electronics

MDPI

Article

Graph Embedding-Based Money Laundering Detection for Ethereum

Jiayi Liu [1], Changchun Yin [1], Hao Wang [1], Xiaofei Wu [2], Dongwan Lan [1], Lu Zhou [1,*] and Chunpeng Ge [3]

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210000, China; liujiayi@nuaa.edu.cn (J.L.); ycc0801@nuaa.edu.cn (C.Y.); wangh24@nuaa.edu.cn (H.W.); lan.dw@nuaa.edu.cn (D.L.)
[2] College of Software Engineering, East China Normal University, Shanghai 200000, China; wuxiaofei@nuaa.edu.cn
[3] Joint SDU-NTU Centre for Artificial Intelligence Research (C-FAIR) & Software School, Shandong University, Jinan 250000, China; gechunpeng2022@126.com
* Correspondence: lu.zhou@nuaa.edu.cn

Abstract: The number of money laundering crimes for Ethereum and the amount involved have grown exponentially in recent years. However, previous studies related to anomaly detection for Ethereum usually consider multiple types of financial crimes as a whole, ignoring the apparent differences between money laundering and other malicious activities and lacking a more granular detection targeting money laundering. In this paper, for the first time, we propose an improved graph embedding algorithm specifically for money laundering detection called GTN2vec. By mining Ethereum transaction records, the algorithm comprehensively considers the behavioral patterns of money launderers and structural information of transaction networks and can automatically extract features of money laundering addresses. Specifically, we fuse the gas price and timestamp from the transaction records into a new weight and set appropriate return and exploration parameters to modulate the sampling tendency of random walk to characterize the money laundering nodes. We construct the dataset using real Ethereum data and evaluate the effectiveness of GTN2vec on the dataset by various classifiers such as random forest. The experimental results show that GTN2vec can accurately and effectively extract money laundering account features and significantly outperform other advanced graph embedding methods.

Keywords: money laundering detection; graph embedding; Ethereum

Citation: Liu, J.; Yin, C.; Wang, H.; Wu, X.; Lan, D.; Zhou, L.; Ge, C. Graph Embedding-Based Money Laundering Detection for Ethereum. *Electronics* **2023**, *12*, 3180. https://doi.org/10.3390/electronics12143180

Academic Editor: Andrei Kelarev

Received: 25 June 2023
Revised: 18 July 2023
Accepted: 20 July 2023
Published: 21 July 2023

1. Introduction

Blockchain is a distributed, open-source, immutable, public digital ledger shared among connected peers [1]. Each block contains details of transactions and assets (i.e., ether or bitcoin) exchanged between users. Ethereum is a special type of blockchain platform that includes more than just monetary transactions. Users can create smart contracts on it to run different types of applications [2]. The code for smart contracts is executed during the mining process, which makes the data structure of Ethereum more complex than other blockchains.

Ethereum [3], the second largest blockchain platform, has been worth nearly $100 billion since its launch in 2015 and is also the largest blockchain platform in support of smart contracts today. However, with its rapid development, several issues related to various cybercrimes, such as money laundering, phishing, bribery, and Ponzi schemes, have been rapidly increasing. Money laundering, in particular, has become an issue worrying the world. In cryptocurrencies like Ethereum, decentralization can easily resist control and censorship [2]. Transactions are difficult to link to real people because of their anonymity, which makes them highly attractive to money launderers. Although governments and

international organizations have implemented some strict anti-money laundering (AML) regulations, the problem of money laundering on Ethereum is still severe.

As of 17 August 2022, Tornado Cash received 74.7% of the total amount laundered on the Ethereum network, up to ETH 300,160, with another approximately 24% still in the hackers' wallets, while 1.5% was sent to trading platforms, according to a report by SlowMist [4]. The FBI announced on its website [5] that the North Korean hacker group Lazarus Group and APT38 were the attackers of Harmony Bridge, a hacker group that used malware called TraderTraitor and a privacy protocol called Railgun to launder more than USD 60 million stolen from Ethereum on 23 June 2022. Such money laundering cases have increased in recent years. Therefore, it is essential to investigate more effective detection methods for money laundering crimes for Ethereum.

Malicious behavior with Ethereum includes attacks that exploit vulnerabilities in smart contract code, such as honeypot contracts. These vulnerabilities in the code can be verified and analyzed using formal modeling methods [6,7]. As for other malicious behaviors, such as phishing scams, money laundering, gambling, Ponzi schemes, etc., researchers in previous studies usually detect them in one category. However, money laundering for Ethereum is significantly different from other malicious activities. Phishing scams for Ethereum, for example, typically employ emails and phishing websites to obtain sensitive information and money from users. Money laundering, on the other hand, is the process by which stolen money obtained by hacking and other means is laundered through layers and disguised as legitimate funds. The transaction patterns of money laundering accounts and phishing scam accounts are also very different in terms of money trajectories [8]. Therefore, we would like to subdivide these categories to more specifically characterize the money laundering accounts among them and design an algorithmic model specifically for money laundering detection. Since blockchain data are transparent and we can directly access all transaction data by synchronizing the full nodes, it is very intuitive to collect blockchain data and analyze the patterns in them. In order to accurately identify whether an Ethereum account is a money laundering account or a normal account, it is critical to extract features that accurately represent a money laundering address.

Ethereum's transaction records can be constructed as a high-dimensional graph of financial transactions. Analyzing the graph can help us to make good use of the information hidden in the graph. The graph embedding algorithm can convert a high-dimensional graph into a single or a set of low-dimensional vectors, preserving the structure and information of the graph, and the whole feature extraction process is automated [9]. Therefore, for data analysis in Ethereum, graph embedding is a more efficient method. In graph embedding, the weights on the edges can more accurately reflect the similarity between nodes. Therefore, we focus on the impact of the money launderer's behavioral patterns on the labels in the transaction records of Ethereum. The gas price in Ethereum transaction records can be set by the trader and affect the processing speed of the transaction, and money launderers usually want to disperse their funds as quickly as possible. The timestamp in the transaction records represents the time of each transaction and can reflect the degree of correlation between transactions. Therefore, we choose to enhance feature extraction by fusing the gas price with the timestamp as the weights of edges in the input graph.

In this paper, we first obtain the real transaction data and addresses with money laundering labels from authoritative Ethereum websites and construct a large-scale Ethereum transaction network. By merging the gas price and timestamp and considering the network's structural information, we propose a graph embedding algorithm based on a biased random walk called GTN2vec. We obtain the embeddings of nodes by GTN2vec and use various classifiers, such as random forest, for the classification task. Overall, the contributions of our paper are as follows:

- We propose a novel Ethereum money laundering-detection scheme based on an improved graph embedding-based algorithm called GTN2vec. To the best of our knowledge, it is the first algorithm dedicated to detecting Ethereum money laundering accounts precisely.

- For the first time, this algorithm uses gas price and timestamp in the Ethereum transaction network as auxiliary information for graph embedding. It comprehensively considers the behavioral patterns of money launderers and the structural information of the transaction network, which can accurately extract the characteristics of money laundering addresses on Ethereum.
- We obtain real money laundering addresses from Ethereum and construct the dataset. On the real dataset, we evaluate the effectiveness of GTN2vec. Experimental results show that the GTN2vec algorithm outperforms other advanced graph embedding methods.

The rest of this paper is organized as follows. Section 2 introduces our paper's background, including Ethereum, graph embedding algorithms, money laundering, and related works. Section 3 presents the technical details of the proposed overall detection framework and the GTN2vec embedding algorithm. Section 4 describes the experimental evaluation of the proposed approach's effectiveness in detecting Ethereum money laundering. Finally, we conclude the paper in Section 5.

2. Background and Related Works

2.1. Accounts and Transactions on Ethereum

There are two types of accounts on Ethereum, externally owned accounts (EOAs) and smart contract accounts. Each EOA has a private key and can send a message to another EOA or smart contract account by creating and signing a transaction using its private key. Smart contract accounts are automatically created by the EOA when the contract deploys. Smart contract accounts cannot initiate transactions but only passively trigger transactions upon receipt to execute pre-written smart contract code. Various operations on Ethereum consume gas, such as data storage, contract creation, and invocation. The amount of gas consumed to complete the operation and the current price of gas affect the transaction fee. Senders can also decide for themselves the maximum amount of gas they can consume per transaction and specify the gas price. The higher the gas price that the sender pays, the higher the priority of the transaction, as the miner can be paid more. By setting a lower gas price, the sender can save money, but the later the transaction becomes loaded into the block. The timestamp is generated from the time record and the hash value extracted from the block. It is present in the details of each transaction along with the gas price, which is a valid proof of the existence of each transaction containing time information.

2.2. Money Laundering

Even before cryptocurrencies, money laundering was a common financial crime. Criminals obtain funds through illegal channels and then inject them into the financial system as seemingly legitimate funds. Money laundering usually involves large sums of money, and criminals try various methods to evade scrutiny and law enforcement, some of the most basic of which are still used today. There are three distinct stages to the common money laundering model: placement, layering and integration [10]. In the placement phase, criminals like hackers introduce illicit funds into the financial system. In the layering phase, these illicit funds are transferred to different accounts and financial institutions as discretely as possible to hide the source. Finally, in the integration phase, the money is remitted to the criminals when the source of these funds looks legitimate.

Similar activity exists in cryptocurrencies such as Ethereum and is much more difficult to control. Despite calls for adopting a global AML regime for cryptocurrencies based on know-your-customer (KYC) and customer identification procedures (CIPs) to capture accurate customer information and prevent illicit financial activity, there is currently no uniform regime. AML regulatory regimes for cryptocurrencies vary widely from country to country [11]. Current AML systems are often designed with complex rules for money laundering activities and generate red alerts when suspicious financial transactions are identified. However, these rules can erroneously intercept many legitimate transactions, so a significant amount of human resources is devoted to verifying that red-alert transactions are involved in money laundering crimes [12].

2.3. Graph Embedding

Graphs in life are generally high-dimensional and challenging, so most graph analysis methods require significant space and time resources. Graph embedding can transform graph data into a low-dimensional space with full retention of the graph structure information and attributes. The output of graph embedding can help us to implement many applications, such as node classification, node clustering, and link prediction. With the popularity of graphs in various fields, research on graph embedding has started to use graphs as input and auxiliary information to facilitate embedding [9]. Existing approaches to network embedding mainly include factorization-based approaches [13], random walk-based approaches [14], and deep learning-based approaches [15]. What we are trying to implement is a node classification problem. An exemplary node embedding must preserve the graph's structure while focusing on node connections. Choosing the appropriate auxiliary information is the focus of this paper's research.

2.4. Related Work

2.4.1. Money Laundering Detection.

In the current regulatory research on money laundering crimes, some parts consider improving the existing AML system regarding the cost of review and validation. Ref. [12] proposed a machine learning classification model to reduce the false positives of AML rule-based systems, but it does not replace the rule-based system and is only used to handle alert events. The study explored anti-money laundering for traditional financial institutions, such as banks. Cryptocurrencies, such as Ethereum, are more popular with money launderers due to their decentralization and other characteristics, and the methods used to launder money are more flexible and therefore more difficult to control. On the other hand, some studies have considered using machine learning methods to identify illegal behaviors, such as money laundering. However, their detection targets are coarse grained and encompass many financial crimes. Ref. [16] used the XGboost classifier to detect malicious accounts based on Ethereum's transaction history. Ref. [17] used the LGBM method for detection, with a dataset consistent with [16], and their model achieved 98.60% accuracy. Ref. [18] used supervised learning methods to classify malicious and non-malicious addresses in Ethereum and found that linear and non-linear machine learning methods outperformed integrated learning methods for address classification. These studies do not separate the detection of various malicious activities. In contrast to these studies, we specialize in the detection of money laundering among them and instead of using traditional feature learning methods, we choose to use graph embedding techniques for the automatic extraction of features.

2.4.2. Anomaly Detection Based on Graph Embedding.

In recent years, there has been a large amount of research work applying graph embedding methods to anomaly detection on Ethereum. Ref. [19] constructed a heterogeneous graph transformer network (S_HGTNs) by extracting features, using the relationship obtained from the meta-paths learned from the network matrix as the input to the convolutional network, and finally classifying malicious smart contracts by node embedding. Ref. [20] proposed a biased random walk-based link prediction framework to study transaction tracking, demonstrating the impact of transaction frequency and amount on the evolution of the transaction network. Refs. [21,22] used different graph embedding methods to detect phishing scams on Ethereum, respectively. Ref. [21] used the Node2vec [23] method, and the absolute accuracy of the model was 84.6%. Ref. [22] proposed a new network embedding model, trans2vec, for detecting phishing scams on Ethereum by combining the amount and timestamp of transactions. Although these studies use graph embedding techniques, they target other financial crimes on Ethereum. The effectiveness of these methods in detecting money laundering is yet to be proven, considering that the behavioral patterns of various financial crimes are very different. Ref. [24] used the Metapath2vec graph embedding algorithm to compute a feature vector representation of

malicious nodes and performed a clustering analysis based on the feature vectors. However, the experimental results could not prove the effectiveness of the identification.

3. Methodology

In this section, we propose a graph embedding-based GTN2vec algorithm for money laundering detection.

3.1. Overview

As shown in Figure 1, the overall framework we use to detect money laundering is divided into three parts. First, we obtain the Ethereum real data to construct the graph dataset as an input to the graph embedding. In the graph, we denote addresses with money laundering markers in orange and normal addresses in blue. Using these addresses as source points, we obtain records of transactions between them and their adjacent second-order neighbors. Each edge contains the gas price and timestamp of that transaction, ultimately constructing a directed weighted graph. In the second part, we design a GTN2vec biased random walk strategy that can proportionally fuse the gas price and timestamp of the transaction edges into new weights and generate the transition probability of the walk in conjunction with the structure of the graph. Based on the transition probability, the algorithm obtains embedding vectors that represent the characteristics of each node. Ultimately we use the embedding vectors of the nodes for the classification task to identify the money laundering accounts. The details of our dataset construction approach and the rationale for the GTN2vec algorithm are described next.

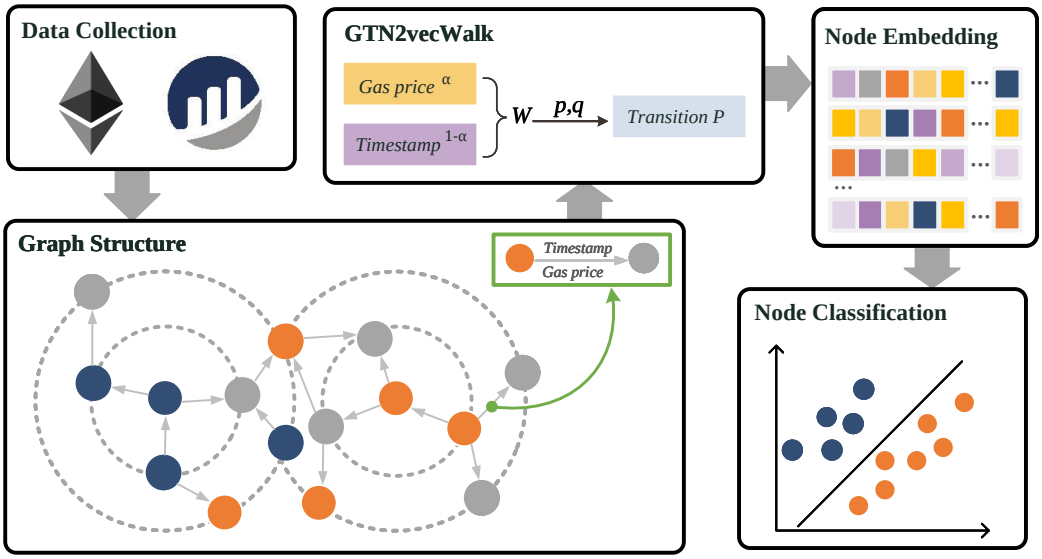

Figure 1. GTN2vec money laundering detection framework (author's own processing).

3.2. Ethereum Money Laundering Dataset

Money laundering on Ethereum is when hackers obtain large amounts of illicit funds after committing theft or fraud and launder the funds through layers of transfers. Obtaining account information and transaction records involved in money laundering crimes on Ethereum to construct graphs is a prerequisite for solving money laundering detection problems using graph analysis. In November 2019, the Upbit exchange was hacked to steal 342,000 ETH (USD 48.1 million).In the year following this highly influential case, hackers continued to launder the stolen funds by sending them to major exchanges through decentralized transfers and other means. The judiciary, blockchain security research organizations, and major exchanges joined forces to analyze the money laundering patterns and continuously track the flow of funds, which led to the freezing and recovery of some of the

funds. In this experiment, we use all the addresses that have been labeled as "Upbit Hack" in this case as money laundering accounts.

First, we obtained all transaction records on Ethereum by synchronizing all block data up to June 2022 through the Ethereum Geth client. Each transaction record contains the address of the sender and receiver of the transaction, the transaction hash, the transaction amount, the gas price, the timestamp, and other information. Our approach requires some of the information on the transaction records as an aid for the subsequent embedding phase.

Second, we obtained the addresses of 815 EOA and smart contract accounts tagged with the Upbit Hack, according to Etherscan (https://etherscan.io/ (accessed on 15 May 2022)), the authoritative Ethereum block browser. The detection of money laundering accounts in Ethereum can be modeled as a binary classification problem, so we randomly selected 815 addresses of accounts without money laundering labels as normal data from the span of blocks involved in the data with money laundering labels. To ensure that these non-money laundering accounts were normal enough, we further compared them to the 1259 fraud accounts in the Ethereum Fraud dataset [22]. In the end, we selected 815 accounts that did not have any negative labels.

Finally, we obtained the 2-order transaction records for 815 money laundering and 815 normal addresses. We represent each account address as a network node and each transaction between the accounts as an edge connecting the nodes. By taking the 1630 nodes as centroids, all transactions from each centroid to within its 2-order neighbors are tracked, constituting a large number of directed weighted subgraphs. After cleaning the duplicate and failed transactions, we combine all the subgraphs to construct a large directed weighted financial transaction graph as the required dataset for the experiment. The final dataset size is 45,585 nodes and 53,356 edges.

3.3. GTN2vec Algorithm

3.3.1. Problem Definition

The gas price is the cost of transaction execution on the Ethereum network, which determines the total fee for a given transaction and is related to transaction speed and trustworthiness. A high gas price indicates that the transaction may involve enormous amounts of money. Criminals are usually eager to send illegal funds to financial institutions. By setting a higher gas price, they can ensure that the transaction is prioritized. Moreover, in order to avoid supervision, money launderers usually split large sums of money into several small sums that do not attract attention, and disperse them. Therefore, we consider that the gas price can be used to distinguish money laundering activities from normal activities more prominently than the transaction amount. The timestamp contains the time information of each transaction, and time is also a basic factor in the transaction analysis. In the stage of the decentralized transfer of stolen money, criminals have a strategy for the transaction time and the choice of recipients. It is very important to understand the time pattern of funds in and out of the transfer. The fusion of gas price and timestamp as the weight of the input graph can be extremely effective in assisting the feature extraction of nodes in the graph embedding process. Therefore, we propose a graph embedding algorithm based on a biased random walk, which uses the gas price and time stamp to facilitate embedding while considering local and global information about the transaction network. Algorithm 1 shows the pseudocode of GTN2vec. We summarize the relevant notations in Table 1.

We denote the constructed directed weighted graph as $G = (V, E)$, where V is a node set, and E is an edge set. For each edge $(u, v) \in E$, $w(u, v)$ represents the gas price on the edge pointing from node u to node v, and $t(u, v)$ represents the timestamp on the edge pointing from node u to node v. Given a graph G and a dimension d, we aim to learn the mapping function ($f : V \to \mathbb{R}^{|V| \times d}$) from nodes to node embeddings while preserving as much information about the nodes as possible. Graph embedding is divided into two steps, and we first perform a graph-based random walk. Given a source node u, by sampling the neighborhoods of the nodes through a specific search strategy, information about the nodes

and the network structure is captured in the form of node sequences. The node sequences are then embedded using the Skip-gram architecture. Skip-gram is a Word2vec model that predicts the context from the target vocabulary. Here, we input the node sequences into the model, and by maximizing the probability of predicting neighboring nodes, Skip-gram can train and learn the prediction of neighboring nodes. The sampling strategy we devised during the random walk phase is described next.

Table 1. Summary of notations (author's own processing).

Notation	Meaning
ρ	The transition probability from the current node to the neighboring nodes
G'	A transaction graph weighted by the transition probability
$paths$	A set of r random walk sequences of length l
$path$	A sequence of random walks of length l
$node_{cur}$	The current node during random walks
$V_{curnode}$	The set of neighboring nodes of the current node
s	The next node in the random walk sampling
f	The d-dimensional vector representation of each node

Algorithm 1 The GTN2vec algorithm.

LearnFeatures (Transaction graph $G = (V, E, A)$ where A includes the gas price and timestamp on each edge, dimentions d, walks per node r, path length l, context size c, return p, exploration q, balance α)
$\rho = \text{TransitionProbability}(G, p, q, \alpha)$
$G' = (V, E, \rho)$
Initialize $paths$ to $\emptyset$
for $m = 1$ to r **do**
  for each node $x \in V$ **do**
    $path = \text{GTN2vecWalk}(G', x, l)$
    Append $path$ to $paths$
$f = \text{StochasticGradientDescent}(c, d, paths)$
return f

GTN2vecWalk (Graph $G' = (V, E, \rho)$, Start node x, Path length l)
Initialize $path$ to $[x]$
for $path\_m = 1$ to l **do**
  $node_{cur} = path[-1]$
  $V_{curnode} = \text{GetNodeNeighbors}(node_{cur}, G')$
  $s = \text{AliasSample}(V_{curnode}, \rho)$
  Append s to $path$
return $path$

3.3.2. Random Walk

Given a source node u, walk l steps from u, selecting its neighbor nodes each time according to a specific transition probability. Finally, a sequence of nodes of length l is generated. Specifically, let the ith node of the sequence be b_i, and the probability of selecting a given neighbor node x as b_i starting from $b_{i-1} = u$ is

$$P(b_i = x | b_{i-1} = u) = \begin{cases} \frac{\rho_{ux}}{Z}, & \text{if } (u, x) \in E \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where ρ_{ux} is the transition probability of node u to x. We normalize it by the constant Z.

3.3.3. Search Strategy

The critical issue in the random walk process is whether the sampled node neighborhoods are comprehensive enough to preserve the graph properties and accurate node

characteristics. Among several current random walk-based graph embedding algorithms, DeepWalk [14] was the first to use a random walk mechanism for each node. The transition probability from the source node to each neighbor is the same. The random sampling of neighboring nodes is repeated until a sequence of nodes of length l is found. Node2vec [23] defines two parameters based on DeepWalk to make the random walk have the properties of both depth-first sampling (DFS) and breadth-first sampling (BFS) by adjusting the transition probability between nodes.

When detecting money laundering nodes, it is not enough to use the general graph embedding method. For a complex financial transaction network such as Ethereum, the information on the transaction edges cannot be ignored, and choosing the correct information can better characterize the target node. Therefore, we use the critical information of the gas price and timestamp as the new attributes affecting the embedding for the first time based on Node2vec. During the random walk, these new attributes can cause the neighboring node that is more closely connected to the source node to be selected as the next node.

We define three parameters in the algorithm, the return parameter p, the exploration parameter q, and the balance parameter α. In a random walk, the size of parameter p determines whether to return to the previous node in the next step. The parameter q determines whether the neighborhood sampling is closer to BFS or DFS. If $q > 1$, then the random walk favors BFS and biases the selection of the surrounding nodes. If $q < 1$, the random walk tends to be DFS, preferring to visit distant nodes. The parameter α balances the gas price and timestamp weight in the transition probability, respectively. With these three parameters, we comprehensively consider the key factors that may affect the embedding of money laundering nodes. Specifically, as shown in Figure 2, in a random walk, suppose the current node is u, which has just been transferred from node m and now needs to decide the next step. We denote V_u as the set of all neighboring nodes that can be reached directly from node u. The transition probability ρ_{ux} from node u to its neighboring node $x \in V_u$ is given by

$$\rho_{ux} = Pwt_{ux} \cdot \beta_{pq}(m, x) \tag{2}$$

Using the parameter α, we fuse the gas price and timestamp on edge (u, x) starting from u in the most appropriate ratio. We denote $w(u, x)$ and $t(u, x)$ as the gas price and timestamp of the latest transaction from node u to x. The fused weight Pwt_{ux} can be calculated as follows:

$$Pwt_{ux} = Pw_{ux}^{\alpha} \cdot Pt_{ux}^{1-\alpha} \tag{3}$$

$$Pw_{ux} = \frac{w(u, x)}{\sum\limits_{x' \in V_u} w(u, x')} \tag{4}$$

$$Pt_{ux} = \frac{t(u, x)}{\sum\limits_{x' \in V_u} t(u, x')} \tag{5}$$

Depending on the shortest distance from node m to x, we use the parameters p and q further to adapt the search process between DFS and BFS to obtain information about the graph's structure:

$$\beta_{pq}(m, x) = \begin{cases} \frac{1}{p}, & \text{if } d_{mx} = 0 \\ 1, & \text{if } d_{mx} = 1 \\ \frac{1}{q}, & \text{if } d_{mx} = 2 \end{cases} \tag{6}$$

Thus, our approach is to first perform r random walks of walk length l from each source node, where each step is chosen based on the transition probability ρ_{ux}, which takes

into account the gas price, timestamp, and structural information of the graph, and then perform node representation learning using the Skip-gram method of Word2vec.

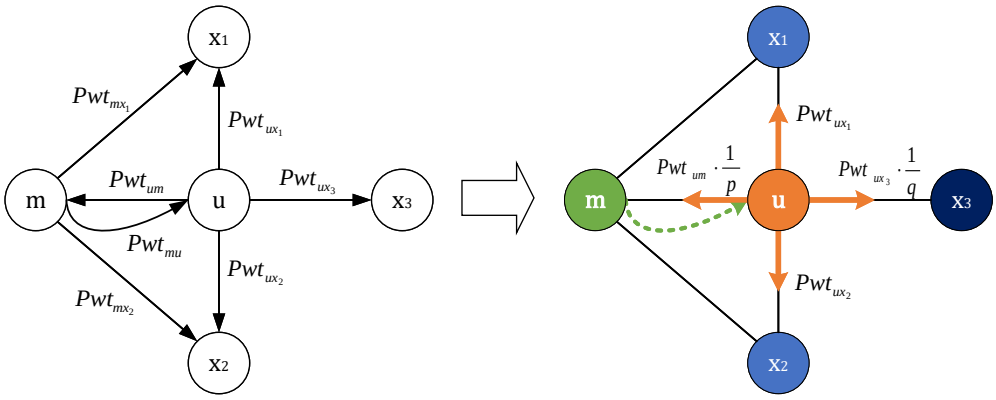

Figure 2. Illustration of the random walk procedure in GTN2vec (author's own processing).

4. Experiment

We experimentally demonstrate the effectiveness of this algorithm in this section. All experiments are run on a desktop computer configured with Intel Core i5-9400 CPU @ 2.90 GHz and 8 GB RAM. The operating system is Windows 10, and the programming language is Python 3.7.0.

4.1. Dataset

As stated in Section 3, in order to perform the node embedding and downstream node classification tasks, we need to construct an input graph. First, we write scripts to collect second-order transaction records for each of the 1630 addresses through the API provided by etherscan for developers and saved them as CSV files. We then merge these transaction records into a single CSV file, removing duplicate transactions, failed transactions, and handling nulls. We keep only the sender address, receiver address, gas price, and timestamp of each transaction. Finally, the directed weighted graph is constructed through Networkx. The dataset contains 45,585 nodes and 53,356 edges. Among them, money laundering addresses are labeled as 1, and normal addresses are labeled as 0. We divide the dataset with 80% as the training data and the remaining part as the test data. Considering the training speed for large-scale data, we finally choose the random forest classifier.

4.2. Baseline Method

Our experiments evaluate the learning effectiveness of the GTN2vec algorithm for money laundering node features through a node classification task and compare the following two graph embedding algorithms as a baseline. The sampling strategy of DeepWalk, in which the transition probability from the source node to all neighboring nodes is equal, is more random and does not pay special attention to the network structure. Node2vec builds on DeepWalk by adding p and q parameters to capture the local and global network structures. However, it does not pay attention to the impact of auxiliary information on the embedding performance in a particular network.

We want to focus on comparing the effect of the random walk strategy of different embedding methods on the characterization of money laundering nodes. Therefore, to be as fair as possible, we input the same dataset and basic parameters for each algorithm in the sampling phase so that they capture the same length of node sequences and the same number of iterations per walk. Specifically, the basic parameters are set to $d = 128$, $r = 80$, $l = 80$, and $c = 10$. For GTN2vec and Node2vec, which further consider the network structure, we uniformly set $p = 1$ and $q = 0.8$. For the balance parameter α in GTN2vec, we set $\alpha = 0.7$. For each algorithm, we choose the random forest classifier with a number of estimators ranging from 120 to 1200 and a maximum depth from 5 to 30. The hyperparameters for

the best performance are obtained by using a grid search with 5-fold cross validation and repeated 50 times, and we compare the average results.

Depending on the true category of the sample and the recognition by the classifier, there are four cases as follows: true positive—the true category is positive and is recognized as positive by the classifier; true negative—the true category is positive but is recognized as negative by the classifier; false positive—the true category is negative but is recognized as positive by the classifier; and false negative—the true category is negative and is recognized as positive by the classifier. We judge our classification effectiveness by four evaluation metrics, i.e., precision, recall, F1-score, and accuracy. The four metrics are defined as follows:

$$\begin{aligned} \text{Precision} &= \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \\ \text{Recall} &= \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \\ \text{F1-score} &= 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \\ \text{Accuracy} &= \frac{\text{true positive} + \text{true negative}}{\text{Total}} \end{aligned}$$

4.3. Classification Performance

We compare the classification performance of different graph embedding algorithms as shown in Table 2. The experimental results show that the GTN2vec algorithm we design has the best Ethereum money laundering detection results, with an average accuracy of 95.7%. All three graph embedding algorithms perform well in the classification task, indicating that the graph embedding technique is well suited to deal with the anomaly detection problem in financial networks such as Ethereum. Among them, Node2vec slightly outperforms DeepWalk, indicating that careful consideration of the local and global structure of the network can effectively improve the embedding performance. GTN2vec significantly improves over Node2vec, indicating that the gas price and timestamp can significantly improve the characterization of money laundering nodes. Therefore, it can be concluded that the network structure and specific auxiliary information can effectively help capture the characteristics of target nodes in money laundering detection. Moreover, the experimental results also verify our inference that gas price and timestamp can indeed reflect the specific behavioral patterns of money laundering addresses to a large extent.

Table 2. Classification performance of several graph embedding algorithms (author's own processing).

Method	Accuracy	Precision	Recall	F1-Score
DeepWalk [14]	0.939	0.951	0.929	0.940
Node2vec [23]	0.942	0.946	0.940	0.943
GTN2vec (Ours)	0.957	0.953	0.964	0.959

Considering the performance of the classifier will also have some impact on the experiment. In addition to random forest, we select several commonly used classifiers for comparisons, such as logistic regression, SVM, XGBoost, and naive Bayes. The experimental results are shown in Table 3, and it can be observed that our proposed GTN2vec algorithm performs very well with different classifiers, which further proves that GTN2vec is very comprehensive in extracting the features of money laundering nodes. In addition, we experiment with DeepWalk and Node2vec on different classifiers and find that random forest is the best choice for each graph embedding algorithm.

Table 3. Performance of GTN2vec with multiple classifiers (author's own processing).

Classifier	Accuracy	Precision	Recall	F1-Score
Naive Bayes	0.893	0.889	0.905	0.897
XGBoost	0.917	0.917	0.923	0.920
Logistic regression	0.929	0.905	0.964	0.934
SVM	0.936	0.915	0.964	0.939
Random forest	0.957	0.953	0.964	0.959

4.4. Parameter Sensitivity

We investigate the effect of each parameter in the GTN2vec algorithm on the results. To fully account for the accuracy and recall of the algorithm, we choose the F1-score as the measure. For each parameter tested, default values are used for all other parameters. The effect of each parameter on the F1-score is shown in Figure 3. We observe that increasing the dimension d and the number of walks r can significantly improve the performance of GTN2vec because higher-dimensional feature vectors can retain more complex node information, and a larger number of walks can reduce randomness and retain more realistic relationships between nodes. The length l of the node sequence has less impact on the performance of GTN2vec; we speculate that l is already much larger than the number of neighbors that some nodes can access, but a longer sequence still improves the representation of the nodes.

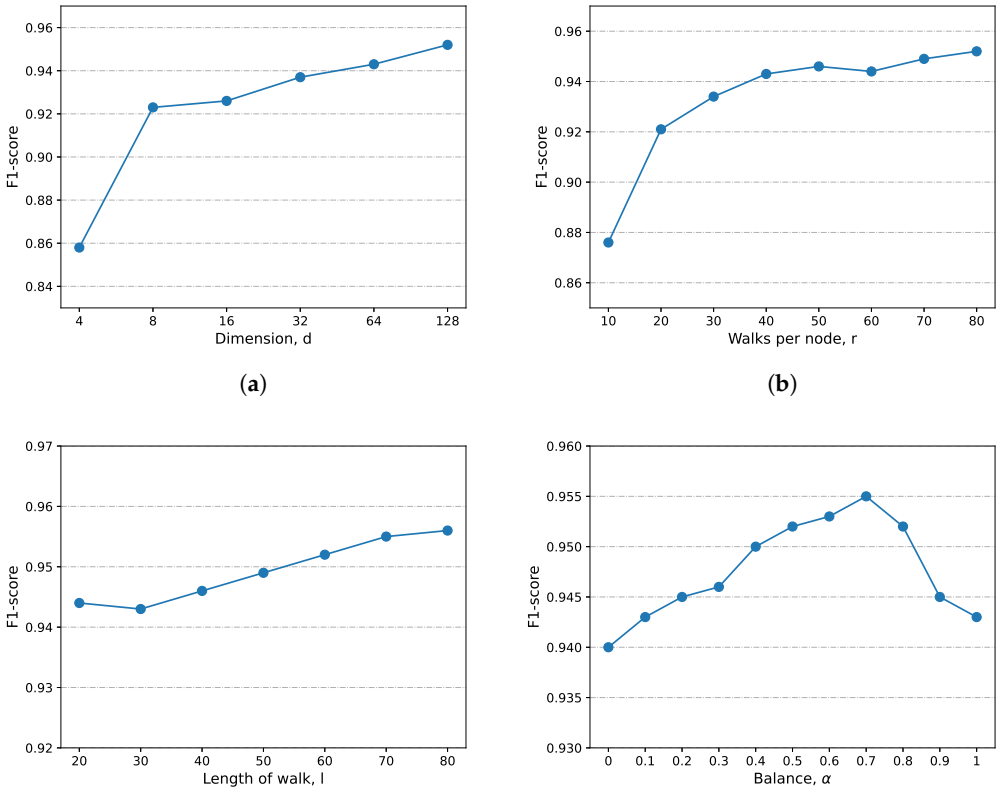

Figure 3. Effect of each parameter of GTN2vec on classification results (author's own processing). (**a**) Dimension, d. (**b**) Walks per node, r. (**c**) Length of walk, l. (**d**) Balance, α.

For the balance parameter α, if $\alpha = 0$, it means only the timestamp on the connection edge, and conversely, if $\alpha = 1$, it means that only the gas price is influential. We adjust α from 0 to 1, explore its effect on the F1-score through lots of experiments, and finally find that the detection effect is optimal when $\alpha = 0.7$, which is significantly better than both gas price only and timestamp only. Therefore, the combination of the gas price and timestamp

is helpful for feature extraction, where the gas price can facilitate embedding more than the timestamp.

For the values of the return parameter p and the exploration parameter q, we first test the effect of a single variable on the experiment. For the parameter p, the results indicate that GTN2vec performs better when $p \geq 1$. In contrast, the exact opposite is true for q. GTN2vec performance is higher when $q < 1$. These results indicate that the optimal strategy is not to return to the previous node and explore further away during the random walk. We also conduct experiments with different combinations, select several representative combinations as shown in Table 4 and find the best results obtained when $p = 1, q = 0.8$.

Table 4. F1-score performance of several representative combinations of return parameter p and exploration parameter q in GTN2vec (author's own processing).

Return-p	Exploration-q	F1-Score
0.5	2	0.934
0.5	1	0.943
0.8	1	0.940
1	1	0.953
1	0.8	0.956
1	0.5	0.946
2	0.5	0.950

4.5. Discussion

In our experiments, we trained the algorithmic model using data labeled with money laundering and achieved 95.7% recognition accuracy on the test set. Moreover, our algorithm has good generalization ability and can adapt well to fresh data, so it has high usability. In practical applications, the GTN2vec model can be utilized for suspicious address identification services. Since all the transaction data of Ethereum are open and transparent, it can obtain the transaction records of the target address through the API of the etherscan website, and retain the gas price and timestamp information in the records. The list of transactions is fed into the algorithmic model to be able to obtain the identification results.

The anti-money laundering systems currently used by financial institutions, such as cryptocurrency exchanges, are typically strict rule-based systems. The system continuously monitors each transaction through complex rules and generates timely alerts for suspicious transactions, such as sudden large fund transfers. However, such systems can incorrectly raise an alert on a large number of normal transactions, with a false positive rate typically around 95–98%. Some financial institutions arrange for 5000 or more employees to verify each alerted transaction [25]. This model is very inefficient and wastes a lot of human resources. Our model can further identify money laundering on the accounts involved in the alerts after they are generated by the system. By helping regulators filter out false alerts and pinpoint suspicious accounts, we can improve the efficiency of later reviews.

5. Conclusions

In this paper, we propose a graph embedding algorithm GTN2vec for money laundering detection in Ethereum by fusing the gas price and timestamp of transactions as auxiliary information for the first time, taking into account the local and global structure of the graph. We first obtain real money laundering addresses from Ethereum and construct the dataset. Then, the money laundering nodes are embedded by GTN2vec to generate vectors that capture the node information and network structure. Finally, the effectiveness of node embedding is verified by classifiers such as random forest. The experimental results show that GTN2vec can accurately and effectively detect money laundering nodes on Ethereum with an average accuracy of 95.7%, which is better than other graph embedding methods.

Existing research has paid little attention to money laundering in Ethereum, which makes our work lack a directly comparable reference. Moreover, our work may have little

ability to harness future changes, such as the Ethereum upgrades. In our future work, we will study more deeply the association between money laundering and other potential features in the transaction records to further improve the user profile of money launderers and increase identification accuracy.

Author Contributions: Conceptualization, J.L.; Data curation, J.L.; Formal analysis, J.L.; Investigation, J.L.; Methodology, J.L.; Project administration, C.G.; Software, J.L.; Supervision, C.Y., H.W. and C.G.; Visualization, J.L.; Writing—original draft, J.L.; Writing—review and editing, C.Y., H.W., X.W., D.L., L.Z. and C.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Key R&D Program of China (2020YFB1005902), the National Natural Science Foundation of China (62032025, 62071222, U20A20176), the Key R&D Program of Guangdong Province (2020B0101090002), the Natural Science Foundation of Jiangsu Province (BK20200418, BE2020106), the Guangdong Basic and Applied Basic Research Foundation (2021A1515012650), and the Shenzhen Science and Technology Program (JCYJ20210324134810028, JCYJ20210324134408023).

Data Availability Statement: Databases and source codes are available at: https://github.com/GTN2vec/GTN2vec (accessed on 7 April 2023).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Bus. Rev.* **2008**, 21260. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 24 June 2023).
2. Bjelajac, Ž.; Bajac, M. Blockchain technology and money laundering. *Pravo-Teor. Praksa* **2022**, *39*, 21–38. [CrossRef]
3. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
4. 2022 Blockchain Security and AML Analysis Annual Report (CN). Available online: https://www.slowmist.com/report/2022-Blockchain-Security-and-AML-Analysis-Annual-Report(CN).pdf (accessed on 6 February 2023).
5. North Korea-Linked Hackers Behind $100 Million Crypto Heist, FBI Says. Available online: https://www.cnbc.com/2023/01/24/north-korea-linked-hackers-behind-100-million-crypto-heist-fbi-says.html (accessed on 8 February 2023).
6. Krichen, M.; Lahami, M.; Al-Haija, Q.A. Formal methods for the verification of smart contracts: A review. In Proceedings of the 2022 15th International Conference on Security of Information and Networks (SIN), Sousse, Tunisia, 11–13 November 2022; pp. 1–8.
7. Abdellatif, T.; Brousmiche, K.L. Formal verification of smart contracts based on users and blockchain behaviors models. In Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 26–28 February 2018; pp. 1–5.
8. Lal, B.; Agarwal, R.; Shukla, S.K. Understanding Money Trails of Suspicious Activities in a cryptocurrency-based Blockchain. *arXiv* **2021**, arXiv:2108.11818.
9. Cai, H.; Zheng, V.W.; Chang, K.C.C. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 1616–1637. [CrossRef]
10. Kolachala, K.; Simsek, E.; Ababneh, M.; Vishwanathan, R. SoK: Money laundering in cryptocurrencies. In Proceedings of the 16th International Conference on Availability, Reliability and Security, Vienna, Austria, 17–20 August 2021; pp. 1–10.
11. Holman, D.; Stettner, B. Anti-money laundering regulation of cryptocurrency: US and global approaches. In *ICLG Anti-Money Laundering*; Global Legal Group: London, UK, 2018 ; pp. 26–39.
12. Eddin, A.N.; Bono, J.; Aparício, D.; Polido, D.; Ascensao, J.T.; Bizarro, P.; Ribeiro, P. Anti-Money Laundering Alert Optimization Using Machine Learning with Graphs. *arXiv* **2021**, arXiv:2112.07508.
13. Belkin, M.; Niyogi, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Adv. Neural Inf. Process. Syst.* **2001**, *14*, 585–591 .
14. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 701–710.
15. Cao, S.; Lu, W.; Xu, Q. Deep neural networks for learning graph representations. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
16. Farrugia, S.; Ellul, J.; Azzopardi, G. Detection of illicit accounts over the Ethereum blockchain. *Expert Syst. Appl.* **2020**, *150*, 113318. [CrossRef]
17. Aziz, R.M.; Baluch, M.F.; Patel, S.; Ganie, A.H. LGBM: A machine learning approach for Ethereum fraud detection. *Int. J. Inf. Technol.* **2022**, *14*, 3321–3331. [CrossRef]
18. Saxena, R.; Arora, D.; Nagar, V. Classifying Transactional Addresses using Supervised Learning Approaches over Ethereum Blockchain. *Procedia Comput. Sci.* **2023**, *218*, 2018–2025. [CrossRef]

19. Liu, L.; Tsai, W.T.; Bhuiyan, M.Z.A.; Peng, H.; Liu, M. Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum. *Future Gener. Comput. Syst.* **2022**, *128*, 158–166. [CrossRef]
20. Lin, D.; Wu, J.; Xuan, Q.; Chi, K.T. Ethereum transaction tracking: Inferring evolution of transaction networks via link prediction. *Phys. A Stat. Mech. Its Appl.* **2022**, *600*, 127504. [CrossRef]
21. Yuan, Q.; Huang, B.; Zhang, J.; Wu, J.; Zhang, H.; Zhang, X. Detecting phishing scams on ethereum based on transaction records. In Proceedings of the 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain, 12–14 October 2020; pp. 1–5.
22. Wu, J.; Yuan, Q.; Lin, D.; You, W.; Chen, W.; Chen, C.; Zheng, Z. Who are the phishers? Phishing scam detection on ethereum via network embedding. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *52*, 1156–1166. [CrossRef]
23. Grover, A.; Leskovec, J. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.
24. Sun, H.; Ruan, N.; Liu, H. Ethereum analysis via node clustering. In Proceedings of the Network and System Security: 13th International Conference, NSS 2019, Sapporo, Japan, 15–18 December 2019; Proceedings 13; Springer: Berlin/Heidelberg, Germany, 2019; pp. 114–129.
25. Lannoo, K.; Parlour, R. Anti-Money Laundering in the EU: Time to get serious. *Tech. Rep. Cent. Eur. Policy Stud.* **2021**, 31980 . Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3805607# (accessed on 24 June 2023).