Time-Aware Metapath Feature Augmentation for Ponzi Detection in Ethereum

Chengxiang Jin, Jiajun Zhou, Jie Jin, Jiajing Wu, Senior Member, IEEE, and Qi Xuan, Senior Member, IEEE

Abstract—With the development of Web 3.0 which emphasizes decentralization, blockchain technology ushers in its revolution and also brings numerous challenges, particularly in the field of cryptocurrency. Recently, a large number of criminal behaviors continuously emerge on blockchain, such as Ponzi schemes and phishing scams, which severely endanger decentralized finance. Existing graph-based abnormal behavior detection methods on blockchain usually focus on constructing homogeneous transaction graphs without distinguishing the heterogeneity of nodes and edges, resulting in partial loss of transaction pattern information. Although existing heterogeneous modeling methods can depict richer information through metapaths, the extracted metapaths generally neglect temporal dependencies between entities and do not reflect real behavior. In this paper, we introduce Time-aware Metapath Feature Augmentation (TMFAug) as a plug-and-play module to capture the real metapath-based transaction patterns during Ponzi scheme detection on Ethereum. The proposed module can be adaptively combined with existing graph-based Ponzi detection methods. Extensive experimental results show that our TMFAug can help existing Ponzi detection methods achieve significant performance improvements on the Ethereum dataset, indicating the effectiveness of heterogeneous temporal information for Ponzi scheme detection.

Index Terms—Ponzi scheme detection, metapath, temporal information, heterogeneous graph, ethereum, blockchain.

I. INTRODUCTION

Blockchain [1] is a peer-to-peer network system based on technologies such as cryptography [2] and consensus mechanisms [3] to create and store huge transaction information. At present, the biggest application scenario of blockchain is cryptocurrency. For example, the initial “Bitcoin” [4] also represents the birth of blockchain. As cryptocurrencies continue to evolve, smart contracts [5] bring blockchain 2.0, also known as Ethereum [6]. Unlike Bitcoin, which prefers a peer-to-peer electronic cash system, Ethereum is a platform for decentralized applications and allows anyone to create and execute smart contracts. The smart contract [7], accompanying Ethereum, is understood as a program on the blockchain that operates when the starting conditions are met. Since smart contracts operate on publicly accessible code and are immutable, it is possible to carry out secure transactions without third-party endorsement.

The Ponzi scheme [8] is a form of fraud that benefits from a poor return on investment to the victim. Traditional Ponzi schemes have existed only offline, but they have gradually taken on an online form as the spotlight of money flows has shifted online. Cryptocurrency is trusted by the public for its security. Due to the transparency and immutability of smart contracts on Ethereum, people tend to be less vigilant, which makes it easier for Ponzi schemes to execute. According to recent reports, the SEC revealed that the Forsage smart contract platform [9] is a large fraudulent pyramid scheme that marketed fraudulent products to investors, involving a total of $300 million and millions of victims. This undermines the public’s trust in cryptocurrencies. Therefore, there is an urgent need to understand the behavior of Ponzi schemes and detect them from cryptocurrency platforms, further maintaining the stability of the investment markets.

Existing Ponzi scheme detection methods based on graph analytics generally rely on homogeneous graph modeling [10], [11] due to their simplicity. However, real transactions on Ethereum generally involve different types of interactions between different types of accounts, which will be neglected during homogeneous modeling. Therefore, it is tough to reflect the complexity and diversity in real transactions simply by homogeneous graphs. Meanwhile, heterogeneous graph [12] is a widely used technique to model complex interactions, which can preserve the semantic information of interactions to the greatest extent. Typical heterogeneous techniques generally employ the metapath structure [13] to capture the specific interaction patterns. However, several approaches [14], [15] are obsessed with
We propose time-aware metapaths that impose times-aware learning and graph-based Ponzi detection methods. Section IV describes the details of constructing an account interaction graph on Ethereum. Section V introduces the details of the proposed TMFAug module. The experimental settings and analysis are presented in Section VI. Finally, we conclude this paper and prospect future work in Section VII.
investment vehicle, another part claiming to be an online bitcoin wallet that offers high daily returns, etc.

Analysis of HYIPs has revealed that a number of traditional Ponzi schemes have gradually emerged in cryptocurrencies, generating unique means of deception. For example, due to the non-interruptible nature of smart contracts, a lack of attention to contract content can lead to the failure to withdraw funds before a Ponzi scheme collapses, thus amplifying the losses of less sophisticated investors. Nowadays, multiple detection methods have been proposed to mitigate the damage caused by Ponzi schemes.

1) Detection Based on Contract Codes: The code of a smart contract defines the functionality of the contract account, enabling analysis to determine whether it is a Ponzi account. Chen et al. [25] extracted features from user accounts and smart contract opcodes, and fed them into a downstream machine learning classifier to detect Ponzi schemes. To enhance the synergy between feature engineering and machine learning, Fan et al. [26] employed the concept of ordered augmentation to train a model for detecting Ponzi schemes. Bartoletti et al. [27] explored the significance of temporal behavior and designed time-dependent features for Ponzi detection. Due to the susceptibility of machine learning models to evasion techniques, Chen et al. [28] proposed a semantics-aware detection method called SADPonzi. This method analyzes the contractual rules of a Ponzi scheme and heuristically generates semantic paths based on the tEthereum [29] vulnerability detection algorithm. Given the inherent challenges in capturing the structural and semantic features of source code behavior through feature engineering, Chen et al. [30] employed graph embedding techniques to automatically acquire highly expressive code features.

2) Detection Based on Transactions: Contract code features provide insights into the inherent characteristics of smart contracts, however, fraudsters possess the ability to intentionally evade fixed patterns, thereby rendering detection more challenging. Fortunately, transaction features describe the actual interaction patterns generated by Ponzi accounts, from which the illegality of transactions can be effectively identified. With the development of machine learning, Zhang et al. [31] proposed a new method for Ethereum Ponzi scheme detection based on an improved LightGBM algorithm, which can alleviate the imbalance problem of Ponzi data. Graph neural networks are capable of learning both account features and temporal and structural information from metapaths, and work on the integration of heterogeneous and temporal information into homogeneous approaches for Ponzi scheme detection.

However, the above-mentioned Ponzi detection methods suffer from several shortcomings. The above methods rely on homogeneous graphs, and neglect timestamps information and the types of nodes and edges, which makes it difficult to capture more complex and temporal behavior patterns. Jin et al. [36] exploited this problem on Ethereum Ponzi detection, and proposed to fuse heterogeneous information in metapaths and their subsets, further enhancing existing Ponzi detection methods. However, they neglected the timestamp information so that the constructed metapaths can not reflect the real behavior patterns. Therefore, in this paper, we focus more on capturing certain temporal and structural information from metapaths, and work on the integration of heterogeneous and temporal information into homogeneous approaches for Ponzi scheme detection.

III. PRELIMINARIES AND TERMINOLOGIES

In this section, we will introduce some concepts related to heterogeneous graph, including the definitions of heterogeneous graph itself, metapath, our proposed time-aware metapath, and the symbiotic relationship of metapaths. Fig. 1 provides a schematic illustration to further complement the description of the aforementioned concepts.

\textbf{Definition 1 (Heterogeneous Graph):} A heterogeneous graph is defined as $G_{het} = (V, E, T_V, T_E)$, where $V$ and $E$ represent the sets of nodes and edges respectively, $T_V$ and $T_E$ represent the sets of node types and edge types respectively, whereby each node (or edge) possesses a type mapping function denoted as $V \rightarrow T_V$ (or $E \rightarrow T_E$) respectively. A heterogeneous graph encompasses multiple types of nodes or edges, thereby satisfying $|T_V| + |T_E| > 2$

\textbf{Definition 2 (Metapath):} In heterogeneous graphs, metapath contains a sequence of relations defined between different types of nodes, such as $v_1 \xrightarrow{r_1} v_2 \xrightarrow{r_2} \ldots \xrightarrow{r_l} v_{l+1}$, which specifies a composite connection between node $v_1$ and $v_{l+1}$ with $r = r_1 \circ r_2 \circ \ldots \circ r_l$, where $r$ denotes the composition operation on relations.

In this paper, we propose the concept of time-aware metapath, which incorporates temporal dependencies in the sequence of relations to capture more realistic and temporally influenced node interaction patterns.

\textbf{Definition 3 (Time-aware Metapath):} For heterogeneous graphs containing temporal relations, time-aware metapath can be defined as $v_1 \xrightarrow{t_1} v_2 \xrightarrow{t_2} \ldots \xrightarrow{t_l} v_{l+1}$, where $t$ represents the timestamp information of the corresponding edge and satisfies $t_1 < t_2 < \ldots < t_l$.

We also introduce a new concept of symbiotic relationship to represent the full consistency of two time-aware metapaths after ignoring temporal information.

\textbf{Definition 4 (Symbiotic Relationship):} For time-aware metapaths $(P_1, P_2, \ldots, P_l)$, they are symbiotic if and only if they have the same node sequence and relation type sequence, i.e.,

\begin{align}
\forall \mathbf{P}_i \cong \mathbf{P}_j \cong \ldots \cong \mathbf{P}_l, \text{ s.t.} \\
\forall \mathbf{V}(P_1) = \mathbf{V}(P_2) = \ldots = \mathbf{V}(P_l) \\
\forall \mathbf{R}_t(P_1) = \mathbf{R}_t(P_2) = \ldots = \mathbf{R}_t(P_l)
\end{align}
where \( \cong \) represents a symbiotic relationship, \( V \) and \( R \) represent the node sequence and relation type sequence, respectively.

IV. ACCOUNT INTERACTION GRAPH MODELING

In this section, we focus on modeling Ethereum transaction data as account interaction graphs, as well as providing a brief introduction to Ethereum data.

A. Ethereum Data

An account in Ethereum is an entity that owns Ether and can be divided into two categories: Externally Owned Account (EOA) and Contract Account (CA) [6]. EOA is controlled by a user with a private key and can initiate transactions on Ethereum, and CA is controlled by smart contract code and can only get triggers to implement the contract function. There are generally two categories of interactions between Ethereum accounts: transaction and contract call. The transaction is the process by which an action is initiated by an EOA and received by another EOA, or ultimately back to itself. Contract calls, which are divided into external calls and internal calls, refer to the process of triggering a smart contract code that can execute many different actions, such as transferring tokens or creating a new contract.

Fig. 2 illustrates a Ponzi contract\(^2\) in the guise of a game, with a large number of interactions between the CA and an EOA involving calls to different contract functions for different purposes:

- **Contract Creation:** Create a new smart contract with built-in opcodes or keywords.
- **repairTheCastle:** Function customized by the contract owner, open to be called by the investor. When the invested amount exceeds 100 ETH, the excess is returned to the last three investors in the ratio of 55%, 30% and 15%, while the initial investor receives a return of 3%. Incur a cumulative fee of 3%.
- **collectFee:** Function customized by the contract owner. The contract owner collects all the accumulated fees.

Notably, the process of contract call is accompanied by the flow of Ether. A transaction initiated by an EOA may be accompanied by multiple sub-transactions related to the smart contract at the same time, as shown in transaction ❺.

\(^2\)https://cn.etherscan.com/address/0xa9fa83d31f1cfdf14b7f9d17f02e48dcedf9bc6b8c7c0
graph modeling, for complex interaction scenarios, we consider multiple sub-transactions with the same transaction hash as different interaction edges but with the same timestamp. The time-aware metapaths can identify whether they belong to the sub-transactions in the same transaction behavior by comparing the timestamp information.

B. Graph Modeling on Ethereum

The existing Ponzi scheme detection methods mainly use homogeneous transaction graphs, which regard all nodes and edges as the same type, neglecting the complex and temporal interaction information. On the contrary, heterogeneous graphs group nodes and edges into various categories, which can represent more complex interaction scenarios and make full use of graph information. Considering that most of the existing Ponzi detection methods are only suitable for homogeneous graphs, we consider extracting information from heterogeneous graphs to assist in Ponzi detection on homogeneous graphs.

More formally, we use $G_{hom} = (V, E, Y)$ and $G_{het} = (V_{eoa}, V_{ca}, E_{trans}, E_{call}, Y)$ to represent the two types of graphs respectively, where $V$ represents the set of accounts in the Ethereum data, $V_{eoa}$ and $V_{ca}$ represent the sets of EOA and CA respectively, $E$ represents the set of directed edges constructed from transaction information, $E_{trans}$ and $E_{call}$ represent the sets of directed edges with timestamps constructed from transaction information and contract call information respectively, $Y = \{(v_p^i, y_i)\}$ is the label information of known Ponzi accounts. Notably, all the known Ponzi schemes we have collected on Ethereum are based on contract accounts and distinguish between the different node and edge types, classifying them into two categories: 4,616 CA nodes and 52,514 EOA nodes, 984,498 transaction edges and an additional 1,780,781 call edges. In addition, we crawled 191 labeled Ponzi data from Xblock, Etherscan and other Blockchain platforms.

For all detection methods, we take all the labeled Ponzi accounts as positive samples, as well as the same number of randomly sampled CA as negative samples. The statistics of the two graphs are shown in Table II.

C. Node Feature Construction

In this paper, we divide the existing Ponzi detection methods into three categories, namely manual feature engineering, graph embedding and graph neural network. For Ponzi detection methods based on manual feature engineering and graph neural network, we construct initial features for account nodes in both $G_{hom}$ and $G_{het}$ using 15 manual features proposed in existing methods.

- The income and expenditure of the target account (including total, average, maximum and variance);
- The expenditure-income ratio of the target account;
- The balance of the target account;
- The number of transactions sent and received by the target account;
- The investment Gini and return Gini of target account;
- The life cycle of the target account.

As for methods based on graph embedding, we generate structural embeddings as account node features rather than the predefined manual features. After that, the initial feature of account and distinguish between the different node and edge types, classifying them into two categories: 4,616 CA nodes and 52,514 EOA nodes, 984,498 transaction edges and an additional 1,780,781 call edges. In addition, we crawled 191 labeled Ponzi data from Xblock, Etherscan and other Blockchain platforms.

For all detection methods, we take all the labeled Ponzi accounts as positive samples, as well as the same number of randomly sampled CA as negative samples. The statistics of the two graphs are shown in Table II.

C. Node Feature Construction

In this paper, we divide the existing Ponzi detection methods into three categories, namely manual feature engineering, graph embedding and graph neural network. For Ponzi detection methods based on manual feature engineering and graph neural network, we construct initial features for account nodes in both $G_{hom}$ and $G_{het}$ using 15 manual features proposed in existing methods.

- The income and expenditure of the target account (including total, average, maximum and variance);
- The expenditure-income ratio of the target account;
- The balance of the target account;
- The number of transactions sent and received by the target account;
- The investment Gini and return Gini of target account;
- The life cycle of the target account.

As for methods based on graph embedding, we generate structural embeddings as account node features rather than the predefined manual features. After that, the initial feature of account and distinguish between the different node and edge types, classifying them into two categories: 4,616 CA nodes and 52,514 EOA nodes, 984,498 transaction edges and an additional 1,780,781 call edges. In addition, we crawled 191 labeled Ponzi data from Xblock, Etherscan and other Blockchain platforms.

For all detection methods, we take all the labeled Ponzi accounts as positive samples, as well as the same number of randomly sampled CA as negative samples. The statistics of the two graphs are shown in Table II.

C. Node Feature Construction

In this paper, we divide the existing Ponzi detection methods into three categories, namely manual feature engineering, graph embedding and graph neural network. For Ponzi detection methods based on manual feature engineering and graph neural network, we construct initial features for account nodes in both $G_{hom}$ and $G_{het}$ using 15 manual features proposed in existing methods.

- The income and expenditure of the target account (including total, average, maximum and variance);
- The expenditure-income ratio of the target account;
- The balance of the target account;
- The number of transactions sent and received by the target account;
- The investment Gini and return Gini of target account;
- The life cycle of the target account.

As for methods based on graph embedding, we generate structural embeddings as account node features rather than the predefined manual features. After that, the initial feature of account and distinguish between the different node and edge types, classifying them into two categories: 4,616 CA nodes and 52,514 EOA nodes, 984,498 transaction edges and an additional 1,780,781 call edges. In addition, we crawled 191 labeled Ponzi data from Xblock, Etherscan and other Blockchain platforms.

For all detection methods, we take all the labeled Ponzi accounts as positive samples, as well as the same number of randomly sampled CA as negative samples. The statistics of the two graphs are shown in Table II.

C. Node Feature Construction

In this paper, we divide the existing Ponzi detection methods into three categories, namely manual feature engineering, graph embedding and graph neural network. For Ponzi detection methods based on manual feature engineering and graph neural network, we construct initial features for account nodes in both $G_{hom}$ and $G_{het}$ using 15 manual features proposed in existing methods.

- The income and expenditure of the target account (including total, average, maximum and variance);
- The expenditure-income ratio of the target account;
- The balance of the target account;
- The number of transactions sent and received by the target account;
- The investment Gini and return Gini of target account;
- The life cycle of the target account.

As for methods based on graph embedding, we generate structural embeddings as account node features rather than the predefined manual features. After that, the initial feature of account and distinguish between the different node and edge types, classifying them into two categories: 4,616 CA nodes and 52,514 EOA nodes, 984,498 transaction edges and an additional 1,780,781 call edges. In addition, we crawled 191 labeled Ponzi data from Xblock, Etherscan and other Blockchain platforms.

For all detection methods, we take all the labeled Ponzi accounts as positive samples, as well as the same number of randomly sampled CA as negative samples. The statistics of the two graphs are shown in Table II.

C. Node Feature Construction

In this paper, we divide the existing Ponzi detection methods into three categories, namely manual feature engineering, graph embedding and graph neural network. For Ponzi detection methods based on manual feature engineering and graph neural network, we construct initial features for account nodes in both $G_{hom}$ and $G_{het}$ using 15 manual features proposed in existing methods.

- The income and expenditure of the target account (including total, average, maximum and variance);
- The expenditure-income ratio of the target account;
- The balance of the target account;
- The number of transactions sent and received by the target account;
- The investment Gini and return Gini of target account;
- The life cycle of the target account.

As for methods based on graph embedding, we generate structural embeddings as account node features rather than the predefined manual features. After that, the initial feature of account and distinguish between the different node and edge types, classifying them into two categories: 4,616 CA nodes and 52,514 EOA nodes, 984,498 transaction edges and an additional 1,780,781 call edges. In addition, we crawled 191 labeled Ponzi data from Xblock, Etherscan and other Blockchain platforms.

For all detection methods, we take all the labeled Ponzi accounts as positive samples, as well as the same number of randomly sampled CA as negative samples. The statistics of the two graphs are shown in Table II.
arbitrary account node \( v_i \) is denoted as follows:

\[
x_i = \begin{cases} 
[x^1_i, x^2_i, \ldots, x^{15}_i] & \text{for manual feature} \\
[x^1_i, x^2_i, \ldots, x^{15}_i] & \text{for graph neural network} \\
\text{GEmb}(G_{\text{hom}}, v_i) & \text{for graph embedding}
\end{cases}
\]

where \( x^i \) represents the \( i \)-th manual feature, and \( \text{GEmb} \) represents an arbitrary graph embedding algorithm that generates node embeddings as initial node features.

V. **TIME-AWARE METAPATH FEATURE AUGMENTATION**

In this section, we introduce the details of \(\text{TMEAug} \), as schematically depicted in Fig. 5, which is composed of three components: symbiotic relationship merging, behavior refinement and filtering, and metadata feature aggregation.

A. **Time-Aware Metapath**

Fig. 4 shows several representative Ponzi scheme interaction networks in Ethereum. The gear icon and portrait icon represent CA and EOA respectively, the black line represents transactions with the specific transaction amount, and the green line represents the contract call with a specific function. The CA on the left is a Ponzi account that defrauds investors by engaging in two-way transactions, in which it earns more and pays less. In addition, Fig. 2 also illustrates that transactions can be triggered by contract calls. Based on the above observations, we can predefine the following metapaths to characterize the critical behavior patterns present in Ethereum Ponzi schemes:

\[
P^1_1: EOA \xrightarrow{\text{call}} CA \xrightarrow{\text{trans}} EOA \\
P^2_2: EOA \xrightarrow{\text{call}} CA \xrightarrow{\text{call}} CA \xrightarrow{\text{trans}} EOA
\]

The rationality of the above predefined metapaths can be confirmed by statistical evidence. Specifically, we extract all metapaths that meet the above definition from the Ethereum interaction graph constructed in Section IV-B, using the labeled 382 CA (Ponzi and non-Ponzi) as \( CA \), and ultimately obtain a total of 6,936,126,959 metapaths, of which the \( P^1_1 \) occupy 99.998%.

Furthermore, for all \( P^1_1 \) and \( P^2_2 \) obtained, the number of metapaths in which \( CA \) is a Ponzi account occupies 94.309%. The above evidence suggests that Ponzi accounts usually interact with EOA via mode \( P^1_1 \). And \( EOA_2 \) could be an external investor or the Ponzi contract creator, the former indicates that \( trans \) is a payback, while the latter indicates that \( trans \) is a transfer of funds to the defrauder. In addition, a tiny percentage of Ponzi accounts transfer money via mode \( P^2_2 \), i.e., internal calls. This mode circumvents the regular transaction pattern to some extent, but it is rarely used because too many contract calls consume more gas fees and increase the cost of fraud.

Ponzi schemes reward old investors through new investment income, i.e., two-way transactions do not occur simultaneously, so here we further introduce the time information of transactions to depict the temporal behavior patterns. Formally, we use the following **time-aware metapaths** to characterize the temporal behavior patterns:

\[
P^1: EOA \xrightarrow{\text{call}} CA \xrightarrow{\text{trans}} EOA \\
P^2: EOA \xrightarrow{\text{call}} CA \xrightarrow{\text{call}} CA \xrightarrow{\text{trans}} EOA
\]  \quad (4)

Note that the timing constraint \( t_1 < t_2 < t_3 \) ensures that the behavior patterns represented by the metapaths are more consistent with the real interaction rules. Under the timing constraint, the number of effective metapaths is reduced to 3,280,360,935.

More specifically, 93.837% of the time-aware metapaths have a \( CA \), that corresponds to a Ponzi account (93.839% for \( P^1 \) and 11.733% for \( P^2 \)), which also suggests that compared with normal accounts, Ponzi accounts are more inclined to use the interaction patterns defined by the above metapaths to commit fraud. Additionally, we refer to the metapaths defined in (3) as timeless metapaths, which ignores the timing of interactions and is based on an after-the-fact perspective.

B. **Symbiotic Relationship Merging**

There are usually frequent interactions between multiple accounts, yielding multiple metapaths with the same sequences of nodes and relationships, but different sequences of timestamps.

Here we consider these metapaths to be symbiotic with each other, and the concept of symbiotic relationship is defined in Section III. As illustrated in Fig. 5(a), \( P_1 \), \( P_2 \) and \( P_3 \) are symbiotic to each other because they have the same sequences of nodes and relationships \( E_1 \xrightarrow{\text{call}} C_1 \xrightarrow{\text{trans}} E_2 \), and so do \( P_4 \) and \( P_5 \). Since symbiotic metapaths are numerous and have the same form, we use a super metapath to substitute them. Specifically, given a set of symbiotic metapaths \( \{ P_1, P_2, \ldots, P_n \mid P_1 \cong P_2 \cong \cdots \cong P_n \} \), we merge these symbiotic metapaths into a super metapath \( P \). Intuitively, the number of symbiotic metapaths reflects the account’s preference in participating in this behavior to a certain extent. So we measure the importance of the super metapath via the number of symbiotic metapaths and further assign it an importance factor \( \omega \). As illustrated in Fig. 5(a), \( \{ P_1, P_2, P_3 \} \) (or \( \{ P_4, P_5 \} \)) is merged into a super metapath \( P_1 \) (or \( P_2 \)) and assigned an importance factor \( \omega_1 = 3 \) (or \( \omega_2 = 2 \)).
Fig. 5. Overall framework of the Time-aware Metapath Feature Augmentation. The complete workflow proceeds as follows: (a) merging symbiotic relations to obtain the super metapaths; (b) refining the super metapaths based on the behavior, and filtering the super metapaths by Top-$K$; and (c) aggregating the information along metapaths to the target nodes.

After symbiotic relationship merging, we recount the number of time-aware metapaths (and the proportion of $CA_1$ that is Ponzi account), obtaining 32,752,370 (81.128%) for $P^1$ and 9,149 (6.285%) for $P^2$ respectively. It can be seen that the number of metapaths is extremely compressed after merging, but the proportion of interactions involving Ponzi accounts remains high, which supports the assumption of using such predefined metapaths to characterize the interaction behavior of Ponzi accounts.

C. Behavior Refinement and Filtering

The temporal behavior patterns in (4) are coarse-grained, because the relationship between the two EOA before and after, or the relationship between $CA$ and $CA_*$, is undetermined. We further refine the coarse-grained metapaths based on the different behavior patterns between Ponzi and normal accounts, yielding six cases (symbolized as $P_{a,b}$):

- $P^{1.1}$: $EOA_1 \xrightarrow{\text{call}_{t_1}} CA_1 \xrightarrow{\text{trans}_{t_2}} EOA_1$
  It reflects four behavior patterns: 1) ponzi account accepts new external investment and then rewards old investor; 2) ponzi account accepts new external investment and then transfers commission to the fraudster; 3) fraudster invests and rewards himself to attract external investors; 4) fraudster sweeps commission.
- $P^{1.2}$: $EOA_1 \xrightarrow{\text{call}_{t_1}} CA_1 \xrightarrow{\text{trans}_{t_2}} EOA_1$
  It reflects three behavior patterns: 1) ponzi account accepts new external investment and rewards this investor later; 2) fraudster invests and rewards himself to attract external investors; 3) fraudster sweeps commission.
- $P^{2.1}$: $EOA_1 \xrightarrow{\text{call}_{t_1}} CA_1 \xrightarrow{\text{call}_{t_2}} CA_2 \xrightarrow{\text{trans}_{t_3}} EOA_2$
  This metapath reflects four behavior patterns, which are similar to $P^{1.1}$.
- $P^{2.2}$: $EOA_1 \xrightarrow{\text{call}_{t_1}} CA_1 \xrightarrow{\text{call}_{t_2}} CA_2 \xrightarrow{\text{trans}_{t_3}} EOA_1$
  This metapath reflects three behavior patterns, which are similar to $P^{1.2}$.
- $P^{2.3}$: $EOA_1 \xrightarrow{\text{call}_{t_1}} CA_1 \xrightarrow{\text{call}_{t_2}} CA_1 \xrightarrow{\text{trans}_{t_3}} EOA_2$
- $P^{2.4}$: $EOA_1 \xrightarrow{\text{call}_{t_1}} CA_1 \xrightarrow{\text{call}_{t_2}} CA_1 \xrightarrow{\text{trans}_{t_3}} EOA_1$

After analyzing the extracted metapaths, we find that the behavior patterns involving contract self-call only occur in normal interactions.

Notably, $a$ indicates the coarse-grained metapath from which this fine-grained metapath is derived. For example, $a = 1$ indicates that $P^{1,b}$ is derived from $P^1$. And $b$ is the index of a
different fine-grained pattern produced by a further refinement of $P^n$. For example, $P^1$ can further derive two fine-grained metapath patterns, so $b \in \{1, 2\}$ when $a = 1$; $P^2$ can further derive four fine-grained metapath patterns, so $b \in \{1, 2, 3, 4\}$ when $a = 2$.

According to the heterogeneous interaction graph constructed in Table II, we count the number of all coarse and refined metapaths, as reported in Table III. It can be clearly seen that the number of coarse metapaths is quite large, while the number of different refined metapaths varies significantly. In order to avoid the information redundancy and noise caused by too many metapaths, we introduce an importance-based Top-$K$ filtering strategy to sample the more important metapaths. Specifically, we first merge refined symbiotic metapaths into refined super metapaths, during which an importance factor $\omega$, which reflects the number of merged symbiotic metapaths, is derived and appended to the corresponding super metapaths. We believe that the more frequently a class of metapaths occurs, the more helpful it is in characterizing account features by the behavioral patterns it carries. Then we perform Top-$K$ filtering guided by $\omega$, i.e., sampling the refined super metapaths with importance factors in the top $K$ for subsequent feature aggregation. As illustrated in Fig. 5(b), for each kind of refined super metapaths $P^1$ (or $P^2$), those with larger importance factors will be preserved, such as $P^1_{1,1}$ (or $P^2_{2,2}$).

### D. Metapath Feature Aggregation

The above two steps aim to obtain the important behavior information represented by super metapaths, and the super metapaths retained during Top-$K$ filtering will participate in the subsequent feature aggregation step. TMFAug is designed to fuse the behavior features represented by metapaths and finally enhance existing Ponzi detection methods on homogeneous Ethereum interaction graph. However, the preserved super metapaths are still numerous. In order to alleviate information redundancy and feature explosion during metapath feature aggregation, we adjust the importance factor of super metapaths before doing so. Specifically, for super metapaths starting from the same head node $v_{ea}$, we first group them as follows:

$$M^{\square} = \{ P^{\square}_i \mid \bar{V}(P^{\square}_i)[1] = v_{ea} \}, \quad \text{s.t. } \square \in \{1, 2\}$$

where $\bar{V}(P)[1]$ represents getting the first element in the node sequence (i.e., head node) of super metapath $P$. Then we compute the normalized importance factor for each super metapath $P^{\square}_i$ in $M^{\square}$ as follows:

$$\hat{\omega}^{\square}_i = \omega^{\square}_i / \sum_{P \in M^{\square}_i} \omega, \quad \text{s.t. } \square \in \{1, 2\}$$

After adjusting the importance factor of all the super metapaths, we then perform feature aggregation to update the CA features. Specifically, for a target CA node $v_{ea}$ (CA in (4)), we first group the super metapaths that contain it as follows:

$$M^{\square}_e = \{ P^{\square}_i \mid \bar{V}(P^{\square}_i)[2] = v_{ea} \}, \quad \text{s.t. } \square \in \{1, 2\}$$

After grouping the super metapaths, we update the features of the target CA by aggregating the features of other nodes in the metapath, and the final target CA feature is obtained by processing multiple super metapaths of the same group, as illustrated in Fig. 5(c). The process of feature update can be represented as follows:

$$\hat{x}^{\square}_{ea} = \sum_{P \in M^{\square}_e} \hat{\omega} \cdot \sum_{v \in V(P)} x_v, \quad \text{s.t. } \square \in \{1, 2\}$$

where $x_v$ is the initial account feature of nodes along the metapath. Finally, the updated features $\hat{x}$ contain heterogeneous structural information associated with temporal behavior patterns and will be fed into the downstream classification task.

### VI. Experiment

In this section, we evaluate the effectiveness of our TMFAug on improving Ponzi scheme detection by answering the following research questions:
- **RQ1:** Can our TMFAug improve the performance of Ponzi scheme detection when combined with existing detection methods?
- **RQ2:** Whether time information can help capture more effective behavior patterns?
- **RQ3:** How do different behavior patterns affect the detection results?
- **RQ4:** How does the behavior refinement affect the detection results?
- **RQ5:** How does the behavior filtering affect the detection results?

### A. Ponzi Detection Methods and Experimental Setup

To illustrate the effectiveness and generality of our TMFAug module, we combine it with three categories of Ponzi detection methods: manual feature engineering, graph embedding and GNN-based methods. To illustrate the superiority of our TMFAug module, we introduce a MFAug module and the HFAug module [36] for comparison. The differences between the three modules are shown in the Table V.

Manual feature engineering is the most common approach for Ponzi detection, we use 15 manual features listed in Section IV-C to construct the initial features for account nodes. For Ponzi detection based on graph embedding, we consider Line, DeepWalk, Node2Vec and Trans2Vec algorithms for constructing initial account node features, respectively. For the above two categories of methods, we achieve Ponzi detection by feeding the generated...
TABLE IV
PONZI DETECTION RESULTS OF RAW METHODS AND THEIR ENHANCED VERSIONS (WITH TMFAUG OR MFAUG) IN TERMS OF MICRO-F1 (GAIN), AND GAIN REPRESENTS THE RELATIVE IMPROVEMENT RATE

| Method       | raw | MFAug(P²) | TMFAug(F²) | MFAug(P²) | TMFAug(F²) | MFAug(P² + P²) | TMFAug(F² + F²) |
|--------------|-----|-----------|------------|-----------|------------|----------------|----------------|
| Manual       | SVM | 73.03     | 71.20 (-2.51%) | 71.98 (-1.44%) | 73.30 (+0.37%) | 73.83 (+1.11%) | 73.03 (+0.00%) | 72.25 (+1.07%) |
| Feature      | RF  | 73.32     | 76.45 (+4.27%) | 75.65 (+3.18%) | 76.44 (+2.62%) | 75.93 (+3.56%) | 76.97 (+4.98%) | 76.96 (+4.96%) |
| Line         | SVM | 76.98     | 76.19 (-1.03%) | 76.98 (+0.00%) | 78.02 (+1.35%) | 78.28 (+1.69%) | 75.40 (-2.05%) | 76.98 (+0.00%) |
|              | RF  | 76.97     | 77.49 (+0.68%) | 76.18 (-1.03%) | 78.27 (+1.69%) | 79.33 (+3.07%) | 76.19 (-1.01%) | 77.23 (+1.32%) |
| Trans2Vec    | SVM | 77.92     | 78.18 (+0.33%) | 80.00 (+2.47%) | 77.66 (-0.33%) | 78.70 (+1.00%) | 78.44 (+0.67%) | 79.22 (+1.67%) |
|              | RF  | 77.92     | 78.44 (+0.66%) | 79.22 (+1.67%) | 76.88 (-1.33%) | 79.48 (+2.00%) | 77.92 (-0.00%) | 79.56 (+2.62%) |
| DeepWalk     | SVM | 83.00     | 85.09 (+2.52%) | 85.09 (+2.52%) | 85.09 (+2.52%) | 84.82 (+2.19%) | 85.09 (+2.52%) | 85.09 (+2.52%) |
|              | RF  | 81.43     | 83.79 (+2.90%) | 84.31 (+3.54%) | 83.00 (+1.93%) | 84.57 (+3.86%) | 84.83 (+4.18%) | 85.26 (+2.25%) |
| Node2Vec     | SVM | 84.56     | 86.15 (+1.87%) | 86.15 (+1.87%) | 85.08 (+0.61%) | 85.36 (+0.95%) | 86.67 (+2.50%) | 86.15 (+1.87%) |
|              | RF  | 86.13     | 86.40 (+0.31%) | 86.92 (+0.92%) | 87.19 (+1.23%) | 86.41 (+0.33%) | 86.66 (+0.62%) | 86.66 (+0.62%) |
| LightGBM     | [31] | 75.14     | 76.98 (+2.45%) | 79.07 (+5.23%) | 77.23 (+2.78%) | 78.54 (+4.52%) | 79.40 (+0.35%) | 77.25 (+2.81%) |
| GCN          | SVM | 84.50     | 85.71 (+1.43%) | 85.71 (+1.43%) | 85.71 (+1.43%) | 86.39 (+2.24%) | 85.60 (+1.30%) | 85.92 (+1.68%) |
| GAT          | SVM | 84.24     | 87.06 (+3.35%) | 87.59 (+3.98%) | 86.40 (+2.80%) | 87.12 (+3.42%) | 87.06 (+3.35%) | 87.68 (+4.08%) |
| GIN          | SVM | 84.59     | 85.33 (+0.87%) | 85.08 (+0.58%) | 85.86 (+1.50%) | 86.38 (+2.12%) | 86.12 (+1.81%) | 85.08 (+0.58%) |
| SAGE         | SVM | 85.70     | 86.75 (+1.23%) | 86.91 (+1.41%) | 86.65 (+1.11%) | 86.86 (+1.35%) | 86.06 (+0.42%) | 86.59 (+1.04%) |
| FAGNN        | SVM | 85.34     | 85.92 (+0.68%) | 86.07 (+0.86%) | 85.76 (+0.49%) | 86.35 (+1.42%) | 85.76 (+0.49%) | 85.92 (+0.68%) |
| Avg. Rank    | 6.19 | 3.94    | 3.00     | 3.94    | 2.56    | 3.81    | 2.94    |

The best enhanced results in each method are marked with boldface. Avg. Rank represents the average rank.

TABLE V
SUMMARY OF THE DIFFERENCES IN DIFFERENT FEATURE AUGMENTATION MODULES

| Module | Timestamp | Refinement | Num. of Metapaths |
|--------|-----------|------------|------------------|
| HFAug  | ✓         | ✓          | One              |
| MFAug  | ✓         | ✓          | Multiple         |
| TMFAug | ✓         | ✓          | Multiple         |

account features into two machine learning classifiers: Support Vector Machine (SVM) [37] and Random Forest (RF) [38]. Besides, we use LightGBM from Zhang et al. [31] as a new classifier for comparison experiments. When combined with TMFAug, we feed the updated features generated via TMFAug into downstream classifiers. For GNN-based methods, we use four popular GNNs: GCN, GAT, GIN, SAGE and FAGNN, with the same initial features as manual feature engineering.

For DeepWalk and Node2Vec, we set the dimension of embedding, window size, walk length and the number of walks per node to 128, 10, 50 and 5 respectively. For Node2Vec, we perform a grid search of return parameter p and in-out parameter q in {0.5, 1, 2}. For Trans2Vec, we set the dimension of embedding, window size, walk length and the number of walks per node to 64, 10, 50, 5 respectively. For Line, we set the embedding dimension and order to 32 and 2, respectively. For GNN-based methods, we set the hidden dimension of GCN, GAT, GIN, SAGE and FAGNN to 128, 32, 128, 128 and 128 respectively, and the learning rate to 0.01, 0.1, 0.1, 0.01 and 0.01 respectively. For filtering parameter K, we vary it in {1%, 3%, 5%, 7%, 9%, 10%, 20%, 30%, 40%, 50%} and choose the best results for Ponzi detection. For all methods, we repeat 5-fold cross-validation five times with five different random seeds and report the average micro-F1 score over $5 \times 5 = 25$ experiments.

B. Enhancement for Ponzi Detection (RQ1)

Table IV reports the results of performance comparison between the raw methods and their enhanced version (with TMFAug), from which we can observe that there is a significant boost in detection performance across all methods. Overall, these detection methods combined with TMFAug obtain higher average detection performance in most cases, and the TMFAug achieves a 93.75% success rate$^3$ on improving Ponzi detection.

Specifically, except for the partial results of manual feature engineering and Line, our TMFAug consistently enhances detection performance, yielding $1.11\%$−$5.23\%$, $0.32\%$−$3.86\%$, $0.58\%$−$4.08\%$ relative improvement on the three categories of detection methods, respectively. It is noteworthy that detection methods exhibiting higher initial performance are generally more likely to experience positive gains from our TMFAug. Meanwhile, our TMFAug achieves a 100% success rate when combining with state-of-the-art detection approaches, suggesting its powerful generality. In this regard, we make the following reasonable explanation. Both random-walk-based methods and GNN-based methods are capable of acquiring structured account features during detection, while our module can also capture structured behavior features. The combination of these two aspects brings consistent positive gains.

Fig. 6 shows the difference in the number of Ponzi accounts detected by different Ponzi detection algorithms before and after the enhancement.

$^3$The success rate refers to the percentage of enhanced methods with F1 score higher than that of the corresponding raw methods.
after applying TMFAug, from which we can observe that our module effectively helps various Ponzi detection algorithms to identify more Ponzi accounts, demonstrating its effectiveness and generality. However, it should be noted that due to the limited number of Ponzi account samples in the testing set (only 38), the increase in identified Ponzi accounts after applying TMFAug may be relatively small.

Fig. 7 illustrates the performance difference between the three different feature augmentation modules on GNN-based detection algorithms, from which it can be seen that our TMFAug consistently achieves the best results across all settings, validating its superiority. Specifically, comparing MFAug and HFAug, the former’s performance advantage stems from utilizing a large number of refined metapaths to capture diverse interaction features. Comparing MFAug and TMFAug, the performance advantage of the latter benefits from utilizing time-aware metapaths to capture dynamic interaction features.

These phenomena provide a positive answer to RQ1, indicating that the TMFAug module can benefit the existing Ponzi detection methods via feature augmentation and improve their performance without adjusting them.

C. Superiority of Temporal Information (RQ2)

After evaluating the overall performance of our method, we further investigate the superiority of time-aware metapaths over time-less ones. By comparing the two modules (TMFAug and MFAug) in Table IV, we observe that methods with TMFAug outperform those with MFAug at most cases. Specifically, TMFAug achieves a higher average performance ranking than MFAug, i.e., $Rank(3.00) > Rank(3.94)$ in $P^1$, $Rank(2.56) > Rank(3.94)$ in $P^2$, $Rank(2.94) > Rank(3.81)$ in $(P^1 + P^2)$. Taking Trans2Vec as an example, in the experiment of integrating $P^2$ metapaths, MFAug that neglect dynamics brings negative gains for identifying Ponzi accounts. These phenomena provide a positive answer to RQ2, indicating that the temporal behavior patterns captured by the TMFAug module are more effective for improving Ponzi detection than time-less ones.

D. Impact of Behavior Pattern (RQ3)

We further investigate the impact of different behavior patterns on detecting Ponzi schemes. Specifically, we compare a total of six combinations of different metapaths ($P^1$, $P^2$ and $P^1 + P^2$) and different modules (TMFAug and MFAug), as shown in Table IV. As we can see, for TMFAug or MFAug, the performance ranking of different metapaths is consistent: $TMFAug(P^2) > TMFAug(P^1 + P^2) > TMFAug(P^1)$ and $MFAug(P^2) > MFAug(P^1 + P^2) > MFAug(P^1)$, which suggests that the enhancement effect relies on the choice of metapaths and answers to RQ3. Both $P^1$ and $P^2$ are extracted from the basic behavior patterns of Ponzi and normal accounts defined in (4), and we have reasonable explanations for their performance difference. As shown in Table III, the number of $P^1$ far exceeds that of $P^2$, even with the existence of relation merging and behavior filtering, the number of retained super metapaths is still huge, which inevitably leads to information redundancy and over-smoothing during feature aggregation. On the other hand, compared with $P^1$, $P^2$ contains more diverse and fine-grained behavior patterns, which can better characterize the differences between Ponzi accounts and normal accounts. In addition, this is also illustrated by the results on manual feature engineering and Line, e.g., aggregating features from $P^1$ may lead to a negative gain in performance.

E. Impact of Behavior Refinement (RQ4)

Behavior refinement is proposed to characterize different behavior patterns in fine granularity, and we further investigate its effectiveness. Table VI reports the performance comparison of TMFAug with and without behavior refinement. Overall, we can observe that TMFAug with behavior refinement achieves a higher average local ranking across different metapaths, i.e., $Rank(1.56) > Rank(1.88)$ in $P^1$, $Rank(1.31) > Rank(2.06)$ in $P^2$, and $Rank(1.56) > Rank(1.88)$ in $(P^1 + P^2)$. Furthermore, behavior refinement works better on $P^2$, manifested as a larger ranking boost. Here we combine the properties of different metapaths to provide further analysis. First, $P^1$ has a straightforward structure, so we can just refine it into two cases. Despite the behavior refinement, simple interactions do not lead to more discriminative patterns, nor do they

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
further yield gains. While $P^2$ has a more complex structure and contains more behavior patterns, especially patterns unique to normal accounts, and thus can be refined into four cases. Behavior refinement can process coarse-grained information into fine-grained and discriminative patterns, so it is more beneficial for more complex $P^2$. Moreover, behavior refinement can help to better perform subsequent behavior filtering. These phenomena provide a positive answer to RQ4, indicating that the TMF Aug somewhat relies on behavior refinement, and the more complex interaction patterns access to their superior performance.

F. Impact of Behavior Filtering (RQ5)

Behavior filtering is proposed to alleviate information redundancy and feature explosion during metapath feature aggregation, and we further investigate its effectiveness. Table VI reports the performance comparison of TMF Aug with and without behavior filtering. Overall, we can observe that TMF Aug with behavior filtering achieves a higher average local ranking across different metapaths, i.e., $Rank(P1) > Rank(2.38)$ in $P^1$, $Rank(1.56) > Rank(2.56)$ in $P^2$, and $Rank(1.56) > Rank(2.31)$ in $(P^1 + P^2)$, indicating its effectiveness. In this regard, we make the following reasonable explanation. Without behavior filtering, metapaths are not only redundant in number, but also contain much unimportant information. When updating node features along metapaths, using too many metapaths could greatly destroy the original feature representation of manual feature engineering and Line, so noisy metapaths do not severely affect them. These phenomena provide a positive answer to RQ5, indicating that the behavior filtering can effectively alleviate information redundancy and feature explosion during metapath feature aggregation, further improving Ponzi detection.

VII. CONCLUSION

Existing methods for Ponzi scheme detection usually neglect complex and temporal interaction behaviors. In this paper, we propose a Time-aware Metapath Feature Augmentation module, which includes symbiotic relationship merging, behavior refinement and filtering, and metapath feature aggregation. This module can effectively capture the real metapath-based transaction patterns and aggregate the temporal behavior information during Ponzi scheme detection on Ethereum. Experiments show that this module can significantly improve the performance of existing Ponzi scheme detection methods. Light of the fact that the current method requires expert knowledge to capture metapaths, which is time-consuming. Our future research will attempt to reduce the manual definition of metapaths, for instance by automatically learning metapaths [39], [40], [41], to save time and improve generalization.

REFERENCES

[1] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, “A survey on the security of blockchain systems,” Future Gener. Comput. Syst., vol. 107, pp. 841–853, 2020.
[2] D. R. Stinson, Cryptography: Theory and Practice. Boca Raton, FL, USA: Chapman and Hall/CRC, 2005.
[3] B. Lashkari and P. Musilek, “A comprehensive review of blockchain consensus mechanisms,” IEEE Access, vol. 9, pp. 43620–43652, 2021.
[4] C. S. Wright, “Bitcoin: A peer-to-peer electronic cash system,” SSRN Electron. J., vol. 3440802, pp. 1–2139, 2008.
[5] V. Buterin, “A next-generation smart contract and decentralized application platform,” White Paper, vol. 3, no. 37, pp. 2/1, 2014.
[6] G. Wood, “Ethereum: A secure decentralised generalised transaction ledger,” Ethereum Project Yellow Paper, vol. 151, 2014, pp. 1–32, 2014.
[7] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F.-Y. Wang, “Blockchain-enabled smart contracts: Architecture, applications, and future trends,” IEEE Trans. Syst., Man, Cybern., Syst., vol. 49, no. 11, pp. 2266–2277, Nov. 2019.
[8] M. Artzrouni, “The mathematics of Ponzi schemes,” Math. Soc. Sci., vol. 84, no. 2, pp. 190–201, 2009.
[9] T. Kell, H. Yousaf, S. Allen, S. Meiklejohn, and A. Juels, “Forsage: Anatomy of a smart-contract pyramid scheme,” in Proc. Int. Conf. Financial Cryptography Data Secur., 2023, pp. 241–258.
[10] D. Macpherson, “A survey of homogenous structures,” Discrete Math., vol. 311, no. 15, pp. 1599–1634, 2011.
[11] J. Zhou, C. Hu, J. Chi, J. Wu, M. Shen, and Q. Xuan, “Behavior-aware account de-anonymization on ethereum interaction graph,” IEEE Trans. Inf. Forensics Secur., vol. 17, pp. 3433–3448, 2022.
[12] C. Shi, Y. Li, J. Zhang, Y. Sun, and Y. Philip, “A survey of heterogeneous information network analysis,” IEEE Trans. Knowl. Data Eng., vol. 29, no. 1, pp. 17–37, Jan. 2017.
[13] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, “PathSim: Meta path-based top-k similarity search in heterogeneous information networks,” Proc. VLDB Endowment, vol. 4, no. 11, pp. 992–1003, 2011.
[14] Y. Dong, N. V. Chawla, and A. Swanii, “metapath2vec: Scalable representation learning for heterogeneous networks,” in Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2017, pp. 135–144.

[15] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, “Heterogeneous information networks embedding for recommendation,” IEEE Trans. Knowl. Data Eng., vol. 31, no. 2, pp. 357–370, Feb. 2019.

[16] B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: Online learning of social representations,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2014, pp. 701–710.

[17] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2016, pp. 855–864.

[18] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “Line: Large-scale information network embedding,” in Proc. 24th Int. Conf. World Wide Web, 2015, pp. 1067–1077.

[19] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in Proc. Int. Conf. Learn. Representations, 2017, pp. 1–14.

[20] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in Proc. Int. Conf. Learn. Representations, pp. 1–12, 2018.

[21] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” Adv. Neural Inf. Process. Syst., vol. 30, pp. 1–11, 2017.

[22] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?” in Proc. Int. Conf. Learn. Representations, 2019, pp. 1–17.

[23] T. Moore, J. Han, and R. Clayton, “The postmodern Ponzi scheme: Empirical analysis of high-yield investment programs,” in Proc. Int. Conf. Financial Cryptography and Data Secur., 2012, pp. 41–56.

[24] 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 Cryptography and Data Secur., 2015, pp. 44–61.

[25] 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., 2018, pp. 1409–1418.

[26] S. Fan, S. Fu, H. Xu, and C. Zhu, “Expose your mask: Smart Ponzi schemes detection on blockchain,” in Proc. IEEE Int. Joint Conf. Neural Netw., 2020, pp. 1–7.

[27] M. Bartolletti, S. Carta, T. Cimoli, and R. Saia, “Dissecting Ponzi schemes on ethereum: Identification, analysis, and impact,” Future Gener. Comput. Syst., vol. 102, pp. 259–277, 2020.

[28] W. Chen et al., “SADPonzi: Detecting and characterizing Ponzi schemes in ethereum smart contracts,” Proc. ACM Meas. Anal. Comput. Syst., vol. 5, no. 2, pp. 1–30, 2021.

[29] J. Krupp and C. Rossow, “[teEther]: Gaining at ethereum to automatically exploit smart contracts,” in Proc. 27th USENIX Secur. Symp., 2018, pp. 1317–1333.

[30] Y. Chen, H. Dai, X. Yu, W. Hu, Z. Xie, and C. Tan, “Improving Ponzi scheme detection on ethereum using multi-channel TextCNN and transformer,” Sensors, vol. 21, no. 19, 2021, Art. no. 6417.

[31] Y. Zhang, W. Yu, Z. Li, S. Raza, and H. Cao, “Detecting ethereum Ponzi schemes based on improved lightGBM algorithm,” IEEE Trans. Comput. Social Syst., vol. 9, no. 2, pp. 624–637, Apr. 2022.

[32] S. Yu, J. Jin, Y. Xie, J. Shen, and Q. Xuan, “Ponzi scheme detection in ethereum transaction network,” in Proc. Int. Conf. Blockchain Trustworthy Syst., 2021, pp. 175–186.

[33] J. Liu, J. Zheng, J. Wu, and Z. Zheng, “FA-GNN: Filter and augment graph neural networks for account classification in ethereum,” IEEE Trans. New. Sci. Eng., vol. 9, no. 4, pp. 2579–2588, Jul./Aug. 2022.

[34] J. Wu et al., “Who are the phishers? phishing scam detection on ethereum via network embedding,” IEEE Trans. Man., Cybern., Syst., vol. 52, no. 2, pp. 1156–1166, Feb. 2022.

[35] J. Jin, J. Zhou, C. Jin, S. Yu, Z. Zheng, and Q. Xuan, “Dual-channel early warning framework for ethereum Ponzi schemes,” in Proc. 7th China Nat. Conf. Big Data Social Comput., 2022, pp. 260–274.

[36] C. Jin, J. Jin, J. Zhou, J. Wu, and Q. Xuan, “Heterogeneous feature augmentation for Ponzi detection in ethereum,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 69, no. 9, pp. 3919–3923, Sep. 2022.

[37] W. S. Noble, “What is a support vector machine?”, Nature Biotechnol., vol. 24, no. 12, pp. 1565–1567, 2006.

[38] M. R. Segal, “Machine learning benchmarks and random forest regression,” 2004. [Online]. Available: https://escholarship.org/uc/item/7735h934

[39] L. Wang, C. Gao, C. Huang, R. Liu, W. Ma, and S. Vosoughi, “Embedding heterogeneous networks into hyperbolic space without meta-path,” in Proc. AAAI Conf. Artif. Intell., 2021, vol. 35, no. 11, pp. 10147–10155.

[40] Y. Chang, C. Chen, W. Hu, Z. Zheng, X. Zhou, and S. Chen, “Megnn: Meta-path extracted graph neural network for heterogeneous graph representation learning,” Knowl.-Based Syst., vol. 235, 2022, Art. no. 107611.

Chengxiang Jin received the B.S. degree in electrical engineering and automation from the Zhejiang University of Science and Technology, Hangzhou, China, in 2021. He is currently working toward the M.S. degree in control theory and engineering with the Zhejiang University of Technology, Hangzhou. His research interests include graph data mining and blockchain data analytics, especially for heterogeneous graph mining in ethereum.

Jie Jin received the B.S. degree in automation from Hunan International Economics University, Hunan, China, in 2020. He is currently working toward the M.S. degree with the College of Information Engineering, Zhejiang University of Technology. His research interests include graph data mining, cyberspace security, and data management.

Jiajun Zhou received the Ph.D. degree in control theory and engineering from the Zhejiang University of Technology, Hangzhou, China, in 2023. He is currently a Postdoctoral Research Fellow with the Institute of Cyberspace Security, Zhejiang University of Technology. His research interests include graph data mining, cyberspace security, and data management.

Jiajing Wu (Senior Member, IEEE) received the Ph.D. degree from The Hong Kong Polytechnic University, Hong Kong, in 2014. In 2015, she joined Sun Yat-sen University, Guangzhou, China, where she is currently an Associate Professor. Her research interests include blockchain, graph mining, and network science. Dr. Wu was the recipient of Hong Kong Ph.D. Fellowship Scheme during her Ph.D. degree in Hong Kong from 2010 to 2014. She is also an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS.

Qi Xuan (Senior Member, IEEE) received the B.S. and Ph.D. degrees in control theory and engineering from Zhejiang University, Hangzhou, China, in 2003 and 2008, respectively. From 2008 to 2010, he was a Postdoctoral Researcher with the Department of Information Science and Electronic Engineering, Zhejiang University, and a Research Assistant with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, in 2010 and 2017, respectively. From 2012 to 2014, he was a Postdoctoral Fellow with the Department of Computer Science, University of California at Davis, Davis, CA, USA. He is currently a Professor with the Institute of Cyberspace Security, College of Information Engineering, Zhejiang University of Technology, Hangzhou. His research interests include network science, graph data mining, cyberspace security, machine learning, and computer vision.