \xrightarrow{T_k} v_k$$

is called a *temporal cycle*, if the first and last vertices are identical, i.e. $v_0 = v_k$. In this case we say that W is *rooted at the vertex* v_0. The cycle W is called *simple* if no vertex except v_0 appears more than once, i.e. $v_i \neq v_j$ for all $i \neq j \in \{0, \ldots, k-1\}$. We may at times simply use the term *cycles* to refer to temporal cycles.

Definition 3.8 (Cycle Seed)**.** A *cycle seed* (or simply *seed*) is a triple

$$(t_{\text{start}}, t_{\text{end}}, C)$$

where $t_{\text{start}}, t_{\text{end}} \in \mathbb{R}^+$ with $t_{\text{start}} < t_{\text{end}}$ define a temporal interval $[t_{\text{start}}, t_{\text{end}}]$, and $C \subseteq V$ is a non-empty set of candidate vertices.

A seed represents a potential cycle rooted at some vertex $v \in C$ within the temporal interval $[t_{\text{start}}, t_{\text{end}}]$. The candidate set C contains all vertices that may participate in temporal cycles rooted at v during this interval.

3.2 Automatic Wash Trading and Artificial Volume Inflation

Wash trading represents a form of market manipulation where trades are executed in circles among controlled accounts to create artificial market activity. These circular transactions create trading volume without genuine changes in ownership or market exposure. The emergence of blockchain-based trading infrastructure has fundamentally transformed the execution of these manipulation schemes. Ethereum's smart contracts — self-executing programs that run on the blockchain — function as automated trading systems that can execute predefined operations when certain conditions are met, without requiring human intermediaries [Woo14]. This programmability enables manipulators to orchestrate circular trading patterns continuously and automatically, generating substantial artificial volume with minimal capital exposure.

From a graph-theoretic perspective, wash trading manifests as temporal cycles in dynamic networks. The power of this manipulation strategy lies in its repeatability — once established, these circular trading routes can be executed continuously through smart contract automation. Each transaction requires 'gas' fees (computational costs paid to network validators for executing the transaction) and optional priority tips (additional payments to accelerate transaction processing) [KM23]. However, beyond these per-transaction costs, the same capital can generate virtually unlimited artificial volume through repeated circulation, making it an attractive strategy for manipulating volume-based metrics.

The prevalence of such automated wash trading is fundamentally driven by misaligned economic incentives in platform design. Following La Morgia *et al.* [MMMN23], when market platforms distribute rewards R proportional to trading volume V, rational actors seek to maximize the profit function $\Pi = R(V) - C(V)$, where $C(V)$ represents transaction costs. This incentive structure becomes particularly problematic in the NFT ecosystem, where trading has shifted from decentralized peer-to-peer exchanges to centralized platforms such as 'OpenSea' and 'LooksRare'. These platforms provide user-friendly interfaces and volume-based incentive programs that inadvertently encourage wash trading behaviors.

The evolution toward *royalty-free trading* has further amplified this problem [LC25]. NFTs were initially designed with automatic royalty payments to creators on secondary sales — a feature enforced through smart contracts that ensured artists received residual income from each transaction. These royalties effectively raised $C(V)$ for wash traders, creating a natural deterrent to circular trading. However, competitive pressures among marketplaces have led many platforms to make these automatic royalty payments optional or eliminate them entirely. This reduction in transaction costs widens the profitability window for wash trading, enabling traders to generate artificial volume at minimal cost while capturing disproportionate platform rewards. The ease of creating new blockchain addresses further enables malicious actors to rapidly establish complex trading networks, while transaction pseudo-anonymity complicates the attribution of coordinated behavior to single entities.

Given the automated nature of these manipulation schemes, temporal analysis provides a powerful detection mechanism. While legitimate trading activity exhibits irregular patterns influenced by heterogeneous user demand and stochastic market conditions, automated wash trading systems display distinctive behavioral signatures. Specifically, they exhibit *sender-centric regularity* — consistent, periodic patterns of outgoing transactions from a small number of addresses at predictable intervals. This regularity can be quantified through the burstiness index, which measures deviations from natural patterns, with values approaching -1 indicating the highly regular, mechanical behavior typical of automated systems [GB08].

Formally, for time deltas between consecutive transactions δ_i^u from sender u natural trading follows a Poisson process with broadly distributed gaps and very low probability near $\delta \approx 0$. Automated wash trading, however, produces an excess of short δ_i^u values. The burstiness index captures this pattern:

$$B(D_u) = \frac{\sigma(D_u) - \mu(D_u)}{\sigma(D_u) + \mu(D_u)}$$

where $D_u = \{\delta_i^u\}$ is the set of all inter-event times for sender u, $\mu(D_u)$ and $\sigma(D_u)$ being their mean and standard deviation. Values near $B \approx -1$ indicate highly regular, likely automated activity, while $B \approx 0$ corresponds to Poisson-like randomness typical of natural trading, and $B \approx 1$ reflects strong burstiness. This metric provides a robust measure for detecting the regular, automated trading behavior characteristic of wash trading systems, complementing cycle-based detection methods in identifying market manipulation patterns.

3.3 Chained Components

Having established the theoretical basis for wash trading detection through temporal cycles, we now introduce higher-order structures that capture coordinated manipulation campaigns across multiple cycles.

Individual simple cycles in transaction networks reveal localized wash trading behaviors, but real-world market manipulation rarely operates in isolation. Advanced actors coordinate their activities across multiple trading cycles, deliberately fragmenting their operations across time and addresses to evade detection. These coordinated campaigns require a higher-order analytical framework that can reconnect intentionally fragmented manipulation patterns.

We observe two critical characteristics of real-world wash trading campaigns that simple cycle detection fails to capture:

1. *temporal persistence*: sustained manipulation operations persist across an overarching temporal window, distinguishing them from brief cyclic trades.

2. *network coordination*: manipulators distribute activity across multiple recurring addresses, creating complex webs of interconnected transactions.

To address these challenges, we introduce *chained components* — a graph-theoretic construct that aggregates structurally and temporally related bundled cycles into unified manipulation campaigns. Unlike isolated bundled cycles that capture only immediate wash trading networks, chained components reveal the broader orchestration of manipulation campaigns by reconnecting manipulation patterns that appear innocuous in isolation but reveal coordinated campaigns when properly aggregated.

Definition 3.9 (ω-Proximity)**.** Two bundled walks W_1 and W_2 are *ω-proximate* if there exist edges $e_1 = (u_1, v_1, t_1) \in \mathcal{E}_{W_1}$ and $e_2 = (u_2, v_2, t_2) \in \mathcal{E}_{W_2}$ such that $|t_1 - t_2| \leq \omega$.

Definition 3.10 (Chained Component)**.** A *chained component* $\mathcal{C}$ is a subgraph induced by a set of maximal size of bundled simple cycles $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ such that for each consecutive pair of cycles (C_i, C_{i+1}), the cycles are ω-proximate and $V_{C_i} \cap V_{C_{i+1}} \neq \emptyset$, i.e., they share at least one common actor.

The dual requirements of temporal proximity and structural overlap ensure that chained components capture genuinely coordinated activities rather than spurious correlations. The maximality constraint guarantees that we identify complete campaigns, preventing artificial fragmentation of naturally connected operations.

To characterize these higher-order structures, we employ three key metrics:

Definition 3.11 (Consolidation Ratio)**.** The consolidation ratio quantifies the degree of cycle aggregation:

$$\text{consolidation ratio} = \frac{|\text{bundled cycles}|}{|\text{chained components}|}$$

Definition 3.12 (Component Duration)**.** For a chained component $\mathcal{C}$, the duration quantifies the total temporal span of coordinated trading activity:

$$\text{dur}(\mathcal{C}) = \max_{(u,v,t)=e\in\mathcal{E}_\mathcal{C}} t - \min_{(u,v,t)=e\in\mathcal{E}_\mathcal{C}} t$$

where $\mathcal{E}_\mathcal{C} = \bigcup_{i=1}^{n} \mathcal{E}_{C_i}$ is the set of all edges in the component.

Definition 3.13 (Temporal Expansion Ratio)**.** The temporal expansion ratio reveals how far coordinated trading extends beyond the base temporal window:

$$\text{expansion ratio}(\mathcal{C}) = \frac{\text{dur}(\mathcal{C})}{\omega}$$

To identify the most transparent form of market manipulation, we define trivial components that capture the simplest structural patterns:

Definition 3.14 (Trivial Component)**.** A chained component $\mathcal{C}$ is *trivial* if it contains exactly two participating addresses ($|V_\mathcal{C}| = 2$), involves a single token being traded ($|\mathcal{A}_\mathcal{C}| = 1$), and all edges form a bipartite trading pattern between the two actors.

Trivial components represent the simplest wash trading structure where two addresses repeatedly exchange a single asset. When detected, trivial components signal unambiguous wash trading activity. The prevalence and temporal distribution of trivial components serve as indicators of imminent manipulation.

4 Algorithm Design and Implementation

The presented algorithm DSCENT (Dynamic Streaming Cycle Enumeration with Temporal constraints) extends the foundational principles of 2SCENT [KC18] to operate in streaming and dynamic graph environments. Both algorithms are designed to detect simple bundled cycles with duration less than or equal to ω, where the ω-threshold serves as a configurable input parameter that sets the detection window for cycle enumeration. While 2SCENT processes static temporal graphs, DSCENT introduces several key innovations to handle continuous streams of transactions efficiently.

The core concept from 2SCENT that DSCENT inherits is the generation of *reverse-reachability summaries*, which compactly represent temporal paths in the graph. However, DSCENT fundamentally reimagines how these summaries are maintained and utilized through five key innovations: simultaneous transaction handling, streaming architecture for real-time processing, dynamic seed finalization with continuous output, parallel execution for scalability, and adaptive memory management for long-running applications. These innovations enable DSCENT to operate continuously on unbounded streams while maintaining the correctness guarantees of the original 2SCENT algorithm. The complete implementation is publicly available at `https://github.com/markzmbl/graph-analysis`. We next describe DSCENT's streaming execution framework, which coordinates the continuous processing of incoming blocks while ensuring efficient parallel execution and memory management.

4.1 Core Data Structures

These structures capture different aspects of the temporal graph: reverse-reachability summaries track which nodes can reach a given target from a given timestamp onwards, block interactions organize incoming edges by timestamp, and interval trees manage potential cycle seeds. Importantly, the reverse-reachability summaries and interval trees are created and removed dynamically as nodes become relevant to active temporal windows.

Definition 4.1 (Reverse-Reachability Summaries)**.** The *reverse-reachability summary* $S_t(v)$ for a node v captures all nodes that can reach v starting from time t:

$$S_t(v) = \left\{ (t_1, U) \mid \forall u \in U \exists \text{ temporal path } w : u \xrightarrow{t_1} \cdots \xrightarrow{*} v \text{ with } t_1 > t \right\}.$$

Each entry (t_1, U) represents a timestamp t_1 and the set of nodes U that can reach v starting from time t_1. We denote by

$$T_{S_t(v)} := \{ t : (t, U) \in S_t(v) \text{ for some vertex set } U \}$$

the set of timestamps appearing in $S_t(v)$. When merging two summaries, as performed in line 8 of the `ProcessBlockInteractions` algorithm, timestamp keys remain distinct, and for any shared timestamp t_1, the corresponding vertex sets are combined. More formally, let $v, v' \in V$ be distinct vertices with their respective reverse-reachability summaries $S_t(v)$ and $S_t(v')$. Denote by

$$S_t^*(v, v') := \left\{(t_1, U^*) : U^* = U \cup U', (t_1, U) \in S_t(v), (t_1, U') \in S_t(v')\right\}.$$

We then define the merge of $S_t(v)$ and $S_t(v')$ as

$$S_t(v) \sqcup S_t(v') := S_t^*(v, v') \cup \{(t_1, U) \in S_t(v) \cup S_t(v') : t_1 \in T_{S_t(v)} \triangle T_{S_t(v')}\},$$

where $\triangle$ denotes the symmetric set difference.

These reverse-reachability summaries are instantiated on-demand when a node becomes involved in potential reachability relationships and are deleted once the pruning eliminates stale entries such that no entries remain in the summary.

To organize the incoming edge stream processed in algorithm 1, DSCENT uses block interactions:

Definition 4.2 (Block Interactions)**.** Block interactions represent batches of edges arriving at the same timestamp:

$$B_t = \left\{(U, v) \mid v \in V, \forall u \in U : (u, v, t) \in \mathcal{E}\right\}$$

This structure compactly encodes all incoming edges at time t, where each element (v, U) represents a target node v and the set of source nodes U that have edges into v at that timestamp.

The algorithm maintains a dynamic transaction graph $\mathcal{G}$ (initialized at line 2 and updated at line 8 of algorithm 1) that evolves as new edges arrive and old edges are pruned. This graph serves as the substrate upon which constrained depth-first searches are performed during seed processing.

For candidate seed management, each relevant node v maintains a B-tree $\mathcal{T}_v$ storing temporal intervals of potential cycles. Each entry has the form $(t_{\text{start}}, t_{\text{end}}, C)$, where the closed interval $[t_{\text{start}}, t_{\text{end}}]$ defines the temporal scope and C denotes the associated candidate nodes. Like the reverse-reachability summaries, these interval trees are created dynamically when needed and removed when they no longer contain elements, enabling efficient memory management in the streaming setting.

4.2 Main Algorithm: Streaming Execution Framework

Algorithm 1 presents `YieldBundledCycles`, DSCENT's main procedure that coordinates all subroutines in a streaming fashion. The algorithm maintains four key structures: reverse-reachability summaries (S), interval B-trees ($\mathcal{T}$), a dynamic transaction graph ($\mathcal{G}$), and a task queue (Q) for parallel processing. As block transactions arrive from the stream $\mathcal{B}$, the algorithm orchestrates three main activities: processing incoming blocks to update

Algorithm 1: YieldBundledCycles: Main procedure for detecting and yielding bundled cycles from a block transaction stream $\mathcal{B}$ within the time threshold ω.

Data: Block transaction stream $\mathcal{B}$, threshold ω
Result: Iterator yielding bundled cycles
`// Declare reverse-reachabilities, B-Trees, Transaction Graph and Task Queue`
$S \leftarrow \emptyset$; $\mathcal{T} \leftarrow \emptyset$; $V \leftarrow \emptyset$; $\mathcal{E} \leftarrow \emptyset$; $\gamma \leftarrow \emptyset$ `// Initialize weight function`
$\mathcal{G} \leftarrow (V, \mathcal{E}, \varphi)$; $Q \leftarrow \emptyset$
for $(B_t, \gamma) \in \mathcal{B}$ **do**
  `// Piecewise union of weights`
  $\varphi \leftarrow \varphi(e)$ if $e \in \mathcal{E}$ else $\gamma(e)$
  for $(U, v) \in B_t$ **do**
    $V \leftarrow V \cup \{v\} \cup U$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(u, v, t) \mid u \in U\}$
  $\mathcal{G} \leftarrow (V, \mathcal{E}, \varphi)$ `// Update graph`
  ProcessBlockInteractions(S, t, B_t)
  $t^- \leftarrow t - \omega$; $\hat{\mathcal{S}} \leftarrow \emptyset$
  for all $\mathcal{T}_v$ **do**
    $\mathcal{T}_\Delta \leftarrow \{(t_{\text{start}}, t_{\text{end}}, C) \in \mathcal{T}_v \mid t_{end} < t^-\}$ `// Find ready seeds`
    $\hat{\mathcal{S}} \leftarrow \hat{\mathcal{S}} \cup \mathcal{T}_\Delta$
    $\mathcal{T}_v \leftarrow \mathcal{T}_v \setminus \mathcal{T}_\Delta$
  for $(t_{start}, t_{end}, C) \in \hat{\mathcal{S}}$ **do**
    $\phi \leftarrow$ **async** *ProcessSeed*$(\mathcal{G}, t_{start}, t_{end}, C)$ `// Launch parallel task`
    Q.enqueue$(((t_{\text{start}}, t_{\text{end}}, C), \phi))$
  for $((t_{start}, t_{end}, C), \phi) \in Q$ *where* ϕ*.isFinished()* **do**
    yield ϕ*.result()* `// Yield completed cycles`
    Q.remove$(((t_{\text{start}}, t_{\text{end}}, C), \phi))$
  if *time to prune* **then**
    for all $S_t(v)$ **do**
      $S_t(v) \leftarrow \{(t_x, X) \in S_t(v) \mid t_x > t^-\}$
    `// Compute minimum threshold for graph pruning`
    $t_Q \leftarrow \min\big(\{t_{\text{start}} \mid ((t_{\text{start}}, *), *) \in Q\}\big)$
    $t^* \leftarrow \min\big(t^-, t_Q\big)$
    $\mathcal{E} \leftarrow \{(u, v, t) \in \mathcal{E} \mid t \geq t^*\}$
    $V \leftarrow \{v \in V : \exists (u, u', t) \in \mathcal{E} : v \in \{u, u'\}\}$
    $\mathcal{G} \leftarrow (V, \mathcal{E}, \varphi)$

reachability information (line 9), identifying and launching parallel seed processing tasks (lines 10–17), and pruning outdated data to maintain memory efficiency (lines 23–28).

The main loop processes each block transaction B_t and its associated weight function γ sequentially. Here, γ provides the transaction weights for the edges introduced in block B_t. For each block, the algorithm performs a piecewise union to extend the graph's weight function φ with the new weights from γ. It then incorporates the new vertices and edges from the block to update the dynamic transaction graph structure (line 8), and invokes `ProcessBlockInteractions` (detailed in Section 4.3) to update reverse-reachability summaries and generate new seeds.

DSCENT dynamically determines when seeds are ready for processing based on the temporal window. A seed $(t_{\text{start}}, t_{\text{end}}, C)$ is considered finalized when $t_{\text{end}} < t - \omega$ (line 12), meaning no future edges within the temporal window can affect the cycle enumeration for this seed. This dynamic finalization is a key departure from 2SCENT's batch processing model, allowing DSCENT to begin yielding results before the entire stream is processed, providing early insights and enabling real-time analysis of temporal cycles.

Finalized seeds are processed independently through parallel execution (lines 16–17). Each seed is offloaded as a future ϕ to a process pool with a read-only copy of the relevant portion of the dynamic transaction graph. The `ProcessSeed` function implements constrained depth-first search from 2SCENT, exploring all ω-valid temporal cycles within the seed's interval $[t_{\text{start}}, t_{\text{end}}]$ using candidate set C. This method operates on read-only graph snapshots, ensuring thread safety without synchronization overhead. The parallel architecture allows the main thread to continue processing incoming blocks while seeds are explored asynchronously. Completed cycles are yielded as soon as their corresponding tasks finish (lines 19–20), providing a continuous stream of results rather than waiting for batch completion.

Long-running streams require careful memory management to prevent unbounded growth. DSCENT implements a sophisticated pruning strategy (lines 23–28) that operates at two levels. First, reverse-reachability pruning periodically removes entries older than $t - \omega$ from all summaries, maintaining only the active temporal window. Second, graph pruning computes the minimum timestamp t^* based on both the temporal window and active seeds. The pruning threshold t^* (line 25) is defined as the minimum of two complementary bounds: the base cutoff $t^- = t - \omega$ defines the maximum admissible temporal window for future seed generation, while the queue Q contributes $\min(\{t_{\text{start}} \mid ((t_{\text{start}}, t_{\text{end}}, C), \phi) \in Q\})$ to ensure edges required by parallel computations are retained. Taking the minimum yields:

$$t^* \;=\; \min\Big(t^-,\; \min\{\, t_{\text{start}} \mid ((t_{\text{start}}, t_{\text{end}}, C), \phi) \in Q \,\}\Big)$$

This pruning strategy preserves correctness while limiting memory consumption, with pruning frequency tunable based on available memory and stream characteristics.

Having established the overall streaming framework, we now detail the core data structures and subroutine algorithms that enable DSCENT's efficient operation.

4.3 Block Transaction Processing

The `ProcessBlockInteractions` procedure, invoked at line 9 of algorithm 1, follows a carefully orchestrated three-phase pre-processing approach before seed generation. This ensures consistency while maximizing opportunities for simultaneously appearing edges.

Algorithm 2: ProcessBlockInteractions: Preprocesses reverse-reachabilities from block interactions at time t and updates Seed Interval Trees.

Data: Tentative Reverse-reachabilities $S_{t^\perp}$, possibly containing expired entries, current time t, threshold ω, block interactions B_t

1 $V' \leftarrow \bigcup_{(v,U)\in B_t}(\{v\} \cup U)$
2 $t^- \leftarrow t - \omega$
3 **for** $v \in V'$ **do**
4 $\quad S_{t^-}(v) \leftarrow \{(t_x, X) \in S_{t^\perp}(v) \mid t_x \geq t^- \}$ `// Prune outdated entries`
5 **for** $(U, v) \in B_t$ **do**
6 $\quad S_{t^-}(v) \leftarrow S_{t^-}(v) \cup \{(t, U)\}$ `// Direct updates from block`
7 **for** $(U, v) \in B_t$ **do**
8 $\quad S_{t^-}(v) \leftarrow S_{t^-}(v) \sqcup \bigsqcup_{u\in U} S_{t^-}(u)$ `// Propagate reachabilities`
9 **for** $(U, v) \in B_t$ **do**
10 $\quad \mathcal{S}_v \leftarrow$ ProcessTargetInteractions(v, U, t) `// Generate seeds`
11 $\quad \mathcal{T}_v \leftarrow \mathcal{T}_v \cup \mathcal{S}_v$
12 $\quad \mathcal{T}_v \leftarrow$ MergeEnclosedIntervals$(\mathcal{T}_v, \omega)$

The first phase removes outdated entries from reverse-reachability summaries. Specifically, for all nodes involved in the current block (both sources and targets), we retain only those reverse-reachability entries within the temporal threshold ω (line 4). The threshold $t^- = t - \omega$ (line 2) ensures we maintain exactly the temporal window required for correctness, corresponding to the same threshold used in the main algorithm for seed finalization.

Unlike 2SCENT, which processes edges individually, the update phase of algorithm 2 processes all edges arriving at time t in two steps to ensure correctness and consistency. First, direct updates (line 6) add the entry (t, U) to $S_{t^-}(v)$ for each target interaction (v, U) in the block, where U is the set of all sources targeting v at time t. This records the immediate reverse-reachability information. Second, reachability propagation (line 8) occurs after completing all direct updates. For each edge (v, U), we merge into v's summary all reverse-reachabilities from nodes in U.

The separation into distinct phases is critical: it guarantees that when computing transitive closures, all direct connections at time t are already recorded. This batched processing exploits the block structure of transactions more efficiently than edge-by-edge approaches like 2SCENT.

The final phase generates seeds through `ProcessTargetInteractions` (detailed in the next section) and adds them to the interval trees, which are subsequently accessed by the main algorithm to identify finalized seeds.

4.4 Seed Generation

The `ProcessTargetInteractions` procedure, called at line 10 of algorithm 2, identifies potential cycles by detecting self-reachable nodes and creating temporal seed intervals. This phase exploits the structure created during the update phase to efficiently identify all potential cycles. The algorithm operates on the principle that a cycle exists when node v appears in its own reverse-reachability summary at an earlier time $t' < t$, and there exists a path from v back to itself through the current interaction.

Algorithm 3: ProcessTargetInteractions: Discovery of seed sets $\mathcal{S}_v$ for a target node v at time t

Data: Target node v, source nodes U, current time t,
pruned reverse-reachability summaries S_{t^-}
Result: Discovered seeds $\mathcal{S}_v$ for target node v

```
// Filter for cycles where v reaches itself at earlier times
```

1 $T' \leftarrow \{t' \mid (t', U') \in S_{t^-}(v) : t' < t, v \in U'\}$
2 **if** $T' = \emptyset$ **then**
3 | **return** $\emptyset$
4 $\mathcal{S}_v \leftarrow \emptyset$
5 **for** $u \in U$ **do**
6 | **for** $t' \in T'$ **do**

```
| | // Collect nodes reachable to u (and therefore v) after time t'
```

7 | | $\mathcal{C}^u \leftarrow \{U_1 \mid (t_1, U_1) \in S_{t^-}(u) : t_1 \geq t'\}$
8 | | $C \leftarrow \{v\} \cup \bigcup_{U_1 \in \mathcal{C}^u} U_1$
9 | | $\mathcal{S}_v \leftarrow \mathcal{S}_v \cup \{(t', t, C)\}$

```
// Remove v to avoid duplicate seed generation
```

10 $S_{t^-}(v) \leftarrow \{(t', U' \setminus \{v\}) \mid (t', U') \in S_{t^-}(v) : t' \in T'\}$
11 **return** $\mathcal{S}_v$

Since we propagated reverse reachabilities in the update phase where $S_{t^-}(v)$ was updated with $\bigsqcup_{u \in U} S_{t^-}(u)$, we can efficiently identify cycles by examining these propagated summaries.

The algorithm performs bilateral temporal filtering around each candidate time t': First, we identify times $t' < t$ where v was reachable from itself (line 1), indicating the start of a potential cycle. Then, for each source node $u \in U$ targeting v at time t, we examine $S_t(u)$ to find nodes reachable from u specifically at or after time t' (line 7). This bilateral filtering ensures temporal consistency: the cycle must start after t' when v could reach itself and complete before the current time t.

These temporally-filtered nodes, combined with v itself, form candidate sets for cycles existing in the interval $[t', t]$ (line 8). The generated seeds are then added to the interval trees maintained by algorithm 2, where they await finalization in the main algorithm.

The final step (line 10) removes v from its own reverse-reachability summary to prevent duplicate seed generation in future iterations.

4.5 Seed Management and Merging

Efficient seed management is crucial for DSCENT's performance. The `MergeEnclosedIntervals` procedure, invoked at line 12 of algorithm 2, employs interval B-trees to organize seeds and implements merging strategies to reduce redundant computation. This optimization is particularly important as it reduces the number of parallel tasks launched by the main algorithm.

Algorithm 4: MergeEnclosedIntervals: Merges overlapping and enclosed intervals within a temporal window of size ω.

Data: Interval tree $\mathcal{T}$, threshold ω
Result: Interval tree with merged intervals

1 **if** $|\mathcal{T}| \leq 1$ **then**
2 **return** $\mathcal{T}$ `// Nothing to merge`
3 $I \leftarrow$ sort $\mathcal{T}$ by (t_{start} ascending, t_{end} descending)
4 $\mathcal{M} \leftarrow \emptyset$ `// Empty B-tree for results`
5 $(t_{\text{start}}, t_{\text{end}}, C) \leftarrow \text{pop}(I)$ `// Start with first interval`
6 **while** $I \neq \emptyset$ **do**
7 $t^+ \leftarrow t_{\text{start}} + \omega$
  `// Merge intervals within the temporal window`
8 **while** $I \neq \emptyset$ **do**
9 $(t'_{\text{start}}, t'_{\text{end}}, C') \leftarrow \text{peek}(I)$
10 **if** $t'_{end} \geq t^+$ **then**
  `// Start new group if outside window`
11 $\mathcal{M} \leftarrow \mathcal{M} \cup \{(t_{\text{start}}, t_{\text{end}}, C)\}$
12 $(t_{\text{start}}, t_{\text{end}}, C) \leftarrow \text{pop}(I)$
13 **break**
14 $(t'_{\text{start}}, t'_{\text{end}}, C') \leftarrow \text{pop}(I)$
  `// Extend the merged interval`
15 $t_{\text{end}} \leftarrow \max(t_{\text{end}}, t'_{\text{end}})$
16 $C \leftarrow C \cup C'$
 `// Add the last remaining interval`
17 $\mathcal{M} \leftarrow \mathcal{M} \cup \{(t_{\text{start}}, t_{\text{end}}, C)\}$
18 **return** $\mathcal{M}$

The merging algorithm consolidates overlapping and nearby seed intervals to reduce redundant graph exploration. Seeds are merged when their end times fall within the temporal window ω (line 7), combining their candidate sets (line 16) while extending the interval to cover all merged seeds (line 15).

This merging strategy reduces the number of seeds that will be processed in parallel by algorithm 1 while ensuring no cycles are missed. For the current group, the temporal constraint $t^+ = t_{\text{start}} + \omega$ ensures that only seeds that could potentially explore overlapping temporal regions are merged.

Sorting the intervals by increasing start time and, for ties, by decreasing end time (line 3) ensures that longer intervals enclosing shorter ones are processed first. This prevents enclosed intervals from being prematurely finalized and guarantees correct merging.

Together, these components — `ProcessBlockInteractions`, `ProcessTargetInteractions`, and `MergeEnclosedIntervals` — work in concert with the main streaming framework to enable DSCENT to process unbounded temporal streams while maintaining the correctness guarantees of the original 2SCENT algorithm, providing continuous cycle detection with minimal latency and bounded memory consumption.

5 Experimental Design and Results

This chapter presents evidence of systematic market manipulation in the Meebits NFT collection through a multi-layered analytical approach. Prior work by La Morgia *et al.* [MMMN23] documented the substantial financial gains from such manipulation schemes, revealing that coordinated wash trading networks extracted millions in token rewards while artificially inflating market metrics by orders of magnitude. The investigation reveals coordinated manipulation activities that operated across three distinct levels: market-wide volume distortions, automated actor behaviors, and structured transaction patterns. The convergence of evidence from these different analytical perspectives provides compelling documentation of a large-scale exploitation campaign that artificially inflated trading activity during a critical six-week period in early 2022. Beyond the manipulation analysis, this chapter evaluates DSCENT's computational performance through comparative benchmarking against 2SCENT and comprehensive stress testing on an extensive blockchain dataset. Our analysis leverages two complementary blockchain datasets that enable detection of manipulation patterns across different granularities of transaction data.

1. **Meebits NFT Dataset**: The Meebits trading dataset was collected through systematic web scraping of Etherscan.io [Eth15], covering the period from the project's launch in May 2021 through June 2025. Unlike the raw Layer 1 transaction data, NFT trading data is a processed interpretation layer built on top of the fundamental blockchain transactions. Each NFT trade involves multiple underlying smart contract interactions that are decoded and interpreted to extract meaningful trading relationships. The data collection process involved two parallel extraction procedures: first, NFT transfer logs were extracted to identify source and target actors for each Meebit token movement, and second, the NFT trades section was scraped to capture transaction prices, marketplace identifiers, and specific token IDs involved in each sale. These datasets were subsequently merged using transaction hashes as the primary key, creating a comprehensive trading graph containing 47,577 transactions between 17,516 unique actors involving 10,587 distinct Meebit tokens. This processed dataset captures both the flow of assets and their associated monetary values across different marketplaces, providing a complete view of the Meebits trading ecosystem at a semantic level above the raw blockchain layer.

2. **Ethereum Transaction Dataset**: The Ethereum transaction dataset was acquired by deploying an Ethereum node using the Go-Ethereum (geth) client [Eth13]. The node was synchronized with the Ethereum mainnet to collect transaction data from mid-2019 through late-2024. This approach ensured data integrity and access to all

on-chain transactions without relying on third-party APIs or rate-limited services. The synchronization process captured all transaction metadata including sender addresses, receiver addresses, transaction values, and block numbers, resulting in a dataset containing approximately 1.8 billion edges representing individual transactions across the Ethereum network. As Layer 1 blockchain data, these transactions represent the fundamental, unprocessed transfer events on the network.

5.1 Market-Level Anomalies: Platform Volume Distortions

We now present evidence of systematic manipulation across three analytical levels, beginning with market-wide trading volume distortions. The most compelling evidence of systematic market manipulation emerges from extraordinary volume disparities between NFT marketplaces during early 2022. These trading patterns, illustrated in Figure 5.1, reveal fundamental deviations from normal market behavior that exceed 1 billion USD in transaction volume and cannot be explained by organic demand.

The anomalies center on 'LooksRare', an NFT marketplace that launched in January 2022 with a token-based reward system. LooksRare distributed 'LOOKS' tokens — LooksRare's own native cryptocurrency — to users proportional to their trading volume. This means traders could earn valuable tokens simply by executing high-value transactions, regardless of whether those trades represented genuine market activity. When compared against 'OpenSea' — the established market leader — LooksRare exhibited volume spikes that exceeded typical NFT market activity by orders of magnitude. While OpenSea maintained steady baseline activity throughout the period, LooksRare demonstrated volume concentrations that suggest coordinated exploitation rather than genuine trading demand.

This manipulation manifests most clearly during what the analysis identifies as a critical 6-week period from January 10 to February 21, 2022 — the "LooksRare Exploitation Window". During this interval, LooksRare achieved peak weekly trading volumes that subsequently declined as sharply as it had emerged. Figure 5.2 captures the extreme nature of this anomaly on a linear scale and reveals that LooksRare's trading volumes peaked at just over 2.3 billion USD weekly. The temporal precision of this pattern provides the first signal that these volumes were artificially generated. More revealing is the volume-transaction divergence that exposes the artificial nature of these trades. During the peak period, LooksRare dominated Meebits market transactions, processing approximately 3,100 weekly trades while generating unprecedented volumes. However, after this period, when OpenSea regained market dominance, Meebits trading volumes fell by 16 times to levels consistent with normal transaction frequencies.

The economic mechanism driving this exploitation becomes evident when examining the underlying incentive structure. LooksRare's token-based reward system created powerful financial incentives for artificial trading, as users could earn LOOKS tokens proportional to their trading volume. Figure 5.3 reveals the critical connection: the LOOKS/USD exchange rate peaked at approximately 7 USD in late January 2022, coinciding precisely with maximum Meebits trading volumes. As the LOOKS token

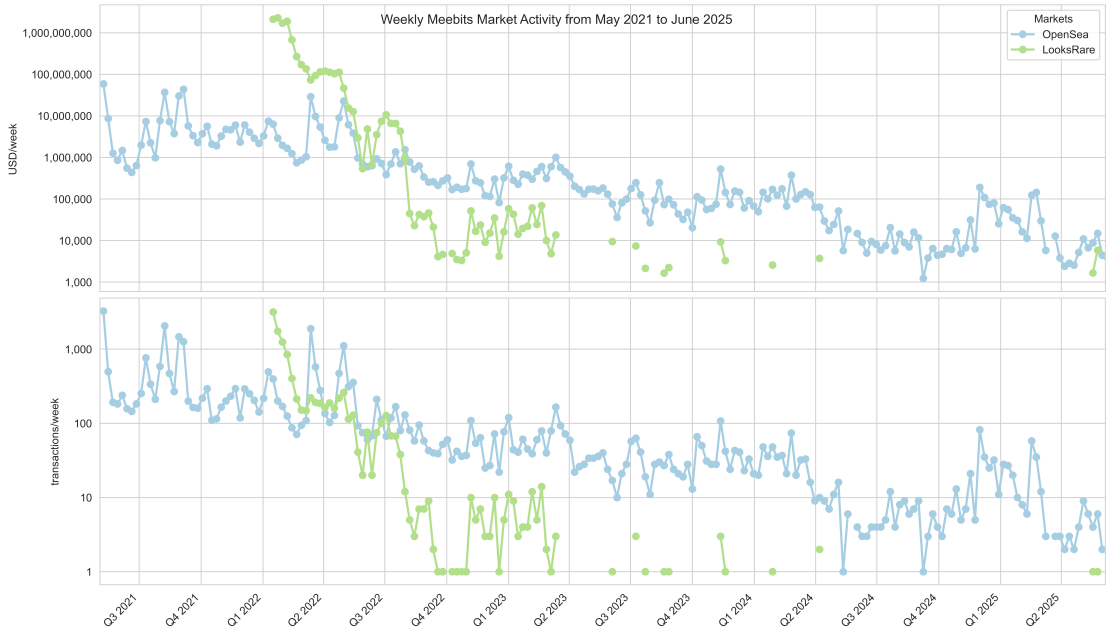

Figure 5.1: Weekly Meebits trading activity on OpenSea and LooksRare from collection launch to June 2025 (logarithmic scale). OpenSea (blue) maintains consistent baseline activity, while LooksRare (green) exhibits extreme volume spikes exceeding 1 billion USD weekly in early 2022.

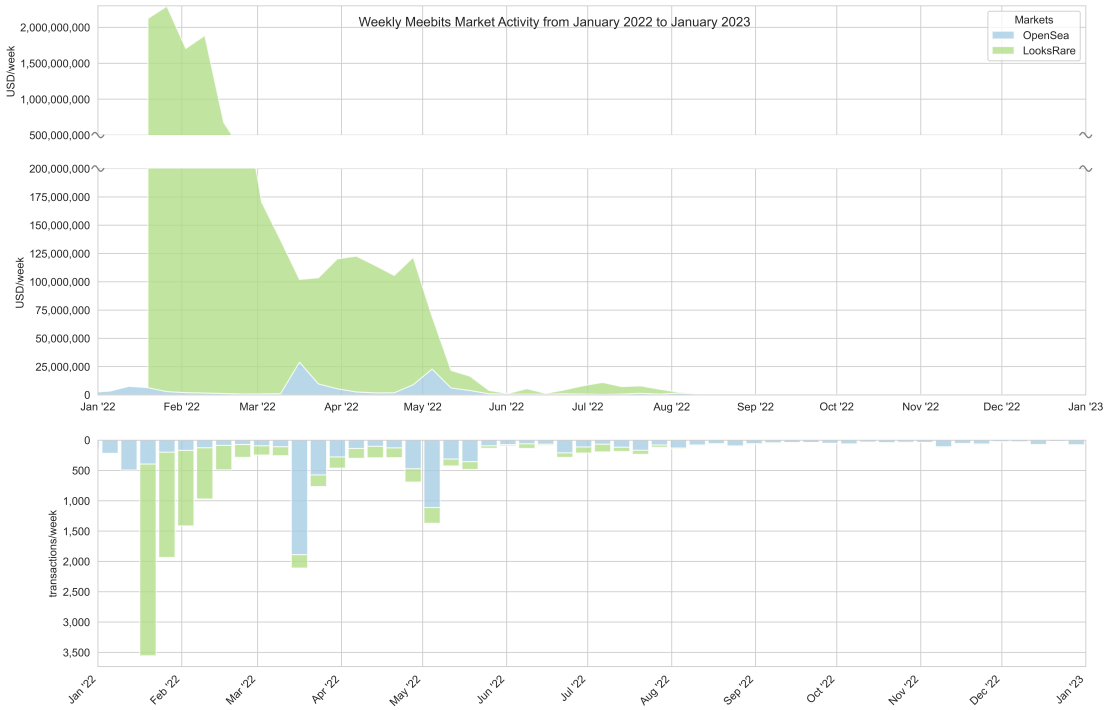

Figure 5.2: Weekly Meebits trading activity in 2022 showing the LooksRare Exploitation Window. Upper panel: USD volume (area chart) with LooksRare (green) dominating mid-January to mid-February 2022. Lower panel: transaction counts demonstrating that peak volumes occurred with relatively few transactions, indicating artificially inflated per-transaction values.

depreciated by 95% through March 2022, the economic incentives driving artificial trading diminished correspondingly. Once LOOKS tokens became nearly worthless, the wash trading schemes that had generated them were no longer profitable — the costs of executing high-volume transactions, while unchanged in LOOKS terms, were worth far less when converted to USD. This temporal alignment between token depreciation and the cessation of anomalous trading activity provides a direct indication that LooksRare's brief market dominance was driven by reward exploitation rather than genuine market demand.

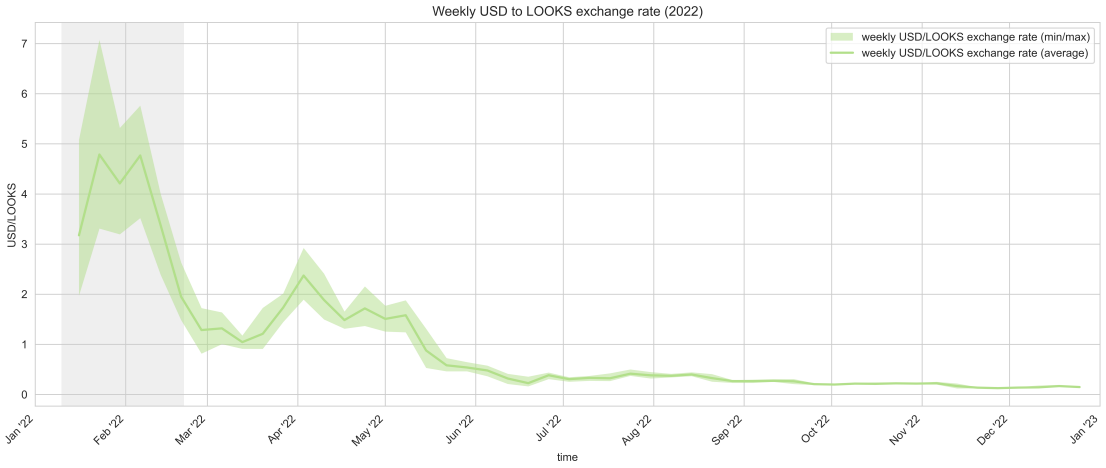

Figure 5.3: LOOKS token price trajectory in 2022. The token peaked at nearly 7 USD during maximum manipulation activity, then declined 95% as exploitation became unprofitable. The highlighted period corresponds to the economically viable exploitation window.

5.2 Actor-Level Behavioral Patterns: Automated Trading Detection

The extraordinary volume patterns documented above raise a critical question whether these trades were executed by genuine market participants or automated systems. To answer this, we turn to actor-level behavioral analysis. The burstiness index analysis of individual trading behaviors provides convincing proof of automated actors operating with computational precision during the manipulation period. Figure 5.4 presents the distribution of sender-centric burstiness indices across all Meebits traders, demonstrating a striking bimodal pattern that distinguishes automated from human trading behaviors.

The most prominent feature is the overwhelming concentration of sender vertices with burstiness indices approaching -1, representing approximately 1,750 addresses (about 10% of all active Meebits traders) exhibiting remarkably regular and repetitive transaction patterns. This level of periodicity is incompatible with human trading behavior, where

genuine collectors typically exhibit irregular patterns driven by market opportunities, personal preferences, and asset availability. The secondary distribution spanning burstiness values from 0.0 to 0.5 represents the more irregular trading patterns consistent with human behavior or manipulation strategies designed to avoid detection. The clear bimodal separation provides quantitative evidence that a substantial portion of Meebits trading was conducted by automated systems rather than human participants. This automated actor identification directly supports the market-level findings: the trading volume anomalies during the LooksRare Exploitation Window required coordinated systematic trading that could only be achieved through automated execution. The presence of approximately a tenth of highly periodic actors provides the mechanism through which the observed market distortions were generated.

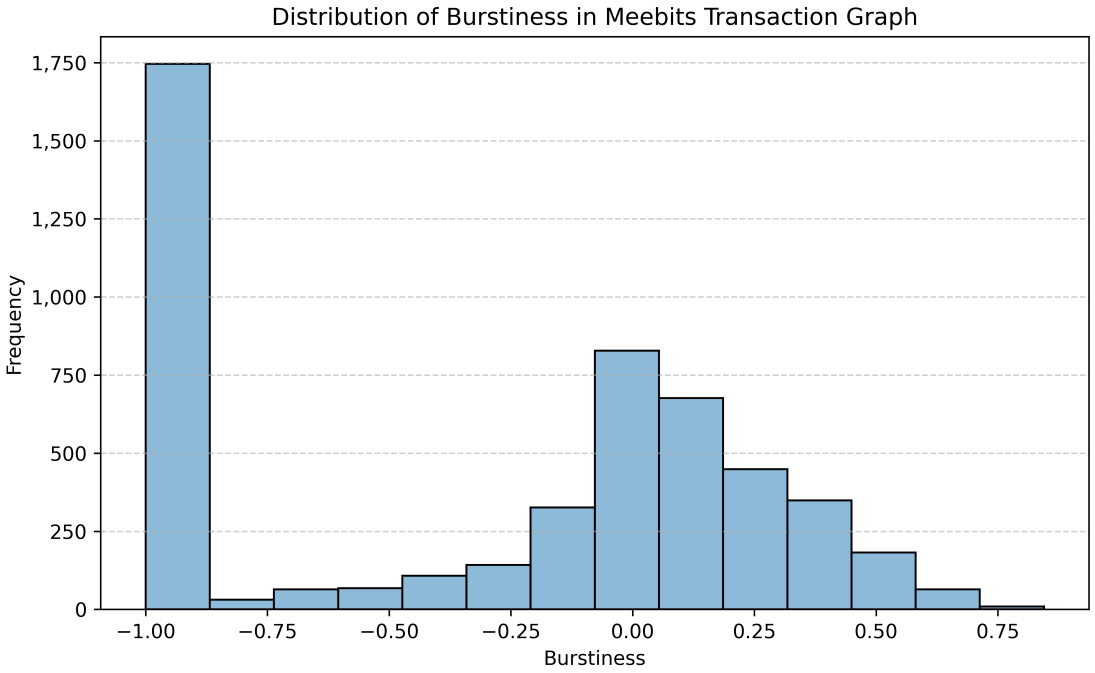

Figure 5.4: Distribution of sender-centric burstiness scores in the Meebits transaction network. The dominant peak at -1.0 represents approximately 1,750 addresses (10% of all traders) exhibiting highly periodic, automated trading patterns. The secondary distribution around 0.0–0.5 represents more irregular human-like behaviors.

5.3 Transaction-Level Manipulation Structures: Cyclic Trading Evidence

While the presence of automated actors explains the mechanism behind the volume anomalies, it does not reveal the specific trading strategies employed. We now examine the underlying transaction structures to uncover how these actors generated artificial volume. The DSCENT algorithm analysis reveals that cyclic trading patterns occurred at multiple temporal ω-thresholds throughout the exploitation period. Table 5.1 presents comprehensive bundled cycle detection results across varying time windows. Initially, from 3–10 minutes, detected cycles remain low with linear-like growth, as these short time frames capture only a small fraction of completed trading cycles. The detection rate then transitions into an accelerated growth phase from 10–30 minutes, with the number of detected cycles increasing substantially. This slightly longer cycle formation is most likely due to the usage of low-priority Ethereum transactions (mentioned in Section 3.2). Malicious actors engaged in wash trading avoid priority tips to minimize transaction costs, as the cumulative expense across thousands of circular trades would otherwise erode the profitability of reward generation schemes. Consequently, their transactions experience longer processing delays as they compete for block inclusion, causing complete cycles to span the 10–30 minute range rather than completing within shorter windows. Finally, beyond the 2-hour mark, detection rates plateau and reach saturation, suggesting that manipulation cycles have natural duration limits that constrain the detection window.

To capture manipulation structures spanning multiple interconnected cycles, we aggregate those into Chained Components. Table 5.1 shows these components increase steadily from 234 at 3-minute windows to a peak of 665 at 30-minute windows, after which they plateau and slightly decline to 605 at 6-hour windows. The subsequent plateau and contraction beyond 30 minutes suggests practical limits to the number of independent manipulation campaigns that were observed. The consolidation ratios — measuring cycles per component — reveal the density of these chained structures and follow a similar three-phase scaling pattern compared to the bundled cycles count. With consolidation ratios ranging from ≈ 4 to ≈ 77, and the combination of plateauing component and high simple cycle counts, demonstrate that the same actors repeatedly participate across multiple bundled cycles within short time frames which suggests coordinated campaigns rather than coincidental cycle formation.

Shifting focus to the temporal positioning of the simple bundled transaction cycles for their respective ω-thresholds, Figure 5.5 reveals the concentration of detected cycles and demonstrates that over 95% occurred within the first three quarters of 2022. Cycle counts peaked during the LooksRare Exploitation Window for longer detection windows ($\omega > 30$ min) — a pattern plausibly driven by human-in-the-loop transactions interleaved with programmatically executed trades, lengthening inter-trade intervals and delaying cycle completion — particularly while LOOKS retained sufficient USD value to keep the strategy profitable. The distinctive inverse sawtooth pattern for shorter windows ($\omega \leq 10$ min) reveal systematic and automated manipulation cycles that commence with high activity, gradually decline as opportunities become exhausted, then briefly resurge.

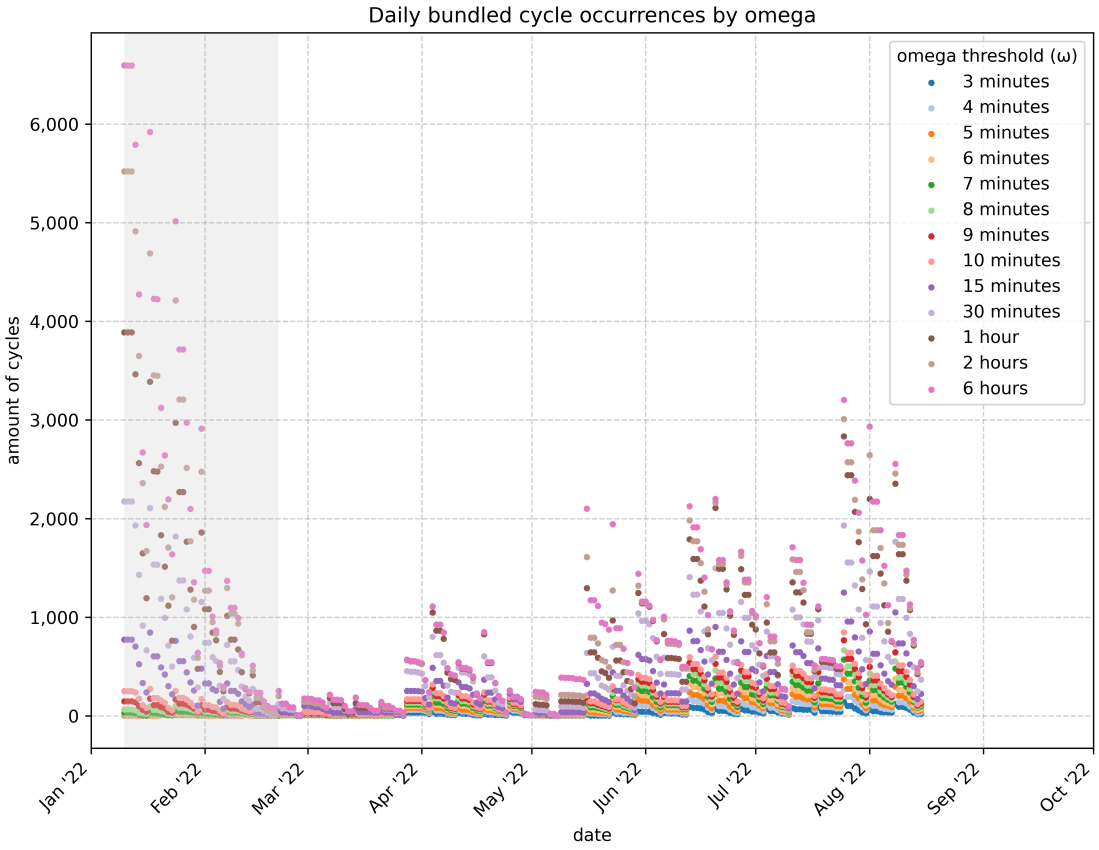

Figure 5.5: Daily bundled cycle occurrences by time window. The grey shaded region indicates the LooksRare Exploitation Window. The inverse sawtooth pattern suggests systematic manipulation campaigns with periodic resurgence.

Omega	Bundled Cycles	Chained Components	Consolidation Ratio	Trivial Components
3 min	927	234	3.96	—
4 min	1.8k	271	6.64	—
5 min	2.7k	299	9.03	2
6 min	3.6k	326	11.04	5
7 min	4.6k	348	13.22	21
8 min	5.6k	396	14.14	59
9 min	6.6k	446	14.80	110
10 min	7.7k	506	15.22	161
15 min	13.0k	611	21.28	261
30 min	23.7k	665	35.64	324
1h	34.0k	643	52.88	314
2h	41.1k	625	65.76	302
6h	46.6k	605	77.02	294

Table 5.1: Manipulation detection results across temporal windows showing bundled cycles, chained components, consolidation ratios, and trivial components.

Beyond temporal concentration, the temporal persistence of these chained components reveals the true duration of coordinated campaigns. Figure 5.6 presents the distribution of temporal expansion ratios across different omega thresholds and ratio groups, demonstrating how wash trading campaigns extend beyond their initial detection windows.

For shorter omega thresholds ($\omega \leq 15$ minutes), the vast majority of components (55-70%) remain confined within their threshold with expansion ratios ≤ 1, indicating no temporal extension beyond their detection windows. Very few components at these thresholds achieve expansion ratios of at least 2. However, a shift occurs at thresholds $\omega = 30$ minutes and beyond. While fewer components maintain ratios ≤ 1, progressively more achieve ratios exceeding 2 — ranging from about 17% at 30 minutes to 85% at 6 hours — with some reaching expansions of 5 and higher. This might suggest that components at lower thresholds cannot be linked together due to either frequent address disposal and account regeneration or deliberate obfuscation methods that intentionally fragment components and only become visible when larger windows are observed, or an orchestrator is caught distributing capital among automated accounts.

While these duration patterns reveal that complex coordination mechanisms exist, identifying the simplest, non-obfuscated manipulation structures can expose the advancement fraudulent actors underwent.

The analysis of the simplest manipulation structures provides more insight about the LooksRare Exploitation Window. These trivial components involve exactly two actors repeatedly exchanging a single token, representing the most basic form of wash trading detected. Figure 5.7 provides compelling visual evidence of the relationship between trivial components and the exploitation period.

The data shows that simple wash trading patterns are almost exclusively concentrated

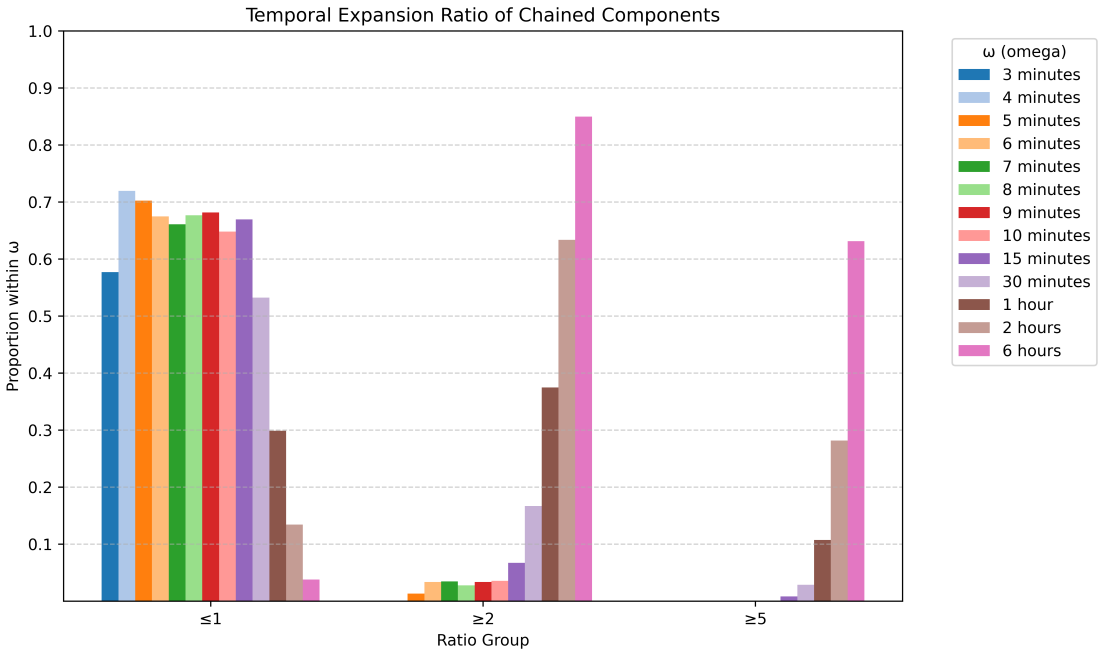

Figure 5.6: Distribution of temporal expansion ratios for chained components across omega thresholds. The expansion ratio measures how far coordinated campaigns persist beyond their initial detection window.

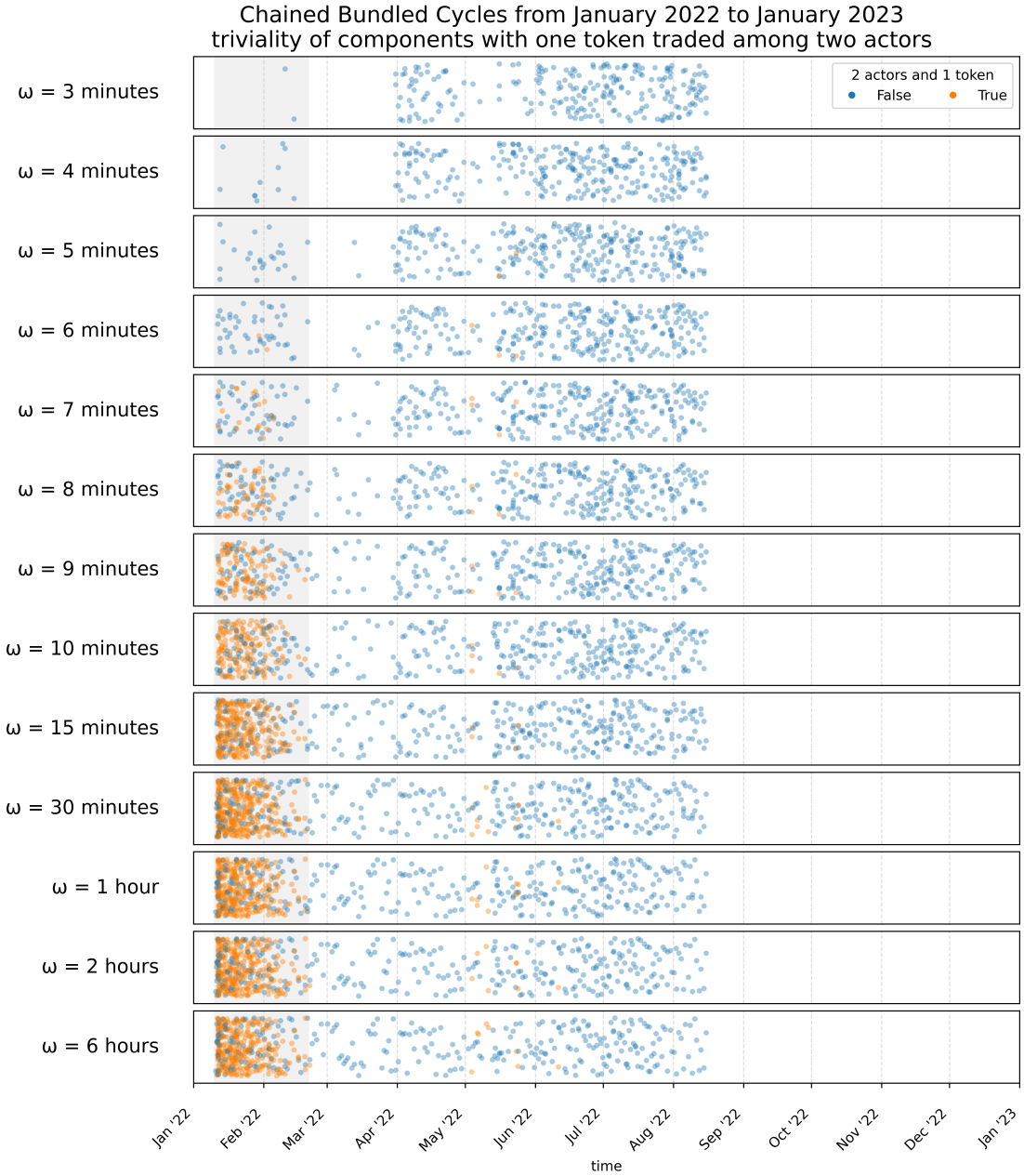

Figure 5.7: Temporal occurrence of chained components from January-September 2022. Orange points indicate trivial components (simple wash trading), while blue points represent complex patterns. Trivial components cluster exclusively within the LooksRare Exploitation Window (grey region).

within the first two months of 2022, creating a clear temporal signature. These patterns appear suddenly in January 2022 and vanish by March 2022, providing an indication that simple wash trading was primarily enabled by LooksRare's token incentive mechanism.

The abrupt disappearance of 2-cycle trading in March 2022 strongly suggests that market operators implemented countermeasures against these simplistic manipulation strategies. This intervention likely included detection algorithms for self-trading patterns that made trivial cycles unprofitable. Following this enforcement period, virtually all detected components are non-trivial, indicating a fundamental shift in manipulation tactics and manipulators were forced to adopt more sophisticated strategies once both the economic incentives for simple manipulation were eliminated and active countermeasures were deployed. This evolution from trivial to complex manipulation patterns represents an arms race between market operators and manipulators.

5.4 Experimental Setup

The convergence of evidence across all three analytical levels — market, actor, and transaction — conclusively demonstrates systematic manipulation during the LooksRare exploitation period. We now evaluate the computational efficiency of our detection methodology to establish its practical viability for real-time monitoring.

The experimental evaluation explores two critical facets of DSCENT's performance characteristics. First, we conduct comparative benchmarking against the existing 2SCENT algorithm using the Meebits dataset to establish performance differentials and validate detection accuracy. Second, we assess the algorithm's scalability through comprehensive stress testing on the complete Ethereum dataset, evaluating performance under extreme computational demands. All experiments were conducted on a standardized high-performance computing environment featuring an Intel Xeon Gold 6226R processor with 16 cores operating at 2.90GHz and 252 GiB memory.

Given that DSCENT extends and builds upon the foundational architecture of 2SCENT — inheriting its concept of reverse-reachability summary mechanism while introducing streaming-specific innovations — we conduct our performance evaluation exclusively against this baseline algorithm. This focused comparison directly isolates the performance impact of DSCENT's architectural enhancements, providing clear attribution of improvements to specific design decisions.

The benchmark comparison between 2SCENT and DSCENT on the Meebits dataset required careful consideration of their definitional differences. The 2SCENT algorithm employs a relaxed temporal ordering constraint where successive edges in a temporal path $p : u \xrightarrow{t} v \xrightarrow{t'} w$ permit $t \leq t'$, while DSCENT enforces strict temporal ordering with $t < t'$. This distinction results in 2SCENT detecting a superset of cycles, as its relaxed constraint identifies cycles where edges occur simultaneously. However, this broader detection scope incurs higher computational costs compared to DSCENT's more restrictive approach.

Each algorithm configuration followed a standardized evaluation protocol. Testing began with initialization in a clean memory state, followed by algorithm execution across multiple omega thresholds. Each ω-threshold underwent 12 iterations to ensure statistical

reliability. For 2SCENT, a separate monitoring thread tracked memory usage and output file metrics at one-second intervals throughout execution, operating independently to avoid interference with the main computation. To balance memory efficiency against computational overhead from graph restructuring, the pruning mechanism activated after processing every 10 million edges.

For 2SCENT, the external monitoring system captured four primary performance metrics: throughput (edges processed per second using a 15-minute rolling average), memory utilization (sampled at 10-minute intervals), task queue depth (indicating active and pending cycle detection tasks), and cycle yield (quantifying bundled cycles detected per ω-window). In contrast, DSCENT measured these same four metrics — throughput, memory utilization, task queue depth, and cycle yield — internally through its built-in Performance Monitoring Framework rather than requiring external instrumentation, with correctness verified through sampling. Results are reported as arithmetic means across runs, with the multiple execution approach enabling outlier identification and ensuring that metrics reflect typical performance characteristics.

The stress testing phase evaluated DSCENT's scalability using 1.8 billion transactions from the Ethereum blockchain. Testing parameters spanned omega thresholds from 2 to 50 Ethereum blocks, corresponding to temporal tolerances of approximately 24 seconds to 10 minutes. This range captures both fine-grained temporal patterns and longer-term cyclic behaviors, providing comprehensive insight into the algorithm's performance across diverse operational scenarios.

5.5 Algorithm Performance Evaluation

With the experimental infrastructure in place, we first assess DSCENT's performance relative to existing methodologies by directly comparing it with the 2SCENT algorithm. This comparison provides essential context for the algorithmic improvements and sets baseline expectations for the subsequent stress-test analysis.

Omega	**2SCENT**	**DSCENT**
3 min	45s	3s
4 min	1 min	5s
5 min	2 min	7s
6 min	3 min	9s
7 min	4 min	10s
8 min	5 min	11s
9 min	6 min	13s
10 min	8 min	15s
15 min	16 min	22s
30 min	42 min	38s

Table 5.2: Runtime comparison for 2SCENT and DSCENT algorithms across varying ω-thresholds.

Table 5.2 presents runtime performance across temporal thresholds ranging from 3 to 30 minutes, revealing substantial efficiency gains achieved by DSCENT. Both algorithms exhibit similar scaling characteristics relative to the ω-threshold parameter. As temporal windows expand, both experience comparable computational challenges with similar relative growth trajectories. However, DSCENT's consistently superior absolute performance stems from its more restricted temporal ordering constraints and streamlined processing pipeline. The temporal dynamics of cycle discovery reveal fundamental algorithmic differences, as illustrated in Figure 5.9. DSCENT operates within a remarkably stable memory footprint of several hundred megabytes across all tested thresholds, while 2SCENT exhibits memory consumption exceeding multiple gigabytes. The contrasting behavior with increasing temporal thresholds further highlights architectural differences. 2SCENT demonstrates steep monotonic increases in memory consumption with larger thresholds, suggesting potential scalability limitations for long-duration temporal analysis. Conversely, DSCENT exhibits counterintuitive behavior where higher thresholds initially result in lower memory consumption before stabilizing at approximately 253 MB, though this pattern may be attributable to the coarse granularity of MB-level measurements — lower thresholds should theoretically also start near zero memory usage, but capturing such fine-grained differences is inherently more challenging when measuring in megabytes rather than gigabytes where variations are more pronounced. DSCENT achieves significant memory optimization through its storage architecture. Rather than accumulating increasingly large data structures like 2SCENT, DSCENT maintains bounded memory usage through its streaming processing approach and resource management. Additionally 2SCENT stores reverse-reachability summaries as individual (node, timestamp) tuples, whereas DSCENT employs a map structure that associates a single timestamp with multiple nodes. This consolidated representation reduces memory overhead, particularly for scenarios where multiple reverse reachable nodes share the same starting timestamp.

Exploring the output dynamics, the comparison reveals distinctly different execution profiles. 2SCENT exhibits a characteristic two-phase pattern: an initial phase with no cycle production while constructing internal data structures, followed by rapid exponential cycle yield once processing commences. This batch-oriented approach delays result availability until significant computational work is completed. In contrast, DSCENT demonstrates immediate and continuous cycle production from initialization, following a more gradual but consistent growth trajectory. This streaming capability is enabled by the binary tree structure used to track rooted seeds, which allows efficient retrieval and deletion of seeds that are ready for processing. This design enables early result availability and supports progressive analysis scenarios where immediate feedback provides value. The algorithm's ability to incrementally produce results while maintaining computational efficiency represents a significant advancement for interactive and real-time applications.

Having demonstrated DSCENT's superior efficiency relative to 2SCENT across varying temporal thresholds, we now examine the algorithm's behavior under real-world computational demands. The following stress test analysis evaluates DSCENT's performance when processing comprehensive Ethereum transaction data, revealing both its capabilities and limitations when handling large-scale blockchain datasets with varying degrees of

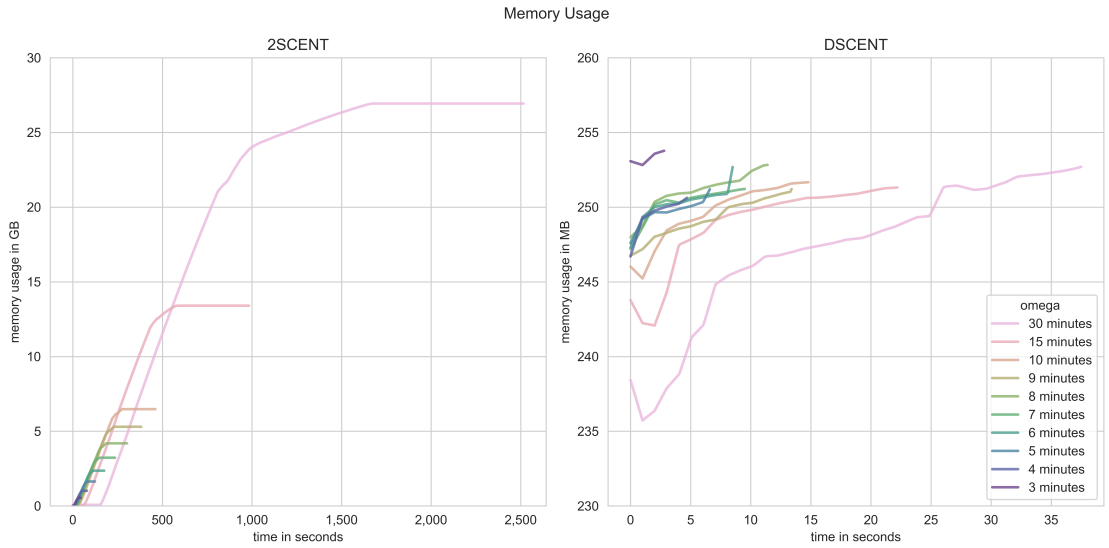

Figure 5.8: Memory consumption profiles for 2SCENT and DSCENT algorithms. 2SCENT (left) shows rapid memory growth with increasing temporal thresholds, while DSCENT (right) maintains stable memory footprint across all ω-threshold values, demonstrating superior scalability characteristics.

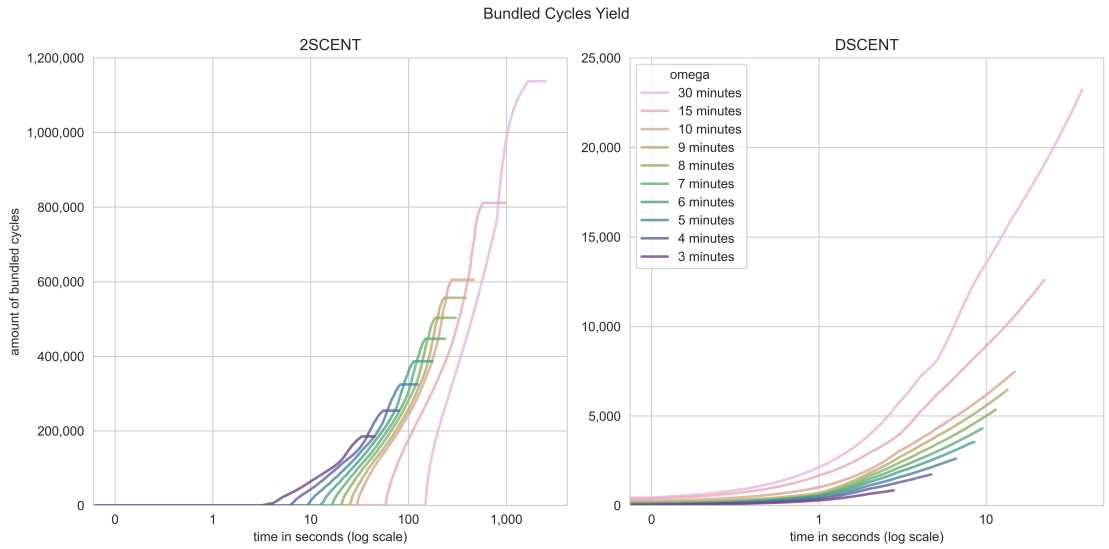

Figure 5.9: Cumulative bundled cycle yield over execution time for varying temporal thresholds. 2SCENT (left) displays delayed cycle production followed by rapid growth on logarithmic time scale. DSCENT (right) demonstrates immediate and progressive cycle discovery with consistent growth patterns.

cyclic activity.

Table 5.3 summarizes the key performance metrics across different omega thresholds, ranging from 2 blocks (≈24 seconds) to 50 blocks (≈10 minutes) using comprehensive Ethereum transaction data.

Omega (blocks)	Duration	Average Throughput (edges/sec)	Average Memory (GB)	Bundled Cycles
2	1d 10h	75.4k	71.5	14.4k
3	1d 7h	76.8k	69.8	30.4k
4	1d 7h	77.1k	65.3	52.5k
5	1d 6h	77.5k	65.1	79.4k
10	1d 6h	77.4k	69.3	261.1k
20	1d 9h	76.0k	83.1	1.08M
30	1d 12h	74.7k	86.8	2.93M
40	1d 15h	73.3k	83.9	6.99M
50	1d 20h	71.9k	85.4	20.4M

Table 5.3: ETH transaction processing stress test results across different omega block intervals.

The experimental results reveal distinct performance patterns that provide crucial insights into optimal algorithm configuration. Counterintuitively, smaller omega values ($\omega = 2$ and $\omega = 3$) require longer processing times despite handling smaller temporal windows. This occurs because transactions that could be efficiently aggregated within larger temporal windows are instead processed as separate computational jobs, creating significant overhead. The algorithm's efficiency benefits from intelligent batching that becomes possible with slightly larger temporal windows. A clear performance sweet spot emerges at $\omega = 5$ and $\omega = 10$ blocks, where DSCENT achieves an average throughput of approximately 77.5k edges per second while maintaining moderate memory consumption between 65-69 GB. Both configurations complete processing in just 1 day and 6 hours, representing the fastest processing times across all tested parameters. The relationship between temporal window size and cycle detection capability demonstrates exponential scaling characteristics. Detected bundled cycles increase sharply from 14.4k cycles at $\omega = 2$ to 20.4M cycles at $\omega = 50$ — approximately a 1,400x increase.

Figure 5.10 reveals how cycle detection efficiency varies throughout the dataset timeline, exposing underlying patterns in trading behavior. One remarkable anomalous period emerges from the temporal analysis: a significant peak that coincides with the LooksRare Exploitation Window. While larger omega values ($\omega = 40$ and $\omega = 50$) capture the highest absolute number of cycles during this period, the $\omega = 20$ configuration demonstrates a pronounced sensitivity for this event.

The parallel processing capabilities of DSCENT were challenged during periods of high cyclic activity, particularly during the parsing of transactions from January 31, 2022. Under normal operating conditions, the work queue maintains minimal depth,

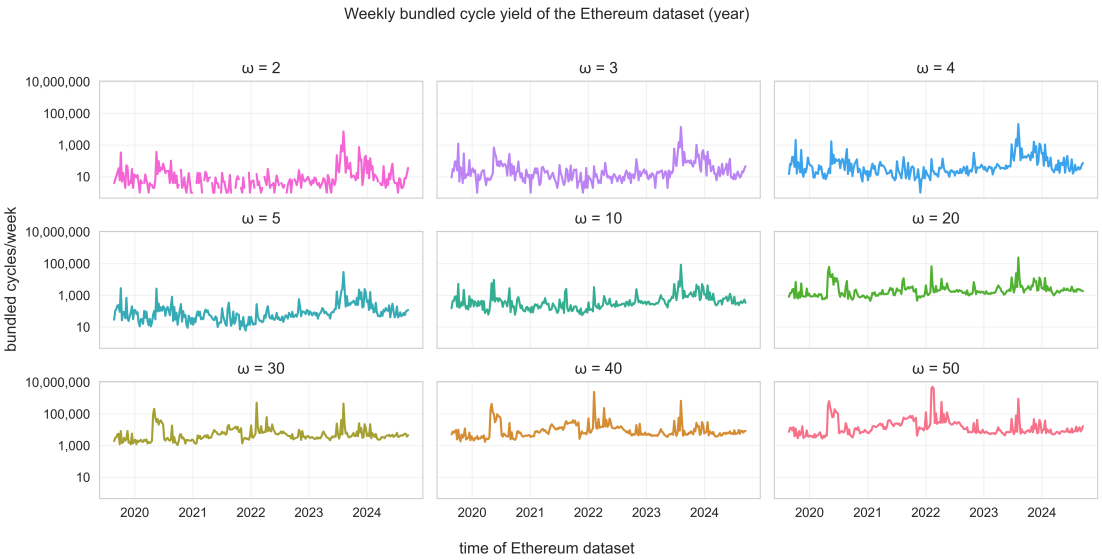

Figure 5.10: Weekly bundled cycle yield (cycles/week) on logarithmic scale for different omega values across the Ethereum dataset timeline (2020-2024). Notable spike in late January 2022 corresponds to periods of increased cyclic trading activity, with $\omega = 20$ showing pronounced sensitivity to this anomalous region.

indicating efficient task processing without significant backlog accumulation. However, an anomaly on that date occurred which represents the singular instance where the system experienced substantial stress. During this atypical period, parallel task queues reached 4,000-6,000 simultaneous items for $\omega = 40$ and $\omega = 50$ configurations. While the algorithm demonstrated robustness by processing this surge without failure, the computational cost of handling highly cyclic regions with large temporal windows becomes evident. Figure 5.11 illustrates how system throughput responds to varying computational demands throughout the processing timeline. Most configurations maintain remarkably stable performance with consistent throughput between 60-80k edges per second, yet the performance contours reveal distinct structural variations during critical processing phases. The temporal curvature between performance peaks and valleys reflects different processing speeds and arrival times at the dataset's anomalous regions. Two significant throughput degradations emerge during processing of the early-2022 anomaly: $\omega = 40$ experiences a 2-3 hour performance valley at approximately 18 hours, while $\omega = 50$ encounters a more severe 4-hour degradation around the 20-hour mark. The extended duration of throughput reduction for $\omega = 50$ correlates directly with its higher task queue depth, requiring substantially more processing time to clear the accumulated computational workload.

Memory consumption patterns provide additional insights into algorithm behavior under varying omega configurations, as shown in Figure 5.12. All configurations exhibit

similar memory usage patterns during the initial 15-hour processing window, following a classic sawtooth pattern as pruning operations trigger every 10 million edges processed. This regular pruning cycle creates predictable memory oscillations, with gradual increases as edges accumulate followed by sharp decreases when obsolete graph structures are removed. Larger omega values require approximately 10% additional memory overhead due to proportionally larger reverse-reachability summaries and more expanded temporal graph structures needed for parallel computations, indicating that the algorithm's memory behavior remains relatively stable during normal operation.

However, memory consumption patterns diverge significantly once the early-2022 anomaly threshold is crossed. A notable memory plateau occurs around the 18 to 20 hour mark for both $\omega = 40$ and $\omega = 50$ configurations. Despite continued processing activity, memory usage stabilizes during the throughput degradation period, as computational resources become dedicated to cycle detection rather than graph expansion. Following this plateau, the $\omega = 40$ configuration exhibits a sharp memory decrease around the 25-hour mark — more pronounced than the regular pruning-induced drops — signaling completion of the computational surge induced by the transactions of January 2022 and successful memory cleanup.

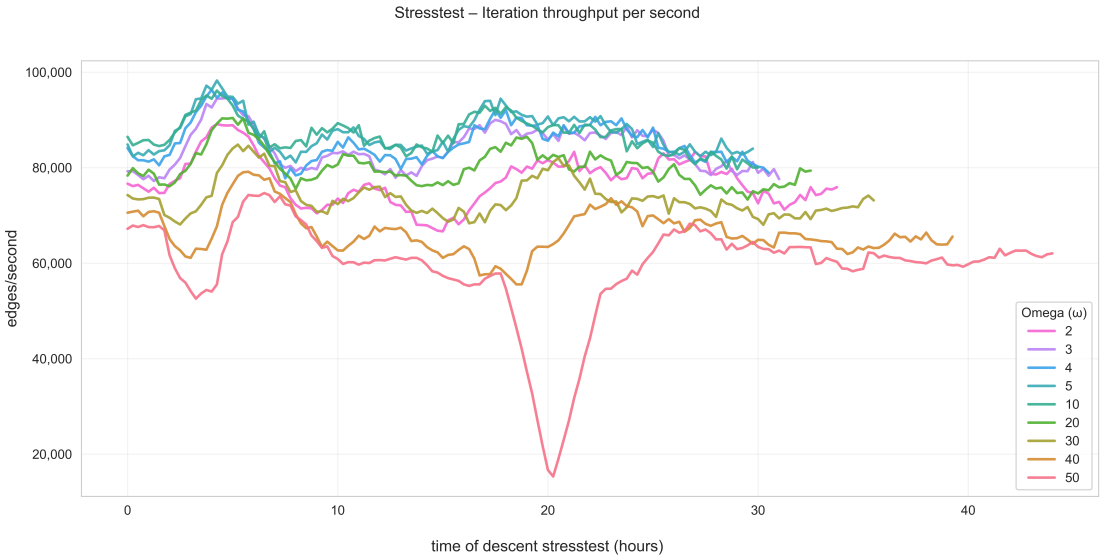

Figure 5.11: Iteration throughput per second across different omega thresholds during the stress test. Measurements show 15-minute rolling averages to smooth short-term fluctuations. Notable performance drops for $\omega = 40$ and $\omega = 50$ around 18-hour and 20-hour marks corresponds to processing the early-2022 anomaly period.

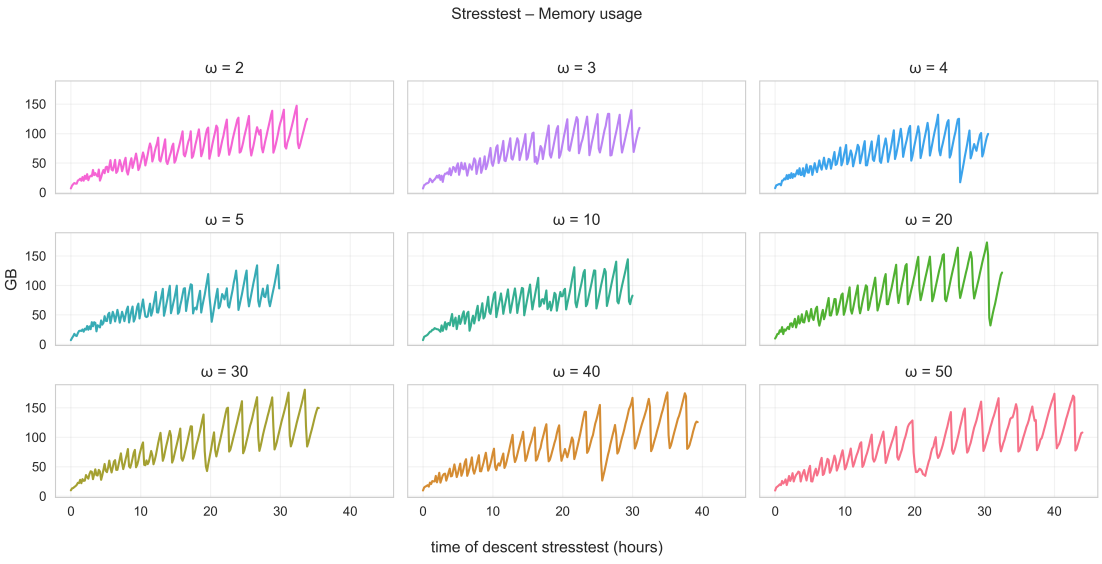

Figure 5.12: Memory usage over time (10-minute averages) for different omega configurations. Sawtooth patterns indicate regular pruning cycles, with divergence after ≈15 hours showing the impact of larger temporal windows on memory requirements.

6 Discussion and Future Directions

Applying DSCENT to the Meebits trading ecosystem illuminated the techniques behind the LooksRare exploitation during Q1 2022, revealing how platform token rewards distorted organic market behavior across three key dimensions: temporal periodicity, structural patterns, and concentrated trading activity. This real-world validation demonstrates the critical need for sophisticated detection mechanisms that can operate in dynamic market environments, moving beyond traditional static analysis approaches that rely on batch processing of historical data.

DSCENT's dynamic approach addresses this need by representing a fundamental advancement in market surveillance capabilities through its streaming architecture that continuously outputs detection results while managing memory through strategic pruning. The algorithm's separation of transaction update and seed generation phases provides crucial architectural flexibility, and by aggregating transactions to single temporal points, it significantly reduces complexity while maintaining detection accuracy, as demonstrated through Ethereum stress testing that processed 1.8 billion edges in 30–44 hours and achieved optimal performance at $\omega = 20$ blocks with a throughput of 76,000 transactions per second. This design choice opens future possibilities for further timestamp discretization to enable higher omega threshold detection windows.

These proven manipulation detection capabilities extend far beyond NFT markets, opening pathways to broader surveillance applications across diverse transaction ecosystems. DSCENT's ability to identify circular trading patterns positions it as a versatile tool for detecting inter-collection coordination schemes, monitoring suspicious activity in credit card networks and currency exchanges, and surveilling any system where manipulation may manifest through coordinated transaction flows.

6.1 Limitations and Future Work

Despite these achievements, the analysis remains constrained by focusing exclusively on trivial components — elementary manipulation tactics consisting of pairwise token swaps between two entities. While these structures provide clear evidence during the LooksRare period, they represent merely a visible portion of a potentially vast manipulation ecosystem. More sophisticated wash trading schemes involving multiple intermediate addresses, complex token routing, or deliberately fragmented transaction sequences likely exist but evade the current analysis approach. This limitation is compounded by the absence of definitive ground truth, which presents the most critical barrier to advancing the wash trading detection capabilities. However, if these insights become available through regulatory actions or exchange cooperation, supervised learning approaches could complement

structural detection with behavioral analysis, significantly enhancing detection capabilities beyond simple cycle identification.

The current pruning strategy exhibits design limitations that create unnecessary memory overhead despite preventing unbounded growth under normal conditions. The algorithm maintains the entire temporal graph spanning from the oldest seed processing task to the current timestamp, creating unnecessary memory overhead. The January 31, 2022 anomaly, where work queue accumulation reached 4,000-6,000 tasks, exposes this vulnerability. A more advanced approach would partition the in-memory graph structure more effectively, maintaining only the actively relevant portions while discarding obsolete segments. This enhancement would improve both memory efficiency and therefore processing speed during extreme market events when surveillance is most critical.

Scaling DSCENT beyond its current limitations demands fundamental architectural improvements that prioritize distributed processing capabilities. By parallelizing seed exploration tasks across multiple nodes, we could achieve uninterruptible processing capacity with additional computational resources. This distributed approach necessitates research into effective sharding strategies for the temporal graph structure — partitioning the network while maintaining consistency guarantees for seed exploration across computation node boundaries.

The strict temporal ordering constraint ($t < t'$) represents a deliberate design choice reflecting assumptions about capital movement in wash trading. However, professional actors distributing capital across multiple addresses could execute simultaneous transactions within the same block, patterns that would require relaxing this constraint.

Parameter sensitivity remains a challenge, with the choice of temporal window ω fundamentally influencing detection results. The steep increase from 2.8k cyclic Meebits trades at 3 minutes to 1.6M at 6 hours illustrates this sensitivity. Adaptive parameter adjustment based on system load and market conditions — tightening windows during volatile periods while scaling back during quiet markets or periods of low system utilization — could optimize detection effectiveness while managing computational resources.

6.2 Conclusion

This research demonstrates how algorithmic approaches can systematically identify market manipulation patterns in digital asset ecosystems, providing crucial insights for maintaining market integrity in an era of accelerated financial markets. The analysis reveals how economic incentives can measurably distort trading behavior, highlighting the need for sophisticated detection mechanisms that evolve alongside manipulation strategies. While current methods address fundamental manipulation patterns, the dynamic surveillance framework established here points toward more comprehensive monitoring systems capable of adapting to increasingly complex schemes. The future of market oversight depends on developing detection capabilities that match the sophistication of malicious actors who continuously refine their approaches to circumvent traditional surveillance. Through advances in distributed processing, machine learning integration, and real-time analysis, surveillance systems can maintain fairness across diverse transaction domains — from

digital assets to traditional financial markets. As financial ecosystems become more complex and interconnected, the principles and methodologies developed in this work contribute to the preservation of transparent and equitable trading environments that protect market participants and preserve confidence in financial institutions.

Bibliography

[ABDY15] Sumit Agarwal, Itzhak Ben-David, and Vincent Yao. Collateral valuation and borrower financial constraints: Evidence from the residential real estate market. *Management Science*, 61(9):2220–2240, 2015.

[Ant22] Lennart Ante. The non-fungible token (NFT) market and its relationship with Bitcoin and Ethereum. *FinTech*, 1(3):216–224, 2022.

[ARO+22] Abdulalem Ali, Shukor Abd Razak, Siti Hajar Othman, Taiseer Abdalla Elfadil Eisa, Arafat Al-Dhaqm, Maged Nasser, Tusneem Elhassan, and Abdu Saif. Financial fraud detection based on machine learning: A systematic literature review. *Applied Sciences*, 12(19), 2022.

[BAI23] Jovan Blanuša, Kubilay Atasu, and Paolo Ienne. Fast parallel algorithms for enumeration of simple, temporal, and hop-constrained cycles. *CoRR*, abs/2301.01068, 2023.

[BIA24] Halima Oluwabunmi Bello, Adebimpe Bolatito Ige, and Maxwell Nana Ameyaw. Adaptive machine learning models: Concepts for real-time financial fraud prevention in dynamic environments. *World Journal of Advanced Engineering Technology and Sciences*, 12(2):021–034, 2024.

[CLC+16] Y. Cao, Y. Li, S. Coleman, A. Belatreche, and T. M. McGinnity. Detecting wash trade in financial market using digraphs and dynamic programming. *IEEE Transactions on Neural Networks and Learning Systems*, 27(11):2351–2363, 2016.

[CLTY23] Lin William Cong, Xi Li, Ke Tang, and Yang Yang. Crypto wash trading. *Management Science*, 69(11):6427–6454, 2023.

[CZWL24] S. Chen, J. Zhang, L. Wang, and H. Liu. AI-powered fraud detection in decentralized finance: A project life cycle perspective. *ACM Computing Surveys*, 57(2):1–35, 2024.

[DSE25] S. M. Darwish, A. I. Salama, and A. A. Elzoghabi. Intelligent approach to detecting online fraudulent trading with solution for imbalanced data in fintech forensics. *Scientific Reports*, 15:17983, 2025.

[Eth13] Ethereum Foundation. Go Ethereum, 2013.

[Eth15] Etherscan. Etherscan: Ethereum blockchain explorer, 2015.

[Fed25] Federal Bureau of Investigation. Property flipping database privacy impact assessment. Criminal Investigative Division, Financial Crimes Section, 2025.

[Fin13] Financial Conduct Authority. Market abuse regulation (MAR) 1.6, 2013.

[GB08] K.-I. Goh and A.-L. Barabási. Burstiness and memory in complex systems. *EPL (Europhysics Letters)*, 81(4):48002, 2008.

[IT18] Serkan İmişiker and Bedri Kamil Onur Taş. Wash trades as a stock market manipulation tool. *Borsa Istanbul Review*, 20:92–98, 2018.

[Joh75] Donald B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM J. Comput.*, 4(1):77–84, 1975.

[KC18] Rohit Kumar and Toon Calders. 2SCENT: An efficient algorithm to enumerate all simple temporal cycles. *Proc. VLDB Endow.*, 11(11):1441–1453, 2018.

[KG24] S. Khodabandehlou and A. Golpayegani. Fifraud: Unsupervised financial fraud detection in dynamic graph streams. *ACM Transactions on Knowledge Discovery from Data*, 18(5):1–29, 2024.

[KM23] Elie Kapengut and Bruce Mizrach. An event study of the Ethereum transition to proof-of-stake. *Commodities*, 2(2):96–110, 2023.

[LC25] Jingchen Liu and Yao Cui. NFT market design: Resale royalty, decentralization, and interoperability. 2025.

[McK24] McKinsey & Company. Global payments in 2024: Simpler interfaces, complex reality, 2024.

[Mic16] Othon Michail. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, 12(4):239–280, 2016.

[MMMN23] Massimo La Morgia, Alessandro Mei, Alberto Maria Mongardini, and Eugenio Nerio Nemmi. A game of NFTs: Characterizing NFT wash trading in the Ethereum blockchain. In *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, pages 13–24, 2023.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[PGM+21] Ismini Psychoula, Andreas Gutmann, Pradip Mainali, Sharon H. Lee, Paul Dunphy, and Fabien Petitcolas. Explainable machine learning for fraud detection. *Computer*, 54(10):49–59, October 2021.

[PZL+19] You Peng, Ying Zhang, Xuemin Lin, Wenjie Zhang, Lu Qin, and Jingren Zhou. Hop-constrained s-t simple path enumeration: Towards bridging theory and practice. *Proc. VLDB Endow.*, 13(4):463–476, 2019.

[QCQ+18] Xiafei Qiu, Wubin Cen, Zhengping Qian, You Peng, Ying Zhang, Xuemin Lin, and Jingren Zhou. Real-time constrained cycle detection in large dynamic graphs. *Proc. VLDB Endow.*, 11(12):1876–1888, 2018.

[RT75] Ronald C. Read and Robert E. Tarjan. Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. *Networks*, 5(3):237–252, 1975.

[TVH25] A. Tošić, J. Vičič, and N. Hrovatin. Beyond the surface: advanced wash-trading detection in decentralized NFT markets. *Financial Innovation*, 11:86, 2025.

[VW19] Friedhelm Victor and Angelika Marsalek Weintraud. Measuring Ethereum-based ERC20 token networks. In *International Conference on Financial Cryptography and Data Security*, pages 113–129. Springer, 2019.

[VW21] Friedhelm Victor and Andrea Marie Weintraud. Detecting and quantifying wash trading on decentralized cryptocurrency exchanges. In *Proceedings of the Web Conference 2021 (WWW '21)*, pages 23–32, New York, NY, USA, 2021. Association for Computing Machinery.

[Woo14] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger, 2014. Ethereum project yellow paper.

[WWY+23] Xiaolin Wen, Yong Wang, Xuanwu Yue, Feida Zhu, and Min Zhu. NFTDisk: Visual detection of wash trading in NFT markets. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 215:1–215:15. ACM, 2023.

[ZKX23] Y. Zhou, M. Kantarcioglu, and B. Xi. A user-centered explainable artificial intelligence approach for financial fraud detection. *Decision Support Systems*, 168:113815, 2023.