Detecting Arbitrage on Ethereum Through Feature Fusion and Positive-Unlabeled Learning

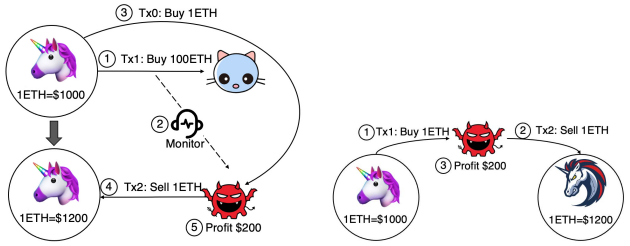
Hai Jin, *Fellow, IEEE*, Chenchen Li, *Student Member, IEEE*, Jiang Xiao, *Member, IEEE*, Teng Zhang, *Member, IEEE*, Xiaohai Dai, *Member, IEEE*, and Bo Li, *Fellow, IEEE*

***Abstract*— Due to the lack of supervision in the *decentralized exchanges* (DEXs), arbitrageurs can utilize information and take advantage of price gap to make profits over such platforms such as Ethereum blockchain. DEX arbitrage poses possibilities and opportunities for defrauding and can seriously impair the operation of the Ethereum ecosystem. It motivates this work to explore and characterize the unique features of arbitrage which differ from other frauds such as money laundering and Ponzi games for better detection. This work makes the first attempt for detecting arbitrage on Ethereum through feature fusion and *positive-unlabeled learning* (PU learning). We first conduct an in-depth analysis and exploit two-fold arbitrage features by fusion including: 1) statistical features that explicitly represent the node activity levels according to expert knowledge; and 2) structural features that implicitly encode the transactions information by graph machine learning. We then apply PU learning to generate negative instances for compensating the imbalanced arbitrage datasets. We evaluate our proposed method through extensive experiments over a real-world dataset and demonstrate that it can achieve 90% accuracy in detecting arbitrage activities on Ethereum.**

***Index Terms*— Arbitrage, Ethereum, feature fusion, PU learning, blockchain.**

I. Introduction

As a *decentralized exchanges* (DEXs) platform, Ethereum blockchain has attracted many investors due to its anonymity [1]. However, anonymity is a double-edged sword. On one hand, it leads to a dramatic growth in crypto markets for decentralized payments and digital assets. On the other hand, it poses significant illicit finance risks and cybercrime [2]. To help regulate the Ethereum ecosystem, many research have been conducted to detect abnormal behaviors such as money laundering [3], [4], [5], [6] and Ponzi games [7], [8]. However, arbitrage, a frequent behavior that makes profits by the price difference, has not been studied comprehensively.



(a) Single-exchange arbitrage (b) Cross-exchange arbitrage

Fig. 1. Two categories of arbitrages.

Arbitrage transactions arise by the price differences between different exchanges at the same time, and due to different supply-demand relationships. Taking "Kimchi premium" as an example, a typical representative of rapid fluctuation of cryptocurrency price, in Jan 2018, the bitcoin prices in South Korea are 43% higher than those in the United States. The price difference will certainly attract speculators to make profits from it. According to [9], the daily potential arbitrage profit is usually more than 75 million U.S. dollars. These arbitrage transactions not only gain the profits of exchanges or normal users deviously, but also exacerbate the congestion of the Ethereum network.

State-of-the-art arbitrage detection [10], [11] have primarily focused on the arbitrage activities in a single exchange as shown in Fig. 1(a), which only capture a small proportion of the arbitrage activities. Even worse, the existing approaches are inadequate for detecting arbitrage transactions across different DEXs, due to the lack of understanding on the unique features of arbitrage activities. Zhou et al. [10] characterized the high gas fee for arbitrage transactions, while it cannot be used for detecting those arbitrage transactions with low gas fees. Flash Boys 2.0 [11] aimed to excavate the potential opportunities for arbitrage instead of detecting the past arbitrage activities, while the latter could be more meaningful for blockchain regulation. To fill this gap, in this paper, we perform a comprehensive characterization of arbitrage transactions on both single-exchange and cross-exchange (shown in Fig. 1(b)).

However, as arbitrage behavior evolves over time and becomes more complex, there are significant challenges in identifying the unique features and distinguishing the arbitrage transactions from other frauds.

Manuscript received 16 March 2022; revised 16 June 2022; accepted 30 June 2022. Date of publication 13 October 2022; date of current version 22 November 2022. This work was supported by the National Key Research and Development Program of China under Grant 2021YFB2700700. *(Corresponding author: Jiang Xiao.)*

Hai Jin, Chenchen Li, Jiang Xiao, Teng Zhang, and Xiaohai Dai are with the Services Computing Technology and System Laboratory, National Engineering Research Center for Big Data Technology and System, and the Cluster and Grid Computing Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: hjin@hust.edu.cn; chenchenli@hust.edu.cn; jiangxiao@hust.edu.cn; tengzhang@hust.edu.cn; xhdai@hust.edu.cn).

Bo Li is with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong (e-mail: bli@cse.ust.hk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2022.3213335.

Digital Object Identifier 10.1109/JSAC.2022.3213335

The first challenge is: how to capture the fast-evolving and heterogeneous features of arbitrage behavior across DEXs? For instance, early arbitrageurs on Ethereum only need to directly buy and sell coins with *Externally Owned Accounts* (EOAs). Due to the rapid development of the Ethereum network and the supervision of DEXs, a majority of current arbitrage behaviors perform by automatic smart contracts and arbitrage bots. In addition, the previous arbitrage detection model fails to discover the heterogeneity consisting of 1) statistical features that explicitly represent the node activity levels according to expert knowledge; and 2) structural features that implicitly encode the transactions information by graph machine learning. As such, the heterogeneous arbitrage features that change dynamically become the performance bottlenecks of the classification model.

The second challenge is: how to compensate for the imbalanced datasets with the absence of negative samples to ensure the effectiveness of the classification model? The effectiveness of the classification model by machine learning suffers from the absence of negative samples. In other words, it requires both positive and negative samples for the training dataset. As of July 21, 2021, 163 million distinct addresses and 1.2 billion transaction records have been recorded in Etherscan. This dataset contains only positive samples and unlabeled samples.

To deal with the above challenges, in this paper, we propose a novel arbitrage addresses detection model by combing heterogeneous feature fusion and *positive-unlabeled learning* (PU learning). We formalize the arbitrage addresses detection as a node classification problem on graphs. Our key insight is that each arbitrage node poses specific network structural characteristics due to its active interaction with other nodes in transaction graphs. We thus exploit structural features by representation learning. Meanwhile, we obtain the inherent statistical features of arbitrage addresses by expert knowledge. We then conduct feature fusion to form the heterogeneous feature vectors.

Heterogeneous feature vectors can not only represent intuitive properties of nodes but also extract the hidden information contained by the transaction graph. Two-fold arbitrage features achieved by feature fusion can better capture the evolving and heterogeneous properties of arbitrage activities. Thus, the first challenge: how to capture the fast-evolving and heterogeneous features of arbitrage behavior across DEXs can be solved. To deal with the absence of negative samples, PU learning is adopted to find reliable negative samples from the unlabeled dataset. In this case, we employ the classic *positive-unlabeled learning* (PU learning) *spy* technique [12] to obtain sufficient reliable negative instances to enrich the dataset, and incorporate them into the XGBoost classifier. In summary, feature fusion and PU learning are leveraged to deal with the two challenges mentioned above. Section III details how we utilize these two methods to overcome difficulties. Our contributions in this work are summarized as follows:

- We make a thorough analysis of arbitrage activities from both statistical and structural features, and conduct feature fusion to characterize behavior patterns of arbitrage addresses that exhibit high utility to detect arbitrage.
- To achieve reliable negative instances from numerous unlabeled Ethereum addresses, we employ PU learning to overcome the lacking negative instances.
- We carry out extensive experiments to verify the feasibility and effectiveness of our detection model, and the results demonstrate that our detection model can achieve 90% precision.

The rest of this paper is organized as follows. Section II introduces the background and related work. Section III presents our detection model with a formal problem description. Methods to extract and fuse the heterogeneous features are proposed in Section IV. We elaborate on our key techniques of PU learning in Section V and conduct classification in Section VI. Section VII shows our evaluation that demonstrates the effectiveness of our approach and Section VIII concludes the paper.

II. Background and Related Work

This section includes background Ethereum blockchain and surveys existing solutions to abnormal behaviors on the blockchain, especially arbitrage detection.

A. Ethereum Blockchain

The advent of blockchain stems from the Bitcoin white paper released by Satoshi Nakamoto in 2008. Blockchain, as the underlying technology of Bitcoin, can establish trustworthy value transfer between any peers without relying on third-party trusted institutions. Since Bitcoin limits its usage in the field of digital finance, Ethereum was further introduced by Vitalik in 2013 to expand the applications based on smart contracts. Ethereum allows anyone to build and use *Decentralized Applications* (DApps) running on blockchain technology. The beneficial characteristics of decentralization, tamper resistance, and verifiability make Ethereum blockchain indispensable in application scenarios such as secure data sharing [13], [14] and *Internet of Things* (IoT) [15], [16].

Ethereum adopts an account-based data model to facilitate complex operations written by smart contract functions. In general, an Ethereum account represents only one owner, and a user can generate multiple accounts. The transaction process of the account includes transferring $Ether$ (i.e., Ethereum cryptocurrency) from one account to another, and deploying a new contract, etc. Based on the account-based data model, accounts in Ethereum can be divided into *external accounts* that are controlled by private keys, and *contract accounts* that are maintained by contract codes and codes associated with them.

Each Ethereum account has a pair of private key and public key that are encrypted for transactions creation and validation. A private key is first generated in creating an account, and the transaction can be initiated after the private key authorizes the external account. The external account address is calculated from the public key generated by the private key. The contract address is calculated from the contract creator's address and the number of transactions sent by the address. Therefore,

the key pair guarantees the anonymity and security of the account. Since accounts in Ethereum can create decentralized applications based on smart contracts, many new blockchain projects have been spawned, resulting in diverse account categories with more roles and more abnormal behaviors [17]. For the rest of the paper, unless otherwise stated, we use the terms 'account' and 'address' interchangeably.

B. Abnormal Behaviors Detection

There has been a lot of work concentrating on detecting the abnormal behaviors on blockchain that disclosures significant illicit finance risks and cybercrime. This necessities the detection and supervision of abnormal behaviors on blockchain to adequately mitigate those risks. Chen et al. [30] point out the leading illegal behaviors on the blockchain, including scams, money laundering, and market manipulation. Our work is related to several prior lines of work on the detection of these abnormal behaviors on the blockchain.

1) Scams: Scams on blockchain have been extensively studied [7], [8], [18], [19], [20], [31], [32], [33]. Among them, Ponzi scams account for a large proportion and attract much attention from researchers. Bartoletti et al. [7] discovered more than one hundred Ponzi scams based on smart contracts through analyzing publicly available source codes manually. It also adopts a Monte Carlo algorithm to calculate the normalized Levenshtein distance to enlarge their Ponzi scams dataset. Chen et al. [8] employed features extraction to train an XGBoost classifier that can detect hidden Ponzi smart contracts both from account and code features. Traditional text analysis methods [32], [33] cannot be directly applied for phishing detection on blockchain due to different propagation mediums. Chen et al. [18] adopted a cascaded feature extraction method to obtain transaction statistical features, and then iteratively train a LightGBM model under a double-sampling ensemble framework to uncover phishing addresses. Other work tried to detect scams such as *Initial Coin Offering* (ICO) [21], and honeypots [19], [20] with high accuracy.

2) Money Laundering: Due to the transaction records of Blockchain being transparent and publicly accessible, researchers try to utilize heuristics methods [34] to aggregate anonymous addresses into entities to de-anonymize blockchain accounts. Mixing service is proposed to improve the anonymity and privacy of accounts [35], [36]. However, some criminals utilize mixing services to do money laundering. Many analysis works focus on the discovery of mixing services. These works utilize heuristic methods based on the characteristics of money-laundering behavior, including multi-input heuristic [3], [4], change addresses heuristic [5], and behavior-based clustering [22]. Network analysis is also widely used in *anti-money laundering* technology (AML). For example, Weber et al. [23] proposed a *Graph Convolutional Network* (GCN) based detection model for anti-money laundering in Bitcoin. Wu et al. [6] adopted network motif to extract subgraph patterns that frequently appeared in Bitcoin and analyze the network features of the mixing service in the transaction graph. A recent study [24] employed network analysis to deeply explore the two mechanisms: exchange and obfuscation in the mixing service. However, these studies suffer from unlabeled imbalance data. Unlabeled data is common in the recommendation system [37].

3) Market Manipulation: Market manipulation is a common phenomenon in blockchain-based transactions. Several studies are performed to find driving factors in the fluctuation of cryptocurrency prices. Ciaian et al. [38] made the first attempt to analyze the formation of Bitcoin price. Since transaction information is time series, [25], [26] combined time series analysis methods such as vector space model with transaction data. Different from them, Chen et al. [27] introduced graph analysis into Bitcoin transaction data and found that Bitcoin price is closely related to some market manipulation patterns. The current situation of the token market is analyzed from the perspective of graph analysis by [28]. Besides that, researches [29] and [39] analyzed the off-chain data of the exchanges from the perspective of economics and drew an important conclusion: exchanges usually conduct some scams or arrange transaction robots to manipulate the price.

4) Arbitrage: In this section, we summarize the prior studies on arbitrage and present their limitations. Compared with these researches, advantages of our work are listed as below.

a) Limitations of related works: Before our work, there are already some researches that attempted to analyze the arbitrage behaviors on Ethereum. Flash boy 2.0 [11] proposed a game theory model that combines continuous time and partial information to formalize and study arbitrage behavior in DEXs. It recognizes arbitrage bots by detecting *Priority Gas Auctions* (PGAs) transactions on DEXs. Its main focus is the design flaws of the decentralized exchange protocol and the security threats to Ethereum caused by arbitrage. Zhou et al. [10] adopted graph cycle detection and subtree traversal methods for arbitrage detection. In their study, negative cycles in the transaction graph are considered as arbitrage behaviors. In summary, the application scenarios of these works focus on single-exchange arbitrage, and the features adopted only constitute the statistical features from a single dimension. Statistical features obtained from the heuristic analysis can not grab the hidden structural and transaction information.

b) Advantage over existing works: While our work is closely related to these prior studies [10], [11], it has major fundamental differences. First, our approach can explore the heterogeneous features of the arbitrage activities based on machine learning techniques, which outperform the heuristic methods that only consider some statistical features such as abnormal gas fees. Specifically, they fail to investigate the implicit information of edges in the transaction graph of Ethereum, and degrade the classification performance. Second, these works prefer to find potential arbitrage opportunities than analyzing the past arbitrage activities, which has little effect on regulating the Ethereum ecosystem. Third, we consider a broader application scenario of cross exchanges covering both DEXs and *Centralized Exchanges* (CEXs), while existing work mostly focuses on the arbitrage with a single exchange. To our best knowledge, our work is the first to apply machine learning techniques to accurately detect arbitrage addresses on

TABLE I
COMPARISON OF THE STATE-OF-THE-ART DETECTION METHODS

Category	Existing work	Specific behavior	Features	Detection method	Limitations
Scams	[7]	Ponzi schemes	Source code feature	Monte Carlo algorithm	Single feature, qualitative analysis
	[8]	Ponzi schemes	Statistical feature	XGboost	Single feature, lacks interpretability
	[18]	Phishing scams	Node & transaction features	LightGBM	Two fold features, imbalance data
	[19]	Honeypot	Bytecode feature	Symbolic execution	Single feature, requires prior knowledge
	[20]	Honeypot	Statistical feature	XGboost	Single feature
	[21]	ICO scams	Text feature	Heuristic analysis	Qualitative analysis
Money laundering	[3], [4], [22]	–	Multi-input heuristic, change addresses heuristic, and behavior-based clusterin	Heuristic analysis	Heuristic feature, qualitative analysis
	[6], [23], [24]	–	Structural feature	Graph analysis	Imbalance data
Market manipulation	[25], [26]	Bitcoin price	Time feature	Time series analysis	Qualitative analysis
	[27], [28], [29]	Exchanges	Network feature	Graph analysis	Qualitative analysis
Arbitrage	[10]	Single exchange arbitrage	Negative cycles	Graph search	Specific structure
	[11]	Single exchange arbitrage	PGAs	Graph search	Single feature
	Our work	Arbitrage	Heterogeneous features	Feature fusion, PU learning	Two fold features, broader application

Ethereum with the challenges of the heterogeneous features and absence of negative samples.

III. OVERVIEW OF ARBITRAGE DETECTION MODEL

In this section, we formalize the arbitrage addresses detection on Ethereum as a node classification problem on graphs. Moreover, we summarize key challenges we shall tackle and give an overview of our proposed method.

A. Problem Definition

Fig. 2(a) plots the raw transaction records. The triangles and circles denote the arbitrage addresses and normal addresses, respectively. The solid triangles are the addresses that have already been labeled as arbitrage addresses. The directed edge represents the coins transformation or contracts calling from one address to another address. Notice that for each pair of addresses, there may exist multiple edges. Our goal is to distinguish the hollow triangles from hollow circles, i.e., identify all the arbitrage addresses that have not been labeled.

The above problem can be formalized as a node classification problem on a directed attributed graph $G = (V, E)$, where the vertex set $V = \{v_1, \ldots, v_m\}$ represents the m transaction addresses, and the directed edge set $E = \{e_{ij} = (i, j) \mid v_i, v_j \in V\}$ represents the transaction records between addresses. Without loss of generality, we assume that the arbitrage addresses are positive class and the normal addresses are negative class. The first p addresses $P = \{v_1, \ldots, v_p\}$ are arbitrage addresses, i.e., $y_1 = \cdots = y_p = 1$. Then the arbitrage addresses detection is to classify the remaining vertexes $U = \{v_{p+1}, \ldots, v_m\}$, that is to predict $y_{p+1}, \ldots, y_m$.

B. Challenges

Our goal is to propose an arbitrage addresses detection algorithm on Ethereum, thus assisting the blockchain regulation. However, two main challenges are encountered while designing the detection algorithm: heterogeneous arbitrage features and the absence of negative samples.

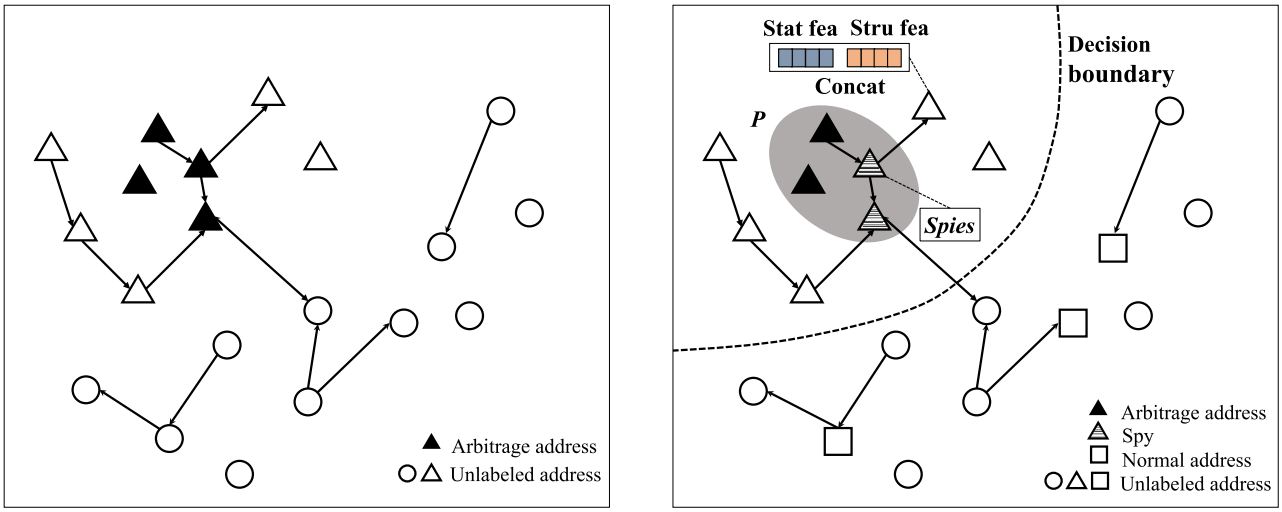
1) Heterogeneous Arbitrage Features: The statistical features of nodes (e.g., account balance) undoubtedly contain numerous dominant information to facilitate node classification. Meanwhile, edges between nodes and the whole transaction graph structure should not be ignored, because they usually contain implicit but useful information. It requires different methods to extract these heterogeneous features reflected by the nodes or edges. Therefore, how to define and extract the features of both the nodes and edges, and how to aggregate the graph structure into feature extraction are the first challenge in our detection algorithm.

2) Absence of Negative Samples: The labeled addresses in the dataset contain only a part of the positive samples, leaving the rest of the positive ones being unlabeled. In other words, the unlabeled addresses may also include the positive samples, and thus cannot be taken as the negative ones directly. Taking Fig. 2(a) as an example, only five arbitrage addresses (*black triangles*) are labeled, while the other nine arbitrage ones (*white triangles*) are unlabeled. It is very challenging to recognize the non-arbitrage ones from the unlabeled samples, leading to the absence of negative samples. Therefore, how to extract the negative samples, which together with the positive samples can facilitate the model training, turns out to be the second challenge.

C. Overview of the Proposed Method

To tackle the above two challenges, feature fusion and PU learning are adopted into the proposed detection method respectively. In this section, we briefly present our methods from the perspective of how to solve the challenges.

1) Heterogeneous Feature Extraction: From previous work, it can be concluded that a single dimension of features can not represent abnormal behaviors well, especially for arbitrage behavior. It is intuitive to understand that the attributes of accounts can reflect account categories to a certain extent, but the evolution of arbitrage seriously hinders the classification performance of statistical feature-based classifiers. To overcome the first challenge, we propose the definition of



(a) The original schematic diagram of transaction graph

(b) The schematic diagram after spy-based optimization

Fig. 2. The overview of the proposed method, where solid triangles denote arbitrage addresses, triangles with shadows are spies, hollow icons represent unlabeled addresses and squares denote normal addresses we selected after PU learning.

two-fold heterogeneous arbitrage features and feature fusion is introduced to achieve them. The heterogeneous features related to arbitrages are divided into two categories: statistical features and structural features.

For the statistical features, including the account features and time features, we conduct the extraction from the heuristic perspective based on expert knowledge. To be more specific, we draw three noteworthy findings of the arbitrage accounts and extract the transaction frequency plus the duration to reflect the time features. To extract the structural features, we incorporate the representation learning method, particularly the network embedding. The embedding representation of a node is constructed by aggregating the edge features, graph features, and neighbor features. After the feature extraction process, we obtain the statistical features and structural features respectively. Feature fusion concatenates these features to formulate the heterogeneous feature vector. Even though the statistical features of arbitrage behaviors evolve over time, heterogeneous features can obtain hidden structural features of nodes by aggregating the features of neighbor nodes, transactions, and the entire graph, which greatly compensates for the insufficiency of single-dimensional features.

2) Two-Step PU Learning Method: In this arbitrage detection case, we only have positive and huge amounts of unlabeled data. Unlabeled data can not be treated as negative samples directly because they may contain positive samples. Directly training the classifier with unlabeled samples as negative samples will lead to classification errors. PU learning is introduced into our method to deal with the second challenge: the absence of negative samples. PU learning can select the data which are most likely to be negative samples from unlabeled datasets to reduce noise to the greatest extent. Specifically, to find reliable negative samples, we adopt a two-step PU learning method, particularly the spy technique [12]. In short, we select some 'spies' from set P randomly, which are highlighted as red triangles in the picture. 'Spies' constitute spy instances dataset S. Then two new dataset P_r and U_r are achieved, namely $P_r = P \setminus S$, $U_r = U \cup S$, respectively. Spy technique locates the reliable negative samples by calculating a threshold θ. The 'spies' have a higher likelihood of being classified as a positive class compared to normal nodes. Thus the samples with the probability that less than the threshold θ are identified as negative samples. Heterogeneous feature fusion and PU learning will be introduced detailed in Section IV and V respectively.

IV. Heterogeneous Feature Fusion

Through the preliminary analysis, we conclude that arbitrage behavior's features consist of two categories: statistical features and structural features. Statistical features are defined as the inherent characteristics of arbitrage behavior itself, while structural features represent arbitrage nodes' interaction with others in the transaction graph. In terms of the data formats, statistical features and structural features are heterogeneous, which requires totally different methods to extract. In this section, we detail how these heterogeneous features are extracted and further fused. A fusion of various features contributes to a better classification model.

A. Statistical Features by Expert Knowledge

Generally speaking, we extract statistical features of the arbitrage addresses denoted as $\boldsymbol{x}$ by expert knowledge. Since statistical features can be divided into two parts, namely account features and time features, we present the feature extraction methods for these two parts respectively. The definitions of various statistical features are summarized in Table II.

1) Account Features: Arbitrage, as a financial behavior, should be closely related to the accounts instinctively. Therefore, account features are introduced to describe these characteristics of addresses. We perform a statistical analysis of labeled arbitrage addresses and unlabeled addresses. After that,

TABLE II
DEFINITION OF STATISTICAL FEATURES

Feature	Notation	Definition
Account	F_1	Balance of each account
	F_2	Degree of input transactions
	F_3	Degree of output transactions
	F_4	Weights of in-degree
	F_5	Weights of out-degree
	F_6	Amount ratio of in-degree to out-degree
Time	F_7	Transaction frequency
	F_8	Transaction duration

TABLE III
COMPARISON OF LABELED AND UNLABELED DATA

		F_1	F_2	F_3	F_4	F_5	F_6
P	Mean	0.0021	1.80	2.11	1.32	1.33	1.00
	Std	0.32	2.78	2.34	4.30	3.98	0.54
	Median	0.00	2.00	2.00	0.54	0.60	1.00
U	Mean	1.12	1.28	1.30	5.43	6.21	49.50
	Std	5.30	10.21	8.53	14.23	18.14	55.19
	Median	0.00	1.00	1.00	0.13	0.11	1.00

we calculate some statistical indexes, such as mean, standard deviation, and median parameter. The calculation results for different account features are listed in Table III. From the table, we can discover some noteworthy findings of arbitrage addresses.

- **Finding 1**: The average account balance of an arbitrage address is tiny, serving as an instinctive feature of arbitrage. The reason is that when an arbitrage opportunity occurs, most arbitrage accounts will transfer all their balances out in order to maximize profits.
- **Finding 2**: Compared to the unlabeled addresses, the labeled arbitrage addresses have a smaller standard deviation. It indicates that the account characteristics of these arbitrage addresses stay in a stable manner. Therefore, the account feature vector can be extracted to represent them.
- **Finding 3**: According to F_6, the input amount and output amount of most arbitrage addresses are the same. For unlabeled addresses, a large standard deviation means that different types of accounts share a wide variation, while a median of 1.00 indicates the input and output amounts of most addresses are roughly the same. The entire Ethereum transaction is indeed affected by a small number of addresses with massive transaction volumes.

2) Time Features: Normally, the opportunity for arbitrage is fleeting. Therefore, from a heuristic point of view, most arbitrage transactions exhibit short-term and high-frequency characteristics. To better display this feature of the account, we introduce the time features consisting of transaction frequency and transaction duration. We analyze arbitrage addresses and non-arbitrage addresses based on these two indexes, and fuse them into a feature vector.

B. Structural Features by Representation Learning

Statistical features can only depict nodes' features, while edges in the transaction graph are unignorable. Edges in the Ethereum transaction graph contain transaction volume and timestamp, which are potentially critical to detect arbitrage. To aggregate edge information and recessive graph structure into a feature vector, we introduce network embedding to aggregate edge features into structural features. In our work, we utilize GraphSAGE [40] to aggregate edge features $\boldsymbol{z}_{ij}$. Different neighbor nodes have different importance in node classification. In the Ethereum trading graph, edges with the greater transaction volume or closer transaction timestamp have a greater impact. Thus we introduce the embedding representation of edges, each edge aggregates two adjacent nodes' features, its own features, and the entire graph feature. Then each node aggregates corresponding edges' features to achieve the structural features.

Neighbor nodes sampling: For the Ethereum transaction network $G = (V, E)$, for each vertex $v_i \in V$, we endow it with an initial characteristic representation. In our work, the initial feature vector is set as statistical features $\boldsymbol{h}_i^{(0)} = \boldsymbol{x}_i$. The embedding of node v_i is generated by sampling neighbor nodes and aggregating their information and edge features. The aggregation method is as follows:

$$\boldsymbol{h}_{(i,j)}^{(k)} = \text{Update}_{\text{edge}}\left(\boldsymbol{h}_{(i,j)}^{(k-1)}, \boldsymbol{h}_i^{(k-1)}, \boldsymbol{h}_j^{(k-1)}, \boldsymbol{h}_G^{(k-1)}\right). \quad (1)$$

$$\boldsymbol{m}_{\mathcal{N}(i)} = \text{Aggregate}_{\text{node}}\left(\left\{\boldsymbol{h}_{(i,j)}^{(k)}, \forall j \in \mathcal{N}(i)\right\}\right). \quad (2)$$

$$\boldsymbol{h}_i^{(k)} = \text{Update}_{\text{node}}\left(\boldsymbol{h}_i^{(k-1)}, \boldsymbol{m}_{\mathcal{N}(i)}, \boldsymbol{h}_G^{(k-1)}\right). \quad (3)$$

$$\boldsymbol{h}_G^{(k)} = \text{Update}_{\text{graph}}\left(\boldsymbol{h}_G^{(k-1)}, \left\{\boldsymbol{h}_i^{(k)}\right\}, \left\{\boldsymbol{h}_{(i,j)}^{(k)}\right\}\right). \quad (4)$$

As can be seen from equations above, the node and edge representation of each layer depend on the embedding vector in the previous layer. The whole aggregation method can be further divided into three parts: node feature aggregation, edge feature aggregation, and graph feature aggregation. We detail each process according to the four equations respectively.

Eqn. (1) describes the updating method of the edge features. The embedding representation of edge e_{ij} in layer k aggregates the feature of node v_i (denoted as $\boldsymbol{h}_i^{(k-1)}$), the feature of node v_j (denoted as $\boldsymbol{h}_j^{(k-1)}$), its own feature in layer $k-1$ (denoted as $\boldsymbol{h}_{(i,j)}^{(k-1)}$), and the whole graph structure (denoted as $\boldsymbol{h}_G^{(k-1)}$). Thus the edge features contains the information of two adjacent nodes and the whole graph.

Eqn. (2) and Eqn. (3) describe the process of node features aggregation. Eqn. (2) represents aggregation of edge features which starts from node v_i. According to Eqn. (3), the feature of node v_i updates by its own feature $\boldsymbol{h}_i^{(k-1)}$, edge features $\boldsymbol{m}_{\mathcal{N}(i)}$, and the graph structure feature $\boldsymbol{h}_G^{(k-1)}$.

Finally, we update the feature of the graph according to Eqn. (4). By aggregating the graph feature in layer $k-1$ (denoted as $\boldsymbol{h}_G^{(k-1)}$), node feature of node v_i (denoted as $\boldsymbol{h}_i^{(k)}$), and edge feature of e_{ij} (denoted as $\boldsymbol{h}_{(i,j)}^{(k)}$), the whole graph feature is updated.

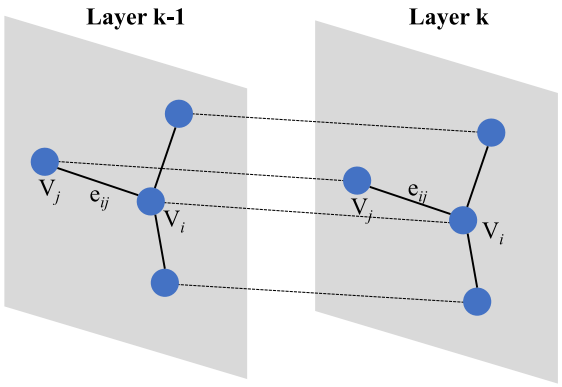


Fig. 3. The process of aggregation.

Aggregator choosing: the goal of this operation is to aggregate all the information contained in the embedding vector of the neighbor nodes into the embedding representation of the target node through the aggregation function. Aggregate functions with better effects include pooling and LSTM. However, LSTM is not symmetric, which means that it is not permutation invariant. Thus, we employ a pooling aggregator.

The pooling aggregator first performs a nonlinear transformation on the embedding vector of the neighbor vertices, and then performs a pooling operation. After that, we concatenate the result with the representation vector of the target node v_i, and finally passes through a nonlinear transformation to get the k-th level node's feature embedding. After the neighbor nodes of the $k-1$ layer of v_i are aggregated, the vector is concatenated with the embedding representation of v_i at the $k-1$ layer and sent to the nonlinear transformation function to obtain the final embedding representation, namely structural features. After we obtain structural features, we execute feature fusion $\tilde{\boldsymbol{x}}_i = \text{Concat}(\boldsymbol{x}_i, \boldsymbol{h}_i^{(k)})$ to acquire heterogeneous feature vector of node v_i by concatenating $\boldsymbol{x}_i$ and $\boldsymbol{h}_i^{(k)}$.

V. Reliable Negative Instances Selection

In the task of arbitrage address detection, arbitrage addresses can be identified as positive instances while unlabeled addresses cannot be considered directly as negative samples. Negative instances are missing so that standard classification algorithms cannot be directly applied. In this work, we utilize the classic spy technique to achieve reliable negative instances.

A. Negative Instances Selection

the goal aims to acquire reliable negative instances set RN, and it contains three procedures.

- **Step 1:** First, we use the spy technique to randomly select a set denoted as S out of labeled positive samples set P at a sampling rate of 15% to act as 'spies'. The sampled spy instances set S and unlabeled instances set U are combined together and first classified as negative instances, namely for each v_i in dataset $U \cup S$, its label is set to be negative. After sampling procession, we acquire two new datasets which $P_r = P \setminus S$ and $U_r = U \cup S$.
- **Step 2:** Then, P_r and U_r are used to train the classifier f. The classifier f outputs the probability of the instance v_i in the dataset belonging to a positive class. Because the spy instances in dataset U_r achieve a higher possibility to be classified as positive instances, we rank the probability values in the dataset U_r in ascending order. It is not difficult to imagine that the probability values similar to the spy instances are the potential positive instances, and the probability values much smaller than the spy instances are the reliable negative instances.
- **Step 3:** While according to threshold θ, the original unlabeled instances set U is predicted to determine reliable negative instances. The threshold θ divides the U_r into two parts. If the probability of an instance $f(\tilde{\boldsymbol{x}}_i)$ is less than the threshold, it belongs to the reliable negative set RN. In our work, the threshold $\theta = 0.15$ is selected. When the predicted probability of an unlabeled address is greater than 0.15, we consider it to be a reliable negative instance.

B. Classifier Training

this stage is utilizing the positive instances and the reliable negative instances selected in stage one to train the standard supervised model and employ it to further predict unlabeled instances. In our work, for efficiency considerations, logistic regression is adopted as the classifier. In order to avoid the imbalance between the positive instance and the predicted reliable negative instance, we adopt a cost-sensitive strategy to minimize the following objective function by setting the penalty weight in the loss function:

$$C_+ \sum_{y_i=1} l\left(y_i, f\left(x_i\right)\right) + C_- \sum_{y_i=-1} l\left(y_i, f\left(x_i\right)\right) + R(w). \quad (5)$$

In Eqn. (5), C_+ and C_- represent penalty weight parameters. λ represents regularization co-efficiency while $R(w)$ means regularization term, which is set as ℓ_2-norm. $l\left(y_i, f\left(x_i\right)\right)$ denotes loss term, and in our work the loss term is log-loss.

In the experiment, we set the values of C_+ and C_- as the inverse ratio of P and RN to increase the probability of the classification algorithm classifying the instances as positive.

VI. Arbitrage Addresses Detection

After the procedures elaborated above, we obtain each node's heterogeneous feature vector $\boldsymbol{b}$ and sufficient negative samples. Then, we can apply standard classification algorithms to solve this problem. In this paper, XGBoost is adopted as our classification model. XGBoost is a widely used machine learning algorithm and it has been proved suitable for many scenarios. In the following part of this section, we will briefly introduce how to apply XGBoost for arbitrage addresses detection.

In essence, XGBoost aims to minimize the following objective function:

$$Obj(\theta) = L(\theta) + \Omega(\theta). \quad (6)$$

Algorithm 1 Arbitrage Addresses Detection Algorithm

Input: graph $G(V, E)$;

1: Initialize node feature $\boldsymbol{h}_i^{(0)} = \boldsymbol{x}_i$, edge feature $\boldsymbol{z}_{ij} = \boldsymbol{h}_{(i,j)}^{(0)}$, weight matrix $\mathbf{W}$, positive instance dataset P, unlabeled instance dataset U, reliable negative instance dataset $RN = \emptyset$, layer number $K = 2$, sampling ratio $r = 0.15$, threshold $\theta = 0.15$;
2: **for** $k = 1, \ldots, K$ **do**
3: **for** $v_i \in V$ **do**
4: $\boldsymbol{h}_{(i,j)}^{(k)} = \text{Update}_{\text{edge}}\left(\boldsymbol{h}_{(i,j)}^{(k-1)}, \boldsymbol{h}_i^{(k-1)}, \boldsymbol{h}_j^{(k-1)}, \boldsymbol{h}_G^{(k-1)}\right)$;
5: $\boldsymbol{m}_{\mathcal{N}(i)} = \text{Aggregate}_{\text{node}}\left(\left\{\boldsymbol{h}_{(i,j)}^{(k)}, \forall j \in \mathcal{N}(i)\right\}\right)$;
6: $\boldsymbol{h}_i^{(k)} = \text{Update}_{\text{node}}\left(\boldsymbol{h}_i^{(k-1)}, \boldsymbol{m}_{\mathcal{N}(i)}, \boldsymbol{h}_G^{(k-1)}\right)$;
7: $\boldsymbol{h}_G^{(k)} = \text{Update}_{\text{graph}}\left(\boldsymbol{h}_G^{(k-1)}, \left\{\boldsymbol{h}_i^{(k)}\right\}, \left\{\boldsymbol{h}_{(i,j)}^{(k)}\right\}\right)$;
8: **end for**
9: $\boldsymbol{h}_i^{(k)} = \boldsymbol{h}_i^{(k)} / \|\boldsymbol{h}_i^{(k)}\|_2$
10: **end for**
11: **for** $v_i \in V$ **do**
12: $\tilde{\boldsymbol{x}}_i = \text{Concat}(\boldsymbol{x}_i, \boldsymbol{h}_i^{(k)})$;
13: **end for**
14: Sample spy set S with $|S| = |P| \cdot r$;
15: $P_r = P \setminus S$, $U_r = U \cup S$;
16: Set the label of instance in P_r as 1 and in U_r as -1;
17: $f = \text{Train}(P_r, U_r)$;
18: **for** each $v_i \in U$ **do**
19: **if** $f(\tilde{\boldsymbol{x}}_i) \leq \theta$ **then**
20: $RN = RN \cup \{v_i\}$;
21: **end if**
22: **end for**
23: $\text{XGBoost}(RN \cup P, \tilde{\mathbf{X}})$
24: **return** arbitrage addresses;

Output: Arbitrage addresses.

In Eqn. (6), $L(\theta)$ represents the training loss function and $\Omega(\theta)$ refers to the regularization term. In our work, we train a logistic regression binary classification model which predicts the probability of addresses belonging to arbitrage. Those nodes with an output probability larger than 0.5 are conceived as potential positive instances. Thus applying logistic regression into loss function, we achieve the loss function as Eqn. (7):

$$L(\theta) = \sum_i \left[y_i \ln\left(1 + e^{-\hat{y}_i}\right) + (1 - y_i) \ln\left(1 + e^{\hat{y}_i}\right)\right]. \quad (7)$$

The training loss function measures how predictive the model is on the training data and the regulation term penalizes the complexity of the model, which helps to avoid overfitting.

The core idea of the XGBoost algorithm is to continuously add trees and perform feature splitting to grow a tree. Each time a tree is added, a new function is actually learned to fit the residuals of the last prediction. In our case, XGBoost adopts L additive functions summing up to predict the output $\hat{y}$, predicted value of the tree groups is as follow:

$$\hat{y} = \phi(x_i) = \sum_{t=1}^{L} f_t(x_i).$$

In the process of model learning, *additive training* is introduced in XGBoost. *Additive training* means the prediction model starts with a constant prediction $\hat{y}_i^{(0)} = 0$. At each iteration, a new function (namely a new tree) $f_t(x_i)$ is added to the existing $\hat{y}_i^{(t-1)}$ to fit the residuals between the predicted results of the previous training round and the true value. The process of *additive training* is as follows:

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\ldots \\
\hat{y}_i^{(t)} &= \sum_{l=1}^{t} f_l(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}$$

In XGBoost, t regression trees are established to make the predicted value of the tree groups close to the real value as much as possible. Thus, the classification issue is transformed into a problem of minimizing the objective function by combining the above equations:

$$Obj^{(t)} = \sum_{i=1}^{n} l\left[y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right] + \Omega(f_t) + \text{cons}.. \quad (8)$$

where $\Omega(f_t)$ represents regularization-term, and it means summing up the complexity of all t regression trees to prevent over fitting of the model. Specifically, the complexity consists of leaf number T and leaf weight w, thus the formula of $\Omega(f_t)$ can be further expressed as:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2.$$

In Eqn. (8), we have to seek the optimum f_t to minimize the objective function. Besides that, to achieve the best performance, post-pruning is utilized to trim unnecessary leaf nodes. The pseudocode for the proposed detection method is listed as Algorithm 1. First, we extract the structural features through a K-layer neural network, as shown by lines 2-10. After obtaining structural features, we concatenate statistical features with them from line 11 to line 13. Finally, PU learning method is described by lines 14-22.

VII. Experiments

In this section, we first briefly describe our experimental settings. After that, several groups of experiments are performed to demonstrate the feasibility and effectiveness of our detection model.

A. Experimental Settings

1) Machine Configuration: We conduct the experiments on a machine with the 8-core Intel Core i9-9900K CPU@3.6GHz, 64GB RAM, and 1TB HDD.

2) Dataset Description: The Ethereum transactions are recorded in the underlying Blockchain and can be retrieved from Ethereum clients. For the sake of arbitrage detection model establishment, we need to collect address datasets with the roles of arbitragers. Bloxy[1] is a website that provides some

[1]https://bloxy.info

real-time data analysis on blockchain. From Bloxy, we collect the dataset containing the labeled addresses belonging to arbitrage addresses and clean these data as ground truth. We use the dataset provided by the economics field papers and related news reports to verify the accuracy of the dataset provided by Bloxy. After the process of data cleaning, we obtain $1,280$ labeled arbitrage addresses and $63,038$ transaction records related to arbitrage. In general, these labeled addresses including both the DEXs (e.g., UniSwap[2] and SushiSwap[3]) and CEXs (e.g., Binance[4] and Huobi[5]). We argue that our collected dataset is reasonable and representative, since the exchanges labeled by us exactly include the top exchanges, such as UniSwap and Binance, according to the statistics on Coinmarketcap.[6]

3) Evaluation Metrics: In order to have a quantitative evaluation index for the accuracy of the proposed model, the following three evaluation metrics were introduced, and the definition of each evaluation metric is as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}$$
$$\text{F-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

where TP and FP are short for true positive and false positive, respectively.

4) Baseline Methods: Referring to the above sections, we propose an arbitrage detection algorithm combing with PU learning and heterogeneous feature fusion. According to the detection algorithm, we further divide the experimental comparison into three parts, namely PU learning, embedding, and classification. In each part, our proposed method is compared with several classic approaches. In the following, we give a brief description of baseline methods in each part.

First, we make a comparison between PU learning and random selection to demonstrate the superiority of PU learning. Furthermore, we evaluate different methods (i.e., listed in Table V) to select 'negative samples', which explains why the spy technique is picked in our work. In PU learning parts, 1-DNF and Rocchio are selected as baseline methods.

- 1-DNF: For each feature if its occurrence frequency in the P set is greater than that in the N set, the feature is recorded as a positive feature, and all the features form the set PF.
- Rocchio: Determine the category of samples by calculating cosine similarity.

Second, to evaluate the importance of feature aggregation, we conduct comparative experiments with different combinations of feature vectors. In our proposed method, the feature vector is composed of 'Structural + Statistical' feature vector.

In the classification part, we adopt XGBoost and take Naive Bayes as the baseline method. Finally, we illustrate the effects brought by the embedding dimension of the structure feature.

[2]https://uniswap.org
[3]https://www.sushi.com
[4]https://www.binance.com
[5]https://www.huobi.com
[6]https://coinmarketcap.com

TABLE IV
COMPARISON OF DIFFERENT METHODS

PUL	Embedding	Classifier	Pre	Rec	F1
W/ PUL	W/E	XGBoost	**0.90**	**0.88**	**0.89**
	W/E	Naive Bayes	0.83	0.81	0.82
	W/o E	XGBoost	0.87	0.86	0.86
	W/o E	Naive Bayes	0.80	0.76	0.80
W/o PUL	W/E	XGBoost	0.70	0.72	0.71
	W/E	Naive Bayes	0.66	0.64	0.65
	W/o E	XGBoost	0.68	0.66	0.67
	W/o E	Naive Bayes	0.60	0.61	0.60

TABLE V
COMPARISON OF DIFFERENT PU LEARNING METHODS

Method	Precision	Recall	F-score
1-DNF	0.71	0.72	0.71
Spy	0.90	0.88	0.89
Rocchio	0.87	0.86	0.86

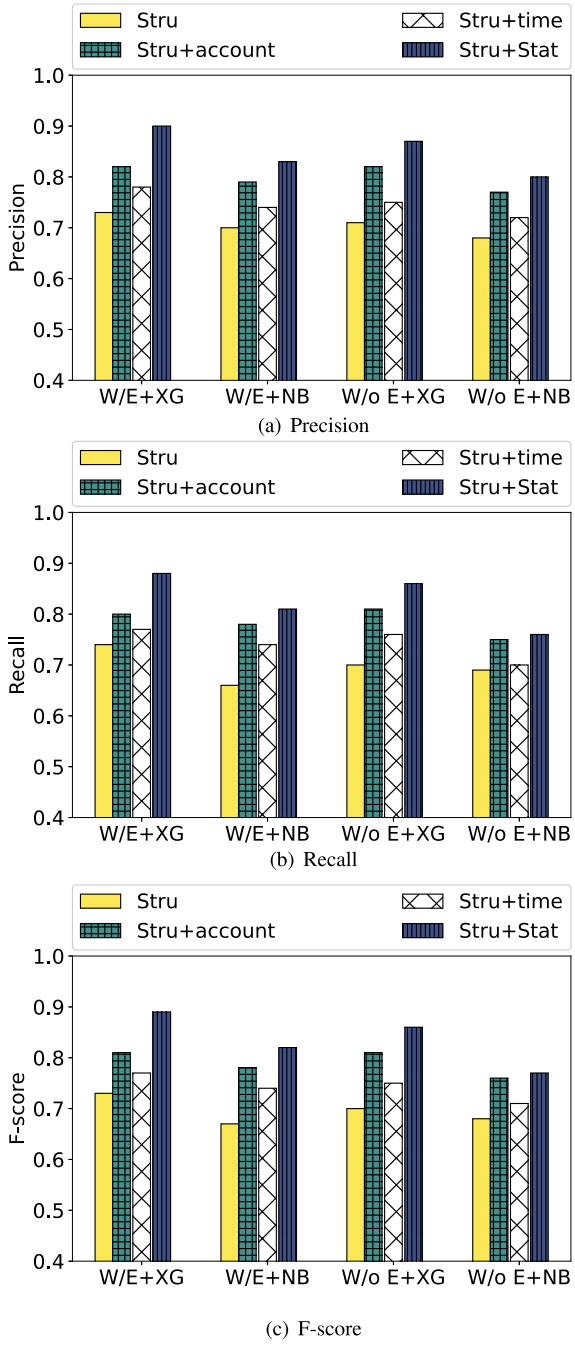
For the experiments which need the negative samples as input, we take the results obtained by the PU learning method. In the PU learning stage, the training dataset includes 70% labeled addresses, while the rest unlabeled addresses are selected randomly. Then, we find $6,547$ negative instances. These negative instances and labeled addresses are used to form the dataset D. In the downstream classification model, 70% of dataset D is sampled to be training set while the rest 30% of D is test set. Metrics, including precision, recall, and F-score, are utilized to reflect the accuracy of the proposed model.

B. Classification Performance

In this section, we detail each group of experimental results and gain some insights through an in-depth analysis of the experimental results.

1) PU Learning Influence: As can be seen from Table IV, the dataset obtained by PU learning outperforms the randomly sampled dataset, in terms of classification performance. To be more specific, in the group of 'W/ PUL', although the accuracy varies from one method to another, all of the four methods deliver good classification performance, whose precisions are above 80%. On the contrary, in the group of 'W/o PUL', the highest precision of the detection model does not exceed 70%. The reason is that the negative instances in the 'W/o PUL' dataset are not 'reliable'. In the 'W/o PUL' dataset, we select some samples from the unlabeled dataset U as negative instances randomly for binary classifier training. Some random samples may be positive instances, which bring great noise interference to the classification model.

Besides that, to demonstrate the superiority of the spy technique, we adopt Rocchio and 1-DNF as comparison methods in the experiment, respectively. In Rocchio, we determine the set of negative samples according to the cosine similarity

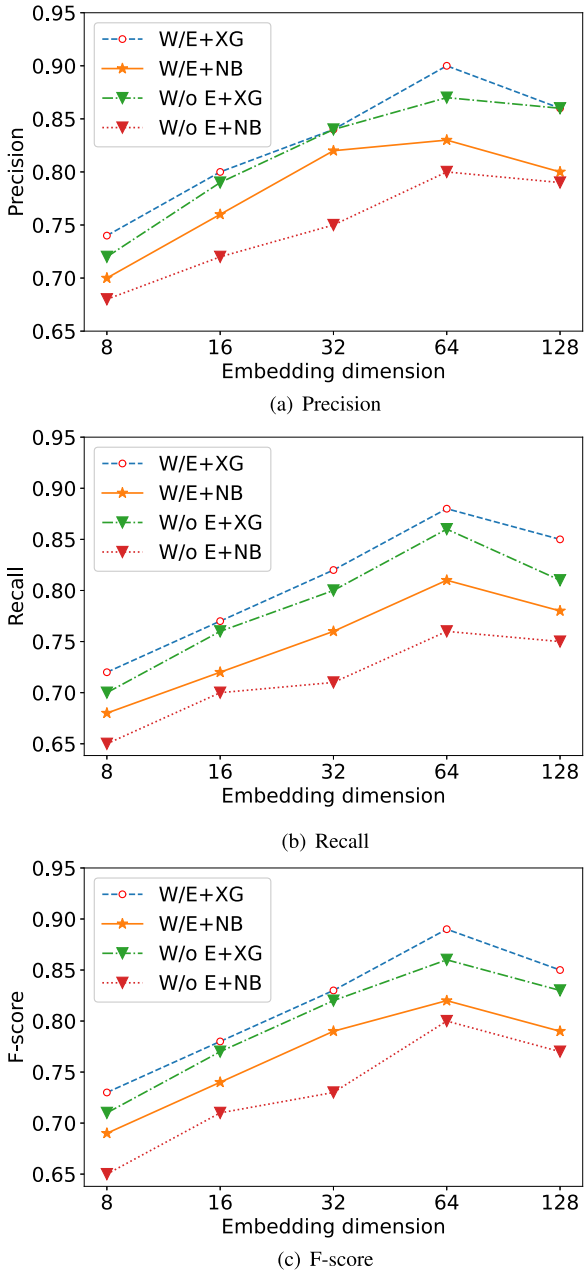


(a) Precision

(b) Recall

(c) F-score

Fig. 4. Comparisons of various methods with different feature vectors.



(a) Precision

(b) Recall

(c) F-score

Fig. 5. Comparisons of various methods with different embedding dimensions.

between node vectors. The cosine similarity of a node is calculated based on its distance from the positive samples. The 1-DNF method compares positive samples and unlabeled samples, and finds the salient features set PF in the dataset of positive samples P. As long as the features in the unlabeled samples do not appear in the PF, it is considered a reliable negative instance. As depicted in Table V, the spy technique can achieve precision performance up to 0.90.

Observation 1: PU learning can obtain reliable negative samples from unlabeled samples to improve classification performance and the spy technique has higher superiority over other methods.

2) Features Vector Influence: To show the impact of features vector on the detection precision and rationality in the fusion process, we choose four groups with the best performance from the eight sets of comparison experiments mentioned above, namely 'W/E'+XGBoost, 'W/E'+Naive Bayes, 'W/o E'+XGBoost, and 'W/o E'+Naive Bayes. We change the composition of feature vectors to judge the impact of our proposed feature fusion method on the classification results. Four sets of comparison experiments are conducted to demonstrate the generality of the effect of feature vectors, whose results are shown in Fig. 4. Under the premise that the embedding dimension is 64, we can observe that both the '*Stru* + time' and '*Stru* + account' feature vector perform better than

only the '*Stru*' vector, and the '*Stru* + account' feature vector plays a more important role in arbitrage detection. It can be summarized that the classification results which incorporate the '*Stat*' features are significantly better than the algorithms that ignore them. This result demonstrates that although the network embedding algorithm can aggregate the structural and corresponding edge features of the node in the transaction network well, it can only represent one single dimension feature. Although the arbitrage behavior itself is a financial behavior, its features are not only reflected in the transaction network, but also in their inherent financial attributes. This verifies the correctness of our statistical features selection strategy.

Observation 2: Combining statistical and structural features can achieve the best performance, which demonstrates the effectiveness of our proposed heterogeneous feature fusion.

3) Embedding Dimension Influence: To study the influence of the different dimensions of the feature vector extracted in embedding, we evaluate the detection model for different dimensions in terms of three evaluation matrices. As can be seen from Fig. 5, when the embedding dimension increases, the detection model has greater performance. Particularly, the best classification performance exceeds 90%, when the embedding dimension equals 64. Besides, as the embedding dimension increases beyond 64, although the classifier is still at a high level, it experiences a slight decline. The reason is that a 64-dimension feature vector is enough to represent the feature of arbitrage addresses, and higher-dimension vectors will not bring more information but consume more computing resources.

Note that, although the classification accuracy grows as the dimension parameter increases, the model's classification accuracy is still within an acceptable range even when the dimension is set as a small value. To be more specific, the accuracy is higher than 65%, even though the dimension is only 8. The reason for this phenomenon may be that the statistical features compensate for the lack of structural features. It further reflects the improvement of the classification effect of our proposed feature fusion method from the side.

Observation 3: The 64-dimension embedding is sufficient to represent node characteristics, while a too-high dimension consumes many resources without improving performance.

VIII. Conclusion

The arbitrage activities on Ethereum cause enormous economic losses to innocent users and aggravate the network congestion, which is quite necessary to be regulated. The major challenges of arbitrage activities detection on Ethereum include the heterogeneous features and lack of negative instances. We propose a novel arbitrage detection approach by machine learning techniques. Specifically, we fuse statistical features and structural features of Ethereum nodes and incorporate PU learning, network embedding, and node classification into the detection process. The experimental results demonstrate that our extracted features have dramatically improved the classifier performance. It also shows the superiority of the arbitrage address detection model compared with the baseline methods.

References

[1] H. Jin and J. Xiao, "Towards trustworthy blockchain systems in the era of 'Internet of value': Development, challenges, and future trends," *Sci. China Inf. Sci.*, vol. 65, no. 5, pp. 1–11, May 2022.

[2] M. Conti, S. Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3416–3452, 4th Quart., 2018.

[3] M. Spagnuolo, F. Maggi, and S. Zanero, "Bitiodine: Extracting intelligence from the bitcoin network," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2014, pp. 457–468.

[4] M. Ober, S. Katzenbeisser, and K. Hamacher, "Structure and anonymity of the bitcoin transaction graph," *Future Internet*, vol. 5, no. 2, pp. 237–250, 2013.

[5] T. Neudecker and H. Hartenstein, "Could network information facilitate address clustering in bitcoin?" in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2017, pp. 155–169.

[6] J. Wu, J. Liu, W. Chen, H. Huang, Z. Zheng, and Y. Zhang, "Detecting mixing services via mining bitcoin transaction network with hybrid motifs," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 52, no. 4, pp. 2237–2249, Apr. 2022.

[7] M. Bartoletti, S. Carta, T. Cimoli, and R. Saia, "Dissecting Ponzi schemes on Ethereum: Identification, analysis, and impact," 2017, *arXiv:1703.03779*.

[8] W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng, and Y. Zhou, "Detecting Ponzi schemes on Ethereum: Towards healthier blockchain technology," in *Proc. World Wide Web Conf. World Wide Web (WWW)*, 2018, pp. 1409–1418.

[9] I. Makarov and A. Schoar, "Trading and arbitrage in cryptocurrency markets," *J. Financial Econ.*, vol. 135, no. 2, pp. 293–319, Feb. 2020.

[10] L. Zhou, K. Qin, A. Cully, B. Livshits, and A. Gervais, "On the just-in-time discovery of profit-generating transactions in DeFi protocols," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2021, pp. 919–936.

[11] P. Daian et al., "Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 910–927.

[12] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu, "Building text classifiers using positive and unlabeled examples," in *Proc. 3rd IEEE Int. Conf. Data Mining*, Nov. 2003, pp. 179–186.

[13] G. S. Aujla and A. Jindal, "A decoupled blockchain approach for edge-envisioned IoT-based healthcare monitoring," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 2, pp. 491–499, Feb. 2021.

[14] M. Shen et al., "Blockchain-assisted secure device authentication for cross-domain industrial IoT," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 5, pp. 942–954, May 2020.

[15] M. Shen, J. Duan, L. Zhu, J. Zhang, X. Du, and M. Guizani, "Blockchain-based incentives for secure and collaborative data sharing in multiple clouds," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1229–1241, Jun. 2020.

[16] Y. Liu, X. Guan, Y. Peng, H. Chen, T. Ohtsuki, and Z. Han, "Blockchain based task offloading for edge computing on low-quality data via distributed learning in the internet of energy," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 2, pp. 657–675, Feb. 2022.

[17] C. Wang et al., "Demystifying Ethereum account diversity: Observations, models and analysis," *Frontiers Comput. Sci.*, vol. 16, no. 4, pp. 1–12, Aug. 2022.

[18] W. Chen, X. Guo, Z. Chen, Z. Zheng, and Y. Lu, "Phishing scam detection on Ethereum: Towards financial security for blockchain ecosystem," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 4506–4512.

[19] C. F. Torres, M. Steichen, and R. State, "The art of the scam: Demystifying honeypots in Ethereum smart contracts," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 1591–1607.

[20] R. Camino, C. F. Torres, M. Baden, and R. State, "A data science approach for honeypot detection in Ethereum," 2019, *arXiv:1910.01449*.

[21] G. Fenu, L. Marchesi, M. Marchesi, and R. Tonelli, "The ICO phenomenon and its relationships with Ethereum smart contract environment," in *Proc. Int. Workshop Blockchain Oriented Softw. Eng. (IWBOSE)*, Mar. 2018, pp. 26–32.

[22] D. Ron and A. Shamir, "How did dread pirate Roberts acquire and protect his bitcoin wealth?" in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2014, pp. 3–15.

[23] M. Weber et al., "Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics," 2019, *arXiv:1908.02591*.

[24] L. Wu et al., "Towards understanding and demystifying bitcoin mixing services," in *Proc. Web Conf.*, Apr. 2021, pp. 33–44.

[25] L. Kristoufek, "What are the main drivers of the bitcoin price? Evidence from wavelet coherence analysis," *PLoS ONE*, vol. 10, no. 4, Apr. 2015, Art. no. e0123923.
[26] I. Georgoula, D. Pournarakis, C. Bilanakos, D. Sotiropoulos, and G. M. Giaglis, "Using time-series and sentiment analysis to detect the determinants of bitcoin prices," in *Proc. 9th Medit. Conf. Inf. Syst.*, 2015, p. 20.
[27] W. Chen, J. Wu, Z. Zheng, C. Chen, and Y. Zhou, "Market manipulation of bitcoin: Evidence from mining the Mt. Gox transaction network," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2019, pp. 964–972.
[28] W. Chen, T. Zhang, Z. Chen, Z. Zheng, and Y. Lu, "Traveling the token world: A graph analysis of Ethereum ERC20 token ecosystem," in *Proc. Web Conf.*, Apr. 2020, pp. 1411–1421.
[29] Q. Ji, E. Bouri, L. Kristoufek, and B. Lucey, "Realised volatility connectedness among bitcoin exchange markets," *Finance Res. Lett.*, vol. 38, Jan. 2021, Art. no. 101391.
[30] W. Chen and Z. Zheng, "Blockchain data analysis: A review of status, trends and challenges," *J. Comput. Res. Dev.*, vol. 55, no. 9, pp. 1853–1870, 2018.
[31] M. Vasek and T. Moore, "There's no free lunch, even using bitcoin: Tracking the popularity and profits of virtual currency scams," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2015, pp. 44–61.
[32] N. Abdelhamid, A. Ayesh, and F. Thabtah, "Phishing detection based associative classification data mining," *Expert Syst. Appl.*, vol. 41, no. 13, pp. 5948–5959, Oct. 2014.
[33] E. Medvet, E. Kirda, and C. Kruegel, "Visual-similarity-based phishing detection," in *Proc. 4th Int. Conf. Secur. Privacy Commun. Netowrks (SecureComm)*, 2008, pp. 1–6.
[34] M. Harrigan and C. Fretter, "The unreasonable effectiveness of address clustering," in *Proc. Int. IEEE Conf. Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People, Smart World Congr. (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, Jul. 2016, pp. 368–373.
[35] G. Maxwell. (Aug. 22, 2013). CoinJoin: Bitcoin Privacy for the Real World. Post on Bitcoin Forum. [Online]. Available: https://bitcointalk.org/index.php?topic=279249.0
[36] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "Coinshuffle: Practical decentralized coin mixing for bitcoin," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2014, pp. 345–364.
[37] Y. Zhang, Y. Li, R. Wang, M. S. Hossain, and H. Lu, "Multi-aspect aware session-based recommendation for intelligent transportation services," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4696–4705, Jul. 2020.
[38] M. Rajcaniova, P. Ciaian, and d'A. Kancs, "The economics of BitCoin price formation," *Appl. Econ.*, vol. 48, no. 19, pp. 1799–1815, 2016.
[39] P. Xia et al., "Characterizing cryptocurrency exchange scams," *Comput. Secur.*, vol. 98, Nov. 2020, Art. no. 101993.
[40] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.

Hai Jin (Fellow, IEEE) received the Ph.D. degree in computer engineering from the Huazhong University of Science and Technology (HUST), China, in 1994.

He worked at The University of Hong Kong from 1998 to 2000 and as a Visiting Scholar at the University of Southern California from 1999 to 2000. He is currently a Chair Professor of computer science and engineering at HUST. He has coauthored more than 20 books and published over 900 research papers. His research interests include computer architecture, parallel and distributed computing, big data processing, data storage, and system security. He is a fellow of CCF and a Life Member of the ACM. In 1996, he was awarded the German Academic Exchange Service Fellowship to visit the Chemnitz University of Technology, Germany. He was awarded the Excellent Youth Award from the National Science Foundation of China in 2001.

Chenchen Li (Student Member, IEEE) received the B.S. degree from the Zhongnan University of Economics and Law. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. His research interests include blockchain, data mining, and finance security.

Jiang Xiao (Member, IEEE) received the B.Sc. degree from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2009, and the Ph.D. degree from The Hong Kong University of Science and Technology (HKUST) in 2014. She is currently an Associate Professor with the School of Computer Science and Technology, HUST. She has been engaged in research on blockchain and distributed computing. Her awards include the CCF-Intel Young Faculty Research Program in 2017, the Hubei Downlight Program in 2018, the ACM Wuhan Rising Star Award in 2019, and the Best Paper Awards from IEEE ICPADS/GLOBECOM/GPC.

Teng Zhang (Member, IEEE) received the Ph.D. degree in computer science from Nanjing University, China, in 2019. He is currently an Assistant Professor at the Huazhong University of Science and Technology. His major research interests include machine learning and data mining, especially on kernel methods and margin-based learning method. He has served as the Senior PC Member for IJCAI and ECAI, and a PC Member for ICML, NIPS, KDD, and AAAI.

Xiaohai Dai (Member, IEEE) received the Ph.D. degree from the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), Wuhan, China, in 2021. He is currently a Post-Doctoral Researcher with the School of Computer Science and Technology, HUST. His current research interests include blockchain and distributed systems. His awards include the Outstanding Creative Award in 2018 FISCO BCOS Blockchain Application Contest and the Top Ten in FinTechathon 2019.

Bo Li (Fellow, IEEE) received the B.Eng. degree (summa cum laude) in computer science from Tsinghua University, Beijing, China, and the Ph.D. degree from the ECE Department, University of Massachusetts Amherst. He was a Cheung Kong Scholar Visiting Chair Professor with Shanghai Jiao Tong University from 2010 to 2016 and was the Chief Technical Advisor for ChinaCache Corporation (NASDAQ:CCIH), a leading CDN provider. He is currently a Chair Professor with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology. He made pioneering contributions in multimedia communications and the internet video broadcast, which attracted significant investment from industry and received the Test-of-Time Best Paper Award from IEEE INFOCOM in 2015. He received six best paper awards from IEEE including INFOCOM in 2021. He was the Co-TPC Chair for IEEE INFOCOM in 2004.