Wang *et al. Journal of Cloud Computing* (2023) 12:88
https://doi.org/10.1186/s13677-023-00466-y

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Ensuring security in edge computing through effective blockchain node detection

Shenqiang Wang[1], Zhaowei Liu[1*], Haiyang Wang[2] and Jianping Wang[3]

Abstract

The rapid development of blockchain technology has garnered increasing attention, particularly in the field of edge computing. It has become a significant subject of research in this area due to its ability to protect the privacy of data. Despite the advantages that blockchain technology offers, there are also security threats that must be addressed. Attackers may manipulate certain nodes in the blockchain network, which can result in tampering with transaction records or other malicious activities. Moreover, the creation of a large number of false nodes can be utilized to gain control and manipulate transaction records of the blockchain network, which can compromise the reliability and security of edge computing. This paper proposes a blockchain node detection method named T^2A2vec that provides a more secure, credible, and reliable solution to address these challenges. In order to achieve T^2A2vec, a transaction dataset that is evenly distributed in both space and time was collected. The transaction dataset is constructed as a transaction graph, where nodes represent accounts and edges describe transactions. BP neural network is used to extract account features, and a random walk strategy based on transaction time, type, and amount is used to extract transaction features. The obtained account features and transaction features are fused to obtain account representation. Finally, the obtained node representation is fed into different classifiers to identify malicious nodes.

Keywords Blockchain, Graph Embedding, Security, Edge Computing

Introduction

Edge computing is a novel computing paradigm that distributes computing and storage resources to the edge of the network, providing significant advantages for various application scenarios, such as the Internet of Things [1–3], smart city [4, 5] and industrial Internet [6–8]. With the significant growth in the number of devices connected to the Internet of Things, the devices generate massive data at the network's edge, making the protection of private data a critical task of edge computing [9–11]. However, traditional privacy protection methods require centralized control, which is not feasible due to the distributed nature of devices within the edge computing environment. Thus, achieving decentralized privacy protection in edge computing is a challenging task. Nonetheless, blockchain technology's decentralized characteristics suggest a promising avenue for privacy protection in edge computing [12].

Blockchain, as a distributed ledger technology, can enable secure data exchange and sharing in an untrusted environment, thereby effectively protecting data privacy and security [13, 14]. Given its significant potential in ensuring data privacy protection, blockchain technology has been applied to edge computing as a solution for data privacy protection. Some studies have focused on how to secure data privacy and address the security issues of edge computing by applying blockchain solutions. In edge computing, devices share data and computing resources by connecting to cloud servers. However, traditional centralized storage methods no longer meet the

*Correspondence:
Zhaowei Liu
lzw@ytu.edu.cn
[1] School of Computer Science and Control Engineering, Yantai University, Yantai, China
[2] Institute of Network Technology Yantai, Yantai, China
[3] Shandong marine resources and environment Research Institute, Yantai, China

requirements due to the sensitivity, privacy and unbalanced data distribution. Liang et al. [15] Therefore, different researchers have proposed blockchain-based privacy protection solutions as an alternative. Encrypted technology and smart contracts have been employed to achieve data sharing and access control [16], while privacy protection and anonymity have been achieved through technologies such as zero-knowledge proof [17]. In addition, some studies have also proposed blockchain-based data ownership and license proof mechanisms to protect data privacy and ensure data security [18].

However, there are also security threats to using blockchain in edge computing. Attackers may manipulate certain nodes in the blockchain network, thus tampering with transaction records or performing other improper operations. This type of attack may result in incorrect transactions or unauthorized operations, thus affecting the reliability and security of edge computing [19]. Attackers may also attempt to steal data from the blockchain network, thereby compromising the privacy of edge computing. Such attacks may lead to data breaches and privacy threats, thereby affecting the security and reliability of edge computing. Attackers may attack the blockchain network by creating a large number of fake nodes in order to gain control and manipulate transaction records [20]. This type of attack may result in an imbalance in the number and capabilities of nodes, thus affecting the security and stability of edge computing [21].

In view of the aforementioned security issues, the rapid and accurate identification of various accounts on blockchain platforms has become a hot topic in the field of blockchain security. Networks are a common language for describing interactive systems in the real world, and network representation learning is widely regarded as an effective tool for analyzing network systems [22, 23]. The current method for detecting illegal accounts on the blockchain is to learn the network representation of nodes and use methods such as node classification and link prediction to complete illegal account detection. These methods can be divided into three categories. One is to extract manual features based on transaction history rules or extract statistical features through automatic feature construction tools, and combine the extracted features with traditional machine learning methods to complete the detection of illegal accounts in blockchain. But this approach requires considerable manpower, and the use of experts to exhaustively list all the relevant characteristics, in the face of an almost infinite number of blockchain addresses, cannot rely on the manual way to mark all the addresses, and this marking cannot guarantee realtime results, for example, the exchange will occasionally change their own address, illegal activities often change their own address to avoid crackdown, this time you need to go to manually update. Another approach is to use the random walk method to mine deep features from the blockchain transaction network. The last approach is to apply graph neural networks, a type of neural network specifically designed to process graphs, to automatically learn representations from the blockchain transaction network. While each method has its advantages and limitations, the use of random walk and graph neural networks shows promising results for accurate and efficient detection of illegal accounts on blockchain platforms.

Despite significant progress in detecting blockchain nodes, two critical issues remain unresolved. Firstly, existing methods suffer from an unbalanced spatial and temporal distribution of datasets, leading to a biased collection of various account types over time. Consequently, such biases can result in incomplete or inaccurate conclusions, distorting understanding of the network's behavior. Secondly, current methods lack transaction information, which is crucial in accurately representing a node's characteristics. Without this information, these methods cannot capture the nuances and complexities of the network, leading to suboptimal node representations. Addressing these two challenges is critical in ensuring a comprehensive understanding of blockchain nodes. It requires developing novel data collection methods that consider spatial-temporal distribution and incorporate transaction information and account characteristics. By optimizing node representations, it is possible to obtain more precise and dependable information about the blockchain network, resulting in an improved comprehension of its intricacies.

In order to solve these challenges, a novel method of blockchain node behavior detection is proposed that integrates the detection of account features, transaction **T**ime, transaction **T**ype and transaction **A**mount, called T^2A2vec. This paper presents a novel method for identifying illicit nodes in the blockchain network. A transaction network is constructed by collecting a dataset of blockchain transactions that are uniformly distributed across space and time. In this network, nodes represent accounts, and edges describe transactions. To generate node representations, a random walk strategy is utilized that takes into account transaction time, type, and amount. The resulting node representations are then input into different classifiers, and extensive experiments on the collected datasets validate the effectiveness of the algorithm. The method enables the identification and evaluation of transaction risk based on transaction type, providing users with the ability to make informed decisions and prevent fraud. The main contributions of the paper are as follows:

- This paper proposes a method to address the issue of uneven spatiotemporal distribution of blockchain labels. The proposed method captures transaction data based on time to obtain a blockchain transactions dataset with a uniform distribution across space and time.
- To identify illicit accounts in the blockchain network, the paper introduces transaction feature extraction and account feature extraction components. These components use transaction information from the blockchain transaction network to generate valid node representations in the network.
- Extensive experiments were conducted on the blockchain dataset collected in this paper. The experimental results show a higher prediction accuracy than the traditional blockchain phishing node prediction method.

The remaining portion of this paper is organized as follows. Section Related Work summarizes related work. Section Method Introduction specifies the method proposed in this paper. Section Experiments conducts experimental evaluation and analysis. Section Conclusion summarizes the work of this paper and provides direction for future work.

Related work

This section discusses related research by academic researchers on nodes detection in blockchain. Due to the popularity and widespread use of cryptocurrencies such as Bitcoin and Ethereum, complex network analysis for cryptocurrencies and their security applications have very important research and application value.

Boosting algorithms based on machine learning can realize blockchain network analysis [24]. Farrugia et al. [25] proposed a method to detect illegal accounts in Ethereum using XGBoost classifier. They analyzed three features that have the greatest impact on the output of the illegal account detection model 'time difference between the first and last transaction', 'total Ether balance' and 'minimum value received', this method effectively used the features of Ethereum nodes but does not analyze the transaction information between nodes, which reduced the accuracy of prediction. Farimah et al. [26] proposed a framework to identify illegal entities in the Ethereum blockchain network, which has good performance in integrated learning methods including random deep forest, stacked classifier and AdaBoost, and can effectively detect illegal Ethereum entities. Zhang et al. [27] proposed a method for detecting Ethereum Ponzi schemes based on an improved LightGBM algorithm. Chen et al. [28] proposed a graph-based cascading feature extraction method based on transaction records by kind of light-GBM-based double sampling integrated framework identification can account. The analysis of blockchain network using the Boosting algorithm focuses on the node attribute information in the blockchain network while ignoring the transaction information between nodes.

In recent years, deep learning techniques have been introduced into graph representation learning due to their effectiveness in deep feature extraction, among which graph convolutional networks are more widely studied [29, 30]. Liu et al. [31] proposed a blockchain identity inference method based on graph convolutional networks. Different types of accounts are collected and node transaction characteristics were analyzed, and some enhancement methods were proposed. Weber et al. [32] proposed a bitcoin antimoney laundering method using graph convolutional networks, which provided a timeseries graph of over 200,000 bitcoin transactions and classifies illegal transactions using LR, RF, MLP, and variants of graph convolutional networks, and the results show the superiority of random forest. The analysis of blockchain networks using graph neural networks places greater emphasis on node features and requires a high-quality dataset.

Graph embedding is a process of steganography graph data into low-dimensional dense vectors, which can be an excellent solution to the problem that graph data is difficult to input into machine learning algorithms efficiently [33, 34]. The data in the blockchain contains multiple information with high dimensionality, and graph embedding can be an excellent solution to this problem. Yuan et al. [35] used node2vec for phishing node classification. Wu et al. [36] proposed a method to detect phishing scams by digging through the transaction records of Ethereum. This method extracted address features by proposing a new network embedding algorithm trans2vec, and then used One-Class SVM to classify Ethereum nodes into ordinary nodes and phishing nodes. Yuan et al. [37] used an improved Graph2Vec based implementation for classification prediction of the constructed transaction subgraphs. Lin et al. [38] proposed a temporal weighted multidigraph embedding method to analyze Ethereum transactions and perform node classification. Blockchain network analysis based on graph embedding emphasizes transaction information and ignores the attributes of illegal nodes, which reduces the prediction accuracy. The analysis of blockchain networks based on graph embedding places greater emphasis on transaction information while ignoring the attributes of illegal nodes.

Table 1 Notations

Symbol	Explanation
G	The blockchain transaction network.
V	The set of nodes.
E	The set of edges.
V_L	The set of nodes with attributes.
y_i	Embedding of the node v_i.
$\pi_{v,x}$	The unnormalized transition probability from node v to x.
$P_{A_{ux}}$	Transaction amount based random walk strategy.
$P_{T_{ux}}$	Transaction time-based random walk strategy.
$P_{E_{ux}}$	Transaction types based random walk strategy.
Z	The embedding results.

Method introduction

This section will discuss the blockchain node detection method based on graph representation learning. The overall framework is shown in Fig. 1, which is divided into three parts: data collection, accounts feature extraction and transaction feature extraction. The graph construction part mainly involves the transaction data collected from blockchain illegal accounts and normal accounts. In the account feature extraction part, statistical features of accounts are calculated and a BP neural network is used to extract the account features. In the transaction feature extraction part, transaction features are extracted using a random walk strategy based on transaction time, transaction type, and transaction amount. The obtained account features and transaction features are fused to obtain a blockchain account representation and used for illegal account identification. The main notations used in this paper are summarized in Table 1.

Data collection

The blockchain's transparency and openness make transaction data containing rich information and complete traces of financial activities available to the public. The most simplest way to obtain blockchain transaction data is to access the blockchain network and synchronize block data through blockchain clients (such as Bitcoin Core[1] and Geth[2]) in order to obtain the original data of the blockchain. Another way is to retrieve transaction data through the blockchain browser.

The problem of identifying addresses on the blockchain, studied in this article, can be modeled as a multi-class problem, requiring a classification model trained through supervised learning. To achieve this goal, sufficient labeled data is needed as the training set, as well as a large dataset as the basis for the research work. Only with enough sample data can the model learn the relationships between features within the data and ultimately achieve a better performance in classification.

Due to the lack of a dataset of illegal nodes in edge computing based on blockchain, this study chose to use the Ethereum transaction dataset to verify the effectiveness of the proposed method. Although there are certain differences between Ethereum transaction data and edge computing data, they share some similarities, including structural similarity, temporal similarity, massive scale, and heterogeneity [39, 40]. Firstly, both can be represented as graph structures. Secondly, since edge computing data typically includes the start and end times of the data, each transaction in Ethereum has a fixed transaction timestamp and also possesses temporal attributes. Thirdly, another characteristic of the data appearing simultaneously in edge computing is its large volume, which requires a considerable amount of storage space. According to statistics, the majority of the data in blockchain transactions are in the tens of millions or more. Lastly, due to the differences between the sending and receiving devices, most edge computing data is represented as heterogeneous data, while Ethereum transaction data includes user types such as traders and miners, as well as transaction types such as calling contract transfers, which also demonstrate heterogeneity. Therefore, the effectiveness of the proposed method is verified by accessing Ethereum transaction records and collecting the necessary information independently.

In this study, collected an Ethereum transaction dataset consisting of two parts: illegal and normal nodes, which illegal nodes represent the attacking nodes in the blockchain. This dataset provides the labeled data necessary for training the classification model and enables us to study the features that distinguish illicit addresses from non-illicit ones. Using this dataset, it is possible to accurately identify illegal addresses.

All illicit nodes were collected from Etherscan[3], a block explorer and analytics platform for Ethereum, a decentralized smart contracts platform. This website also provides Etherscan account addresses, which show not only the content of the scam but also the accounts suspected of being involved in the scam.

All normal nodes are obtained through Infura[4], an excellent open Ethereum node that provides a standard RPC API for developers to call. To address the issue of uneven distribution in the dataset, it is necessary to

[1] Bitcoin Core, https://bitcoincore.org/

[2] Geth, https://geth.ethereum.org/

[3] Etherscan labelcloud, https://etherscan.io/labelcloud

[4] Infura, https://infura.io/

first analyze the active dates of each illegal account and randomly extract Ethereum accounts based on their active dates before obtaining normal accounts. During the extraction process, to avoid having duplicate, non-unique, and smart contract accounts, they will be filtered out. A check is performed to determine if the collected normal accounts have been flagged on the Etherscan website. If a flagged address is found, it will be removed from consideration. Since most accounts on the blockchain are honest, the remaining unflagged accounts that were randomly selected are considered normal accounts. This method of random selection based on active dates helps solve the issue of uneven distribution in the data while improving the accuracy and stability of the illegal account identification algorithm.

Finally, the API provided by Etherscan was used to query the transaction information for each account, obtaining first-order transaction data for each account, which was then saved. The transaction timestamp, transaction amount, and transaction type were all considered as edge attributes, with transaction type being considered as the edge type. Transaction types include transfers, contract creation, and invocation contracts.

By performing the above operations, a comprehensive dataset was obtained. This dataset includes the relevant transaction information and node features for each account. This dataset enables us to accurately identify illegal addresses and gain a deeper understanding of the transaction patterns on the blockchain network.

Graph construction

Construct the blockchain transaction network as a multi-directed graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. The total number of accounts is $N = |V|$. Each node $v \in V$represents an externally owned accounts (EOA) or a contract account (CA). A node represents an Ethereum account, and the rest of this paper will use nodes and accounts interchangeably for representation.

The set V_L is the set of nodes with attributes. The set V_u is the set of nodes connected to node u. The set E contains edges, and each edge can be represented as a quintuplet, i.e., $E = \{(v_i, v_j, w, t, r) | v_i, v_j \in V, w \in \mathbb{R}^+ \cup 0, t \in \mathbb{Z}, r \in R\}$, where (v_i, v_j) denotes the transaction from v_i to v_j, w denotes the transaction amount, t denotes the timestamp of the transaction, and r denotes the transaction type. The final transaction network is shown in Fig. 2, where n_1, n_2, n_3 are EOA, n_4, n_5 are CA. Each account contains a feature vector, and each transaction contains three features: transaction time, transaction type, and transaction amount.

Table 2 Complete list of the 18 extracted features

	Feature	Description
1	NTS	The number of transactions of send.
2	max_VS	The maximum value of send.
3	min_VS	The minimum value of send.
4	TVS	The total value of send.
5	AVS	The average value of send.
6	avg_TIS	The average time interval of send.
7	NTR	The number of transactions of receive.
8	max_VR	The maximum value of receive.
9	min_VR	The minimum value of receive.
10	TVR	The total value of receive.
11	AVR	The average value of receive.
12	avg_TIR	The average time interval of receive.
13	TETF	The total ether transaction fee.
14	AETF	The average ether transaction fee.
15	TDFL	The time difference between the first and last.
16	TEB	The total ether balance.
17	UAS	The unique address of send.
18	UAR	The unique address of receive.

Account features extraction

In this paper, the account feature set is constructed based on the transaction history of the account. It includes data such as the number, value, and frequency of transactions that are easy to calculate. It further reveals the correlation between trading behavior and accounts to discover the variability of trading patterns among different accounts. In this paper, a total of 18 transaction features are extracted. The details are shown in Table 2. Some of the features are described as follows.

Number of Transactions of Send (NTS): the number of transactions sent from an account, NTS_i represents the number of transactions sent from account i.

Total Value of Send (TVS): the sum of the transaction values sent by the account, TVS_i represents the sum of the transaction values sent from account i.

Average Value of Send (AVS): represents the average value of transactions sent by an account, which can be calculated from the current account NTS and TVS, calculated as:

$$SAV_i = \frac{STV_i}{NTS_i} \tag{1}$$

where TVS_i represents the average value of transactions sent from account i.

Maximum Value of Send (max_VS) and Minimum Value of Send (min_VS), which represent the maximum and minimum time interval between two transactions for a given account, respectively. $T_{i,k}$ denotes the timestamp

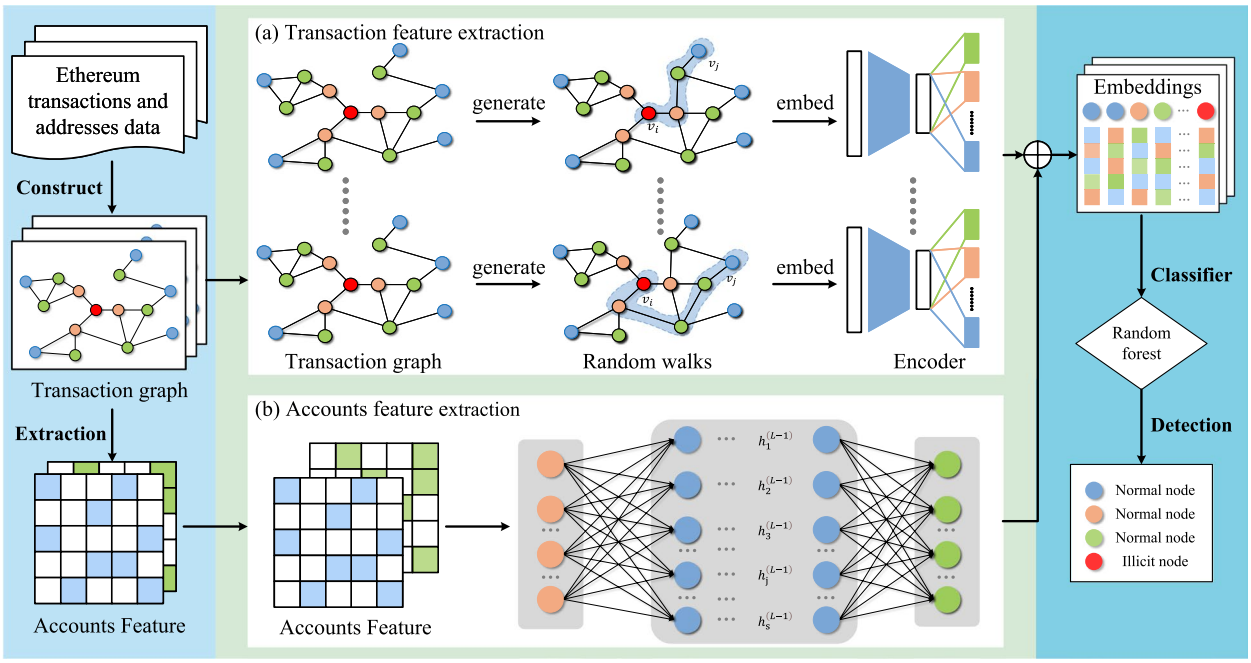

Fig. 1 The overall framework for account detection on the blockchain

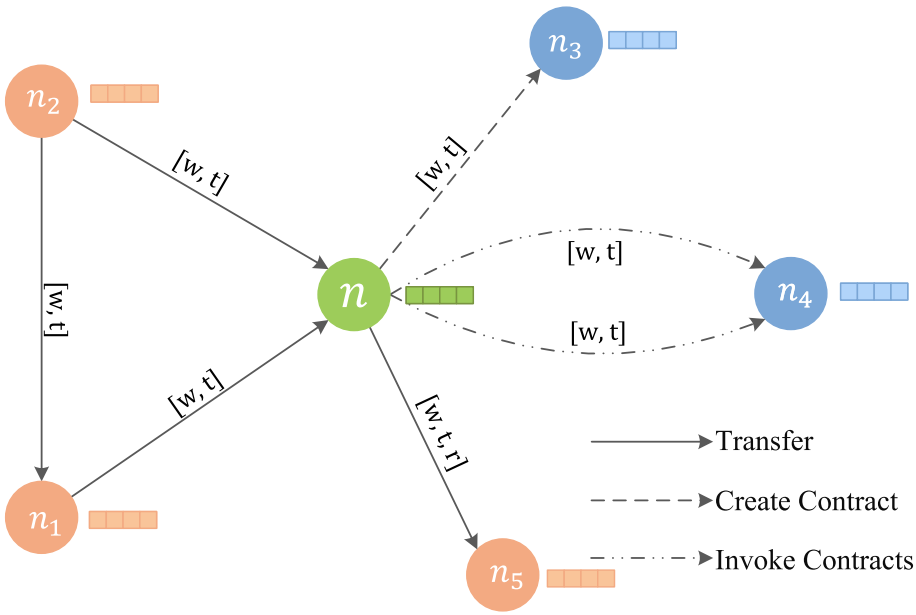

Fig. 2 The blockchain transaction network

of the k-th transaction sent by account i. The max_VS and min_VS are calculated as:

$$\max\_VS_i = \max_k \left(\left|T_{i,k} - T_{i,k+1}\right|\right) \tag{2}$$

$$\min\_VS_i = \min_k \left(\left|T_{i,k} - T_{i,k+1}\right|\right) \tag{3}$$

Average Time Interval of Send (avg_TIS): represents the average time interval of sending transactions for an account, which can be calculated from the time interval of each transaction and NTS. avg_TIS$_i$ represents the

average time interval of sending for account i, k is the total number of transactions for account i and is calculated as:

$$avg\_TIS_i = \frac{\sum_{j=1}^{k} T_{i,j+1} - T_{i,j}}{NTS_i} \tag{4}$$

Number of Transactions of Receive, Maximum Value of Receive, Minimum Value of Receive, Total Value of Receive, Average Value of Receive, Average Time Interval of Receive features are calculated in a manner similar to the features of sending transaction accounts, and are calculated as in Eqs. (1) to (4).

Total Ether Transaction Fee (TETF): the sum of transaction fees for each account, which can be calculated from the price of gas and gas used in the transaction, calculated as:

$$k = NTS_i + NTR_i \tag{5}$$

$$TETF_i = \sum_{j=1}^{k} \left(GU_{i,j} \times PG_{i,j}\right) \times 10^{-18} \tag{6}$$

where k is the number of transactions for the i-th account. $PG_{i,j}$ and $GU_{i,j}$ represent the price of gas and gas used in the j-th transaction for the i-th account, respectively. And uniformly convert Wei to Ether.

Average Ether Transaction Fee (AETF): the average of transaction fees for an account, which can be obtained from the TETF and the number of transactions, calculated as:

$$AETF_i = \frac{TETF_i}{k} \tag{7}$$

Total Ether Balance (TEB): the account balance calculated based on the collected transaction records, which can be obtained from the TVS and TVR, calculated as:

$$TEB_i = TVR_i - TVS_i \tag{8}$$

Each account possesses unique features, such as balance, transaction frequency, transaction type, and transaction amount. However, individual account characteristics alone are insufficient for identifying illegal accounts. Therefore, it is necessary to extract deeper information based on these features in addition to extracting account features. By extracting and analyzing account features, it is possible to more accurately identify illegal accounts.

To achieve this, after collecting account transaction features, a BP neural network is used to extract feature vectors from the collected data. Specifically, transaction features are normalized and passed to the BP neural network to learn the hidden relationships between accounts. The resulting account feature vector is represented as V_L, and the calculation formula is as follows:

$$V_L = f\left(W_{st} \bullet f_s + b_{st}\right) \tag{9}$$

where W_{st} is the weight matrix of the BP neural network, f_s is the feature vectors from the collected data, b_{st} is the bias term, and $f()$ is the activation function.

Transaction feature extraction

Existing methods for identifying illegal accounts on the blockchain have not considered the various types of transactions on the blockchain comprehensively. Instead, different types of transactions have been grouped as one type, without taking into account transaction types like calling and creating smart contracts. In order to address the need for a comprehensive approach to different transaction types, this paper proposes three random walk strategies to transform blockchain nodes into low-dimensional vectors and extract transaction features of blockchain accounts.

To gain a more comprehensive understanding of the characteristics of the transaction network and to effectively sample node neighborhoods, this method fully considers the sampling strategy based on transaction time, transaction type, and transaction amount. The sampling strategy can be defined as follows: Given a source node u, sample a random walk sequence of length l, with the starting vertex $c_0 = u$. To sample c_i, the following strategy is used:

$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{v,x}}{Z}, & if\,(v,x) \in E \\ 0, & else \end{cases} \tag{10}$$

where $\pi_{v,x}$ is the unnormalized transition probability from node v to x, and Z is the normalizing constant.

Transaction time-based random walk

In blockchain when a new block is created, the transactions contained in the block are sorted according to specific rules, and then the transactions are packed into the block and a timestamp is generated, which is the time of the transaction. For illegal nodes, the last few transactions are usually the transfer of funds obtained from illegal activities, so this sampling strategy is biased towards the time of the last few transactions. Also under time-based biased sampling, the transfer probability from node u to neighboring node$x \in V_u$ is

$$P_{T_{ux}} = \frac{MaxT(u,x)}{\Sigma_{x' \in V_u} SumT(u,x')} \tag{11}$$

where $MaxT(u,x)$ denotes the latest timestamp of the transaction between node u and x, and $SumT\left(u,x'\right)$

denotes the sum of the timestamps of the transactions between node u and other nodes.

Transaction type-based random walk

Blockchain transactions have different types, such as money transfers, calling smart contracts, and creating smart contracts. The majority of illegitimate accounts involve transfer transactions, while legitimate accounts perform more transfers and contract calls. Setting different weights for each transaction type can be helpful in restricting the sampling process to specific kinds of transactions. For instance, weights of 1, 2, and 3 can be set for transfers, smart contract calls, and smart contract creations, respectively. This strategy can increase the accuracy of estimating the transfer probability between connecting nodes. The transfer probability from node u to neighboring node$x \in V_u$ is

$$P_{E_{ux}} = \frac{MaxE(u,x)}{\Sigma_{x' \in V_u} SumE(u,x')} \tag{12}$$

where $MaxE(u,x)$ denotes the edge with the largest weight between node u and x, and $SumE\left(u,x'\right)$ denotes the sum of the transaction type weights between node u and other nodes.

Transaction amount-based random walk

The larger the transaction amount between blockchain accounts, the closer the relationship between the two nodes. Therefore, this sampling strategy tends to favor the transactions with the amount-based biased sampling, the transfer probability from node u to neighboring node $x \in V_u$ is

$$P_{A_{ux}} = \frac{MaxA(u,x)}{\Sigma_{x' \in V_u} SumA(u,x')} \tag{13}$$

where $MaxA(u,x)$ denotes the maximum amount of transactions between node u and x, and $SumA\left(u,x'\right)$ denotes the sum of transaction amounts between node u and other nodes.

When using different random walk strategies for random walk, the non-normalized transition probability $\pi_{v,x}$ from node u to x is defined as $P_{T_{ux}}, P_{E_{ux}}, P_{A_{ux}}$, respectively.

Then a random walk is performed using the calculated transfer probabilities, and finally the node embedding is optimized using the stochastic gradient descent method to obtain the objective function f. The objective function f maximizes the log probability of the occurrence of nodes from the neighborhood $N_S(u)$ for a node u conditioned on its node embedding, i.e.,

$$\max_f \sum_{u \in V} \log P_r\left(N_S(u) \mid f(u)\right) \tag{14}$$

Specifically, on the basis of the constructed transaction network, in order to take into account the transaction timestamp, transaction amount, transaction type and original graph structure, this paper propose a method that uses the hyperparameters $\alpha, \beta, \gamma((\alpha, \beta, \gamma) \in [0,1])$ to balance the effects of the multiple embeddings, and the embedding results can be expressed as

$$Z_e = \alpha Z_0 + \beta Z_1 + \gamma Z_2 + Z_3 \tag{15}$$

where Z_e is the embedding result after combining multiple cases, Z_0 is the embedding result of the random walk strategy based on transaction amount, Z_1 is the embedding result of the random walk strategy based on time, Z_2 is the embedding result of the random walk strategy based on transaction type, and Z_3 is the unbiased embedding result.

Feature fusion

Finally, the embedding results are combined with labeled nodes to consider node information. The final embedding result is obtained as

$$Z = Z_e \mid\mid V_L \tag{16}$$

where $\mid\mid$ represents the vector splice between Z_e and V_L.

In conclusion, the pseudocode for T^2A2vec is presented in Algorithm 1 and Algorithm 2. Algorithm 1 aims to generate a random walk sequence based on the calculated transition probabilities of the graph. The input consists of the graph with the transition probabilities, the starting node u of the walk, and the length l of the random walk sequence. The output is a random walk sequence with a length of l. First, initialize the walk result list *walk*. Then, the iterative process of randomly selecting the next node s, via iterating through the current node v and its neighboring nodes b_v, is executed. Finally, the selected node s is added to the walk result sequence. After each iteration is completed, the walk sequence is returned.

Algorithm 2 applies different sampling strategies to the constructed blockchain transaction graph, extracts account features and transaction features, and fuses the account features and transaction features to obtain the fused embedding results. The inputs of the algorithm are the constructed transaction graph G, embedding dimension d, walk length l, neighborhood size k, number of random walk sequences starting from each source node r, bias parameters α, β, γ, and node attributes V_L, and the output is the most embedding result Z. First, the account features are extracted using Eq. 9, and the transfer probabilities are calculated using Eqs. 11 to 13. Then, iterative sampling is performed using

different transfer probabilities, and the specific iterative process is to set weights for the edges based on the calculated transfer probabilities, reconstruct the new graph G', and initialize the list walks to be empty. For each node, a sequence of r random walks of length l is generated starting at that node. The computational procedure is to execute Algorithm 1 for each node $u \in V$ and add the result to walks. Then, stochastic gradient descent is executed to solve for the transaction characteristics of the node. The algorithm finally fuses the account features and transaction features to obtain the feature representation of the node.

Input: Graph $G' = (V, E, P)$, start node u, length l, index i.
Output: Walk results $walk$.
walk=$[u]$
for walk_iter = 1 to l **do**
  v=walk$[-1]$
  b_v=GetNeighbors(v, G')
  s=AliasSample$(b_v, P[i])$
  walk.append(s)
end for
return $walk$

Algorithm 1 *T^2A2walk*

Input: Input the transaction network G, embedding dimension d, walk length l, neighborhood size k, walks per node r, bias parameter α, β, γ, node attribute V_L.
Output: Embedding results Z.
Calculate V_L using Eq.(9)
Calculate unbiased random walk P' using Eq.(10)
Calculate $P_{T_{ux}}$, $P_{E_{ux}}$, $P_{A_{ux}}$ using Eq.(11)-Eq.(13)
P=$[P_{A_{ux}}, P_{T_{ux}}, P_{E_{ux}}, P']$
$G' = (V, E, P)$
for $i = 0$ to 4 **do**
  walks = []
  for $iter = 1$ to r **do**
    for $u \in V$ **do**
      Get walks by executing Algorithm 1
    end for
  end for
  Z_i = StochasticGradientDescent$(k, d, walks)$
end for
Calculate Z using Eq. (15), Eq. (16)
return Z

Algorithm 2 T^2A2vec AlgorithmExperiments

In this section an experimental evaluation is performed to assess the effectiveness of the proposed T^2A2vec algorithm using the dataset of Ethereum transactions collected in this paper. Specifically, the goal of this paper is to answer the following research questions: **RQ1)** Is there a difference between the node characteristics of normal and illicit accounts? **RQ2)** Does T^2A2vec effectiveness in detecting illicit accounts outperform the state-of-the-art benchmark algorithm? How much different classifiers affect the performance? **RQ3)** How much different random walk strategies and different embedding dimensions affect the performance of the method? **RQ4)** How much different parameter settings affect the performance of the method?

Dataset and evaluation Criteria

Experiments were conducted using the data set collected in Section III-A. This dataset has 4,986 illicit nodes, 5,000 normal nodes, and 312,751 accounts transacting with illicit and normal nodes, for a total of 1,129,542 transaction records.

To comprehensively evaluate the effectiveness of the method, the dataset is divided into three ways, as shown in Table 3. The training sets of D_1, D_2, and D_3 contain 50%, 70%, and 80% of randomly selected label nodes, respectively. During the training, validation, and testing of different models, only the classification performance of labeled nodes is considered.

By analyzing baseline methods compared with similar work, the T^2A2vec method is compared with several methods, including: (1) some random walk methods (i.e., DeepWalk [41], Node2Vec [42], LINE [43]), (2) some popular deep learning network-based despicable methods (i.e., GCN [44], GAT [45], GraphSAGE [46]), (3) some of our replicated Blockchain node detection methods (i.e., trans2vec [36]).

- **DeepWalk** [41]: DeepWalk learns the social representation of a network by truncated random walk, which gives better results even when the network has few labeled vertices. The method also has the advantage of being scalable and can adapt to changes in the network.
- **Node2vec** [42]: Node2vec belongs to the class of graph neural network random walk models, which generate a random walk, sample the random walk, get a combination of nodes, context, and then model this combination, and get a representation of the network nodes by processing word vectors.
- **LINE** [43]: By optimizing the first-order similarity and second-order similarity, two representation vectors of the vertex, the source vector and the target vector, can be obtained, and when used, the two vectors are combined as the final representation of the vertex.
- **GCN** [44]: GCN is a neural network architecture that operates on graph data, it is so powerful that even a randomly initialized two-layer GCN can generate feature representations of nodes in a graph network.
- **GAT** [45]: GAT is a representative graph convolutional network that introduces an attention mechanism to achieve better neighbor aggregation, and the attention mechanism also endows the model with a certain degree of interpretability.
- **GraphSAGE** [46]: GraphSAGE uses a multi-layer aggregation function, where each layer aggregates the information of nodes and their neighbors to get the feature vector of the next layer. GraphSAGE uses

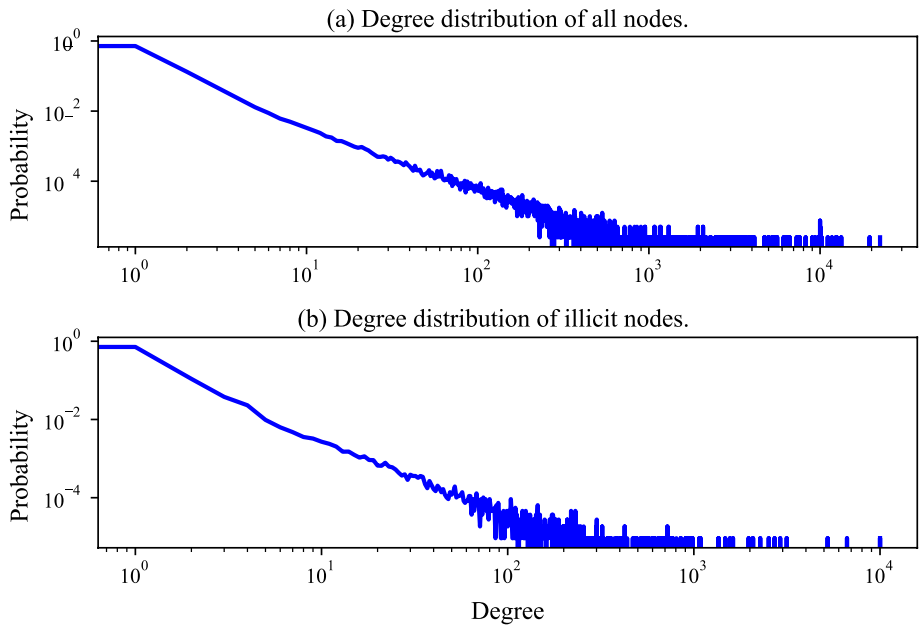

Fig. 3 Degree distribution of all/illicit nodes

Table 3 Three training-validation-test set division methods

Dataset	Train set	Validation set	Test set
D_1	50%	10%	40%
D_2	70%	10%	20%
D_3	80%	10%	10%

the neighborhood information of nodes and does not depend on the global graph structure.

- **trans2vec** [36]: trans2vec is a method to detect Blockchain phishing scams by mining Blockchain transaction records, taking into account the transaction amount and timestamp.

The embedding dimensions of DeepWalk, Node2Vec, LINE, and trans2vec methods are all set to 64, and the rest of the parameters are set according to the optimal parameters in the paper.

This paper implements the proposed algorithm using the graph deep learning toolkit CogDL [47]. All experiments were conducted on a Linux server with GPU (GeForce GTX 3090) and CPU (Intel Xeon E5-2620), running Ubuntu 16.04.1. Python v3.8.6 and CogDL v0.5.3 were the versions used for the implementation of the proposed algorithm. The following parameters need to be specified in this paper: embedding size d, number of walks per node r, walk length l, context size k, and hyper-parameters α, β, and γ. For the experiments conducted in this paper, the parameters were set as follows: d=64, r=20, l=5, k=10, and α=β=γ=0.5. Each experiment in this paper was repeated 10 times.

In order to evaluate the classification effect of different classification methods, three evaluation indicators are selected: Precision, Recall and F1-score. These three indicators are set as follows.

$$Precision = \frac{TP}{TP + FP} \tag{17}$$

$$Recall = \frac{TP}{TP + FN} \tag{18}$$

$$F1 - score = 2 * \frac{Precision * Reacll}{Precision + Recall} \tag{19}$$

Node feature analysis (RQ1)

To answer **RQ1**, the node characteristics of different types of nodes are analyzed. Fig. 3 and Table 4 report the corresponding results. The following conclusions can be drawn:

(1) Compared with normal nodes, the degree of illicit nodes is generally concentrated at 10^2. The main reason for this is that as the volume of transactions increases, the more likely it is to be added to the database of illicit nodes. When it is marked as an illicit node, ordinary users will not transact with it, resulting in a small degree of illicit nodes.

Table 4 Analysis of dataset account attributes

Feature	Normal accounts			Illicit accounts		
	mean	median	std	mean	median	std
NTS	513.40	90.00	1340.54	22.91	2.00	198.20
max_VS	135.67	4.90	2329.42	75.38	4.00	856.16
min_VS	0.61	0.00	41.00	17.49	0.10	434.24
TVS	2963.54	28.20	85792.10	133.99	5.99	1203.30
AVS	3.02	0.30	47.40	27.60	1.57	485.83
avg_TIS	13491.97	2550.67	36591.24	17332.89	111.55	77438.72
NTR	78.92	14.00	381.46	34.61	6.00	248.60
max_VR	114.98	3.46	2545.93	58.31	2.41	747.65
min_VR	1.04	0.00	41.75	21.52	0.02	607.25
TVR	1906.49	14.24	77983.87	103.57	6.00	949.58
AVR	8.48	0.73	80.15	30.36	0.90	623.45
avg_TIR	17083.43	5830.90	34763.81	6684.79	247.67	28455.29
TETF	4.83	0.61	16.99	0.14	0.01	1.16
AETF	8.25E-03	7.12E-03	8.00E-03	1.66E-03	5.93E-04	1.18E-02
TDFL	9.06E+05	7.78E+05	7.57E+05	2.07E+05	2.38E+04	4.26E+05
TEB	-1057.05[a]	-3.24[a]	24067.01	-30.42[a]	0.00	572.08
UAS	88.66	21.00	328.15	9.47	2.00	57.37
UAR	26.19	6.00	240.67	23.55	5.00	188.21

[a] The negative values are present because this paper only collected the last 10,000 transactions, so the old ones are ignored

(2) The ***mean*** and ***standard deviation*** of certain characteristics of illegal accounts (i.e., min_VS, AVS, min_RA, AVR, TEB) are greater than those of normal accounts, and the ***median*** difference is smaller. The minimum amount sent and received by illegal accounts is greater than that of normal users, and the average amount received and sent is approximately that of normal accounts. This means that compared to the average account, the illegal account entices the victim to make a large amount of transfer transactions and then to cash out a large amount of money.

(3) The average number of sending transactions for a normal account is 513.4, with 78.92 receiving transactions, while for illegal accounts, the average number of sending transactions is 22.91, with 34.61 receiving transactions. This suggests that illegal accounts conduct a small number of transactions to reduce the probability of detection. Low transaction volume can make it difficult to imitate the transaction patterns of normal accounts, reducing the risk of being caught.

(4) The average transaction amount for normal accounts is 125.29 ether, while for illegal accounts it is only 3.46 ether. This suggests that illegal accounts tend to have much smaller transaction amounts, possibly to evade regulation or make tracing more difficult. Meanwhile, the transaction amounts for normal accounts are much higher, which is also in line with their typical use cases, such as daily consumption and investment.

(5) Normal accounts have a balanced number of sending and receiving transactions, while illegal accounts are more likely to send transactions than to receive them. This may indicate that illegal accounts are mainly used for sending funds rather than receiving them, consistent with the typical feature of illegal activities involving fund transfers.

Classification performance (RQ2)

To answer **RQ2**, the performance of different algorithms was firstly evaluated, and secondly evaluate the effectiveness of the method when using different classifiers for classification, and the results are shown in Tables 5, 6. The following conclusions can be drawn:

(1) For all the compared methods, our method T^2 A2vec achieves significant advantages under different evaluation metrics. Under the D_2 dataset, our method achieves 93.21% precision, 92.96% recall, and 93.08% F1. Under the D_1 dataset, the precision of this method reaches 88.09%, which exceeds our method by 0.83%. The performance of deep learning based methods fluctuates between 76% and 89%. The performance gap between the methods based on random walk is obvious, and the performance of LINE and Node2vec is 10%-15% higher than that of DeepWalk. The worst performer is Deepwalk, whose lowest recall rate is only 69.52%.

(2) An interesting phenomenon emerges with the increase of training samples. As the number of training

Table 5 The classification results (%) over the methods

Method	Dataset	D_1			D_2			D_3		
	Metric	Pre.	Recall	F1	Pre.	Recall	F1	Pre.	Recall	F1
Random walk	DeepWalk	71.72	69.52	70.60	71.73	73.74	72.73	71.49	73.55	72.51
	LINE	81.89	86.28	84.03	83.42	87.11	85.22	82.59	86.91	84.69
	Node2Vec	81.31	81.86	81.58	86.85	81.98	84.34	87.52	81.96	84.65
Deep learning	GCN	76.21	76.21	76.22	77.75	73.64	75.65	77.72	74.18	75.91
	GAT	83.31	86.58	84.92	84.76	86.69	85.71	84.83	87.23	86.01
	GraphSAGE	83.28	87.12	85.16	84.45	88.00	86.19	89.21	88.38	86.52
Blockchain method	trans2vec	**88.09**	84.47	86.25	88.6	85.19	86.86	89.21	85.79	87.47
Ours	T^2A2vec	87.26	**89.33**	**88.27**	**93.21**	**92.96**	**93.08**	**90.45**	**91.23**	**90.83**

samples increases, the metrics of deep learning-based methods are improved. GraphSAGE has the most obvious improvement, and its precision rate has increased by about 6%. However, the random walk-based method and our method show a performance drop when the training set ratio is increased from 70% to 80%. Among them, the performance of our method drops by about 1.5%-2.7%.

(3) Compared with random walk-based methods, our method outperforms by 3%-20% on different evaluation metrics. Among them, Deepwalk has the worst performance, and the precision index in the D_2 dataset is 21.48% lower than our method. Node2vec has the best performance, and the precision index in the D_3 dataset is only 2.93% lower than our method. Node2vec obtains better node representation through different transfer strategies, however, this method ignores transaction time, transaction amount and transaction type, resulting in incomplete representation learning of nodes. Our T^2A2vec learns a more efficient node representation through a random walk strategy based on transaction time, transaction amount, and transaction type, and achieves better results.

(4) Significant differences in performance have also emerged for deep learning-based graph representation learning methods. Among them, the performance gap of GCN reaches 10%-20%, and GCN performs neighborhood information aggregation in the process of neural network learning, which may cause the error to enlarge as the neural network deepens. GAT and GarphSAGE enrich the features of nodes and enhance the representation ability of nodes by sampling and aggregating neighbors with statistical characteristics.

(5) By comparing several widely selected classifiers, it is found that the classifier is also an important factor affecting the detection performance of malicious nodes. The performance gap of different classifiers is obvious. For dataset D_1, the highest F1 score was achieved by the random forest method with a score of 88.27%, followed by logistic regression with a score of 85.66%. Naïve Bayes had the lowest F1 score of 72.13%. For dataset D_2, the random forest method had the highest F1 score of 93.08%, followed by SVM with a score of 79.75%. Naïve Bayes had the lowest F1 score of 72.17%. For dataset D_3, the logistic regression method had the highest F1 score of 87.08%, followed by random forest with a score of 90.83%. Naïve Bayes had the lowest F1 score of 73.22%. Overall, the random forest method performed the best across all three datasets, while Naïve Bayes had the lowest performance. The logistic regression and SVM methods also had strong performance across the datasets. The decision tree method had moderate performance, ranking fourth out of five methods.

Random walk strategy and embedding dimension analysis (RQ3)

To answer **RQ3**, when the node embedding dimension d is equal to 4, 8, 16, 32 and 64, the classification effects of different walking strategies are tested, and the final classification results are shown in Fig. 4. The following conclusions can be drawn:

(1) Different random strategies behave differently in different situations. The method proposed in this paper outperforms the other three random walk strategies. The random walk strategy based on transaction time outperforms the random walk strategy based on transaction type.

(2) Different embedding dimensions significantly impact the classification effect. With the increase of the embedding dimension, the performance of the algorithm is obviously improved. The most obvious improvement in overall performance is the time-based random walk strategy, whose recall rate has increased from 68% to 87%. Our algorithm achieves excellent results even when the embedding dimension is 4, with an overall performance of 86%. The most obvious performance improvement is that the embedding dimension is increased from 4 to 8.

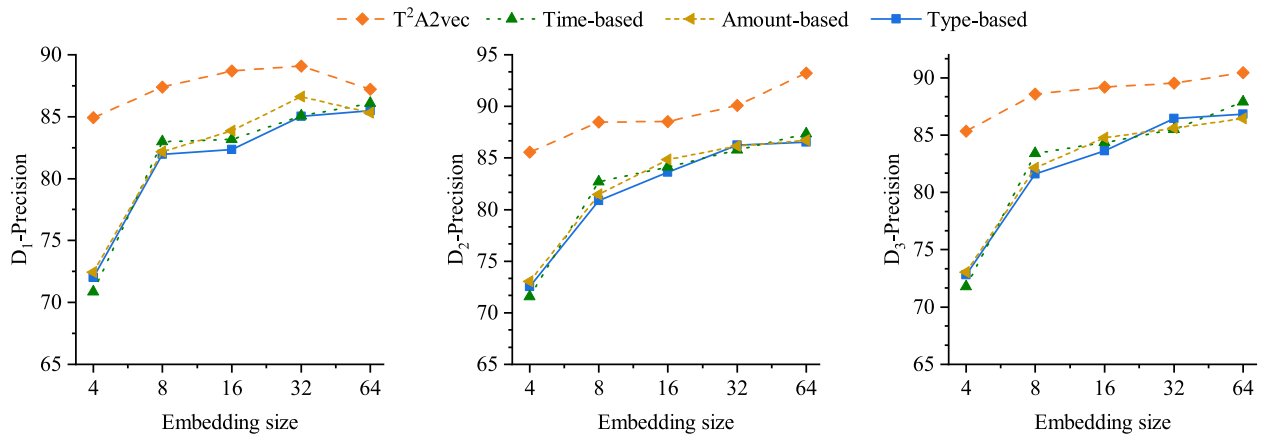

Fig. 4 Performance comparison of different random walk strategies under different embedding dimensions

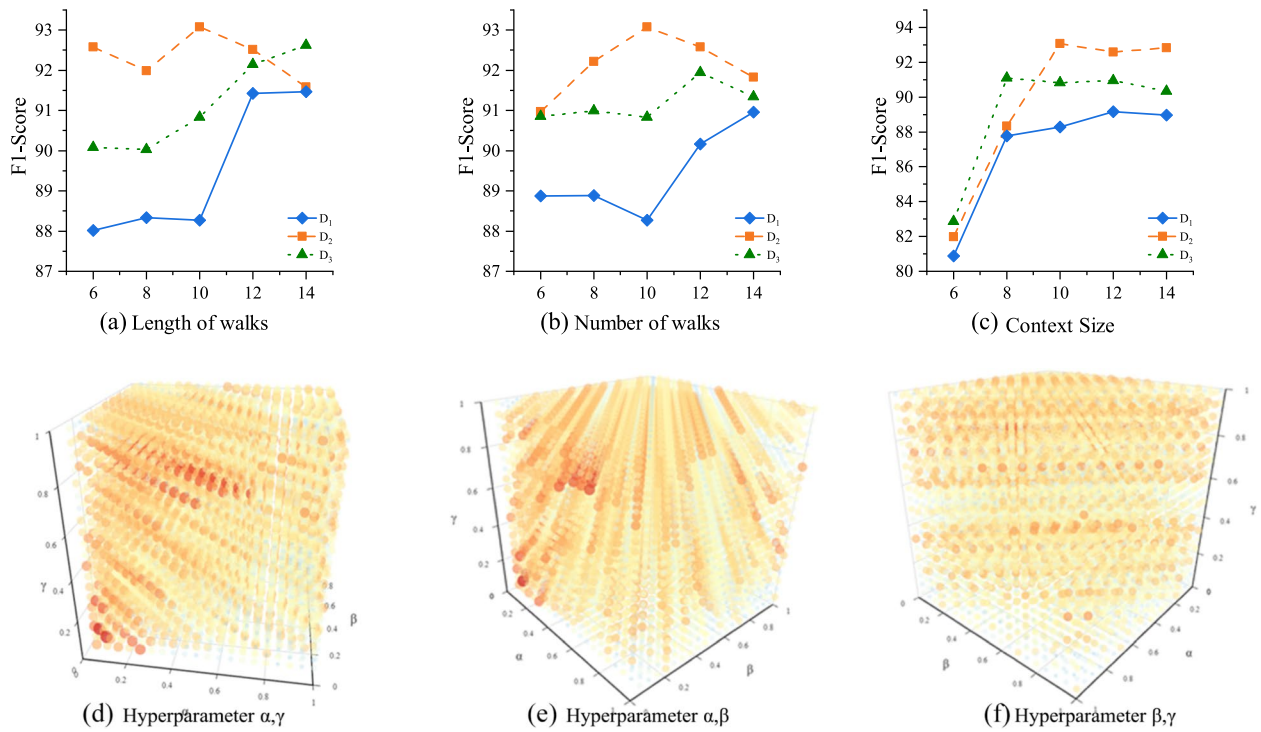

Fig. 5 Results of parameters analysis

Table 6 Classification results(%) of different classification methods

Method	D_1			D_2			D_3		
	Pre.	**Recall**	**F1**	**Pre.**	**Recall**	**F1**	**Pre.**	**Recall**	**F1**
Naïve Bayes	72.72	71.55	72.13	72.72	71.63	72.17	73.81	72.64	73.22
SVM	78.59	78.50	78.54	79.82	79.70	79.75	79.81	79.92	79.86
Logistic regression	85.76	85.57	85.66	86.21	86.27	86.23	87.06	87.12	87.08
Decision tree	80.67	80.76	80.71	83.26	83.23	83.24	82.44	82.55	82.49
Random forest	**87.23**	**89.33**	**88.27**	**93.21**	**92.93**	**93.08**	**90.45**	**91.20**	**90.83**

Afterwards, with the increase of the embedding dimension, the speed of the algorithm performance improvement decreases. When the embedding dimension reaches 64, the performance gap is generally around 2%-3%. This is because the larger the node vector dimension is, the richer the network structure and node information can be retained. When a high accuracy embedding result is needed, a larger embedding dimension should be chosen.

Parameter sensitivity analysis (RQ4)

To answer **RQ4**, the parameters l, r, k, α, β and γ in the D_2 dataset were analyzed. When analyzing parameters, all other parameters are set to default values. The final classification results are shown in Fig. 5. The following conclusions can be drawn:

(1) The parameters l, r and k all achieve the best results at 10. Among them, the parameter l has the least impact on the algorithm performance, and the algorithm performance on the D_2 dataset fluctuates between 91.58% and 93.03%. The parameter r has a greater impact on the algorithm performance than the parameter l, and the algorithm performance of the D_2 dataset fluctuates between 90.97% and 93.03%. The parameter k has the greatest impact on the performance of the algorithm. When k is equal to 6, the F1 of the D_2 dataset is only 82%. When k is increased from 6 to 8, the performance improvement is the most obvious, and the overall performance is improved by about 8%.

(2) The effect of the hyperparameter α, β, γ on the experimental results in the D_2 dataset can be seen in Fig. 5(d)-(f) that the optimal results are achieved when α, β, γ is taken as (0.1, 0.05, 0.1) and (0.1, 0.7, 0.8), respectively.

Conclusion and future work

In this paper, a method called T^2A2vec is proposed to handle the task of classifying nodes in blockchain. Through the analysis of blockchain transaction data, this paper designed a node feature collection strategy , which can completely and accurately describe the node's transaction behavior. T^2A2vec proposes three random walk strategies, based on transaction time, transaction type, and transaction amount to solve the problem of account classification in blockchain. Using these three strategies, more efficient graph learning can be performed on graphs, resulting in stronger node representations. By obtaining node representations from node features and graph learning, this paper improves the performance of account classification detection on blockchain. Extensive experiments demonstrate that our proposed T^2A2vec outperforms state-of-the-art algorithms in performance and utility.

In future work, the openness of blockchain technology will be considered to propose a detection method for specific categories of illegitimate behavior accounts. Instead, this work only targets the detection of nodes on the blockchain and cannot identify the type of nodes.

Acknowledgements
Thanks to Dr. Zhaowei Liu of Yantai University for his help in our work.

Authors' contributions
Shenqiang Wang: Conceptualization, Methodology, Software, Investigation, Formal analysis, Writing – original draft. Zhaowei Liu: Project administration, Supervision, Funding acquisition. Haiyang Wang: Formal analysis, Supervision. Jianping Wang: Formal analysis, Supervision. All authors reviewed the manuscript.

Authors' information
Shenqiang Wang is currently pursuing the M.Sc. degree with the School of Computer and Control Engineering, Yantai University, Yantai, China. His current research interests include blockchain and machine learning with graphs.
Zhaowei Liu received the Ph.D. degree from the Shandong University, Jinan, in 2018. Currently, he is a Professor at the Yantai University, Yantai, China. His research interests include blockchain, and machine learning with graphs.
Haiyang Wang received the master's degree from Peking University, Beijing, China, in 2011. He is currently the director of Institute of Network Technology (Yantai) and of Yantai Association for Science and Technology (YTAST). His main research interests include information security, big data technology and artificial intelligence.
Jianping Wang is an engineer from Shandong Institute of Marine Resources and Environment. He obtained a master's degree in cartography and geographic information engineering from China University of Mining and Technology in 2013. His major is cartography and geographic information engineering. His current research is marine satellite remote sensing.

Funding
This work was supported in part by the National Natural Science Foundation of China under Grant 62272405, School and Locality Integration Development Project of Yantai City (2022), the Youth Innovation Science and Technology Support Program of Shandong Provincial under Grant 2021KJ080, the Natural Science Foundation of Shandong Province, Grant ZR2022MF238, Yantai Science and Technology Innovation Development Plan Project under Grant 2021YT06000645, the Open Foundation of State key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) under Grant SKLNST-2022-1-12.

Availability of data and materials
The data used to support the findings of this study are available from the corresponding author upon request.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 23 March 2023 Accepted: 28 May 2023
Published online: 15 June 2023

References

1. Zhou X, Yang X, Ma J, Kevin I, Wang K (2021) Energy-efficient smart routing based on link correlation mining for wireless edge computing in IoT. IEEE Internet Things J 9(16):14988–14997
2. Hu C, Fan W, Zeng E, Hang Z, Wang F, Qi L, Bhuiyan MZA (2021) Digital twin-assisted real-time traffic data prediction method for 5g-enabled internet of vehicles. IEEE Trans Ind Inf 18(4):2811–2819
3. Zhou X, Xu X, Liang W, Zeng Z, Yan Z (2021) Deep-learning-enhanced multitarget detection for end–edge–cloud surveillance in smart IoT. IEEE Internet Things J 8(16):12588–12596
4. Qi L, Chi X, Zhou X, Liu Q, Dai F, Xu X, Zhang X (2022) Privacy-aware data fusion and prediction for smart city services in edge computing environment. In: 2022 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), IEEE, pp 9–16
5. Qi L, Hu C, Zhang X, Khosravi MR, Sharma S, Pang S, Wang T (2021) Privacy-aware data fusion and prediction with spatial-temporal context for smart city industrial environment. IEEE Trans Ind Inform 17(6):4159–4167
6. Zhang T, Li Y, Chen CP (2021) Edge computing and its role in industrial internet: Methodologies, applications, and future directions. Inf Sci 557:34–65
7. Li X, Li D, Wan J, Liu C, Imran M (2018) Adaptive transmission optimization in sdn-based industrial internet of things with edge computing. IEEE Internet Things J 5(3):1351–1360
8. Dai X, Xiao Z, Jiang H, Alazab M, Lui JC, Dustdar S, Liu J (2022) Task co-offloading for d2d-assisted mobile edge computing in industrial internet of things. IEEE Trans Ind Inf 19(1):480–490
9. Ranaweera P, Jurcut AD, Liyanage M (2021) Survey on multi-access edge computing security and privacy. IEEE Commun Surv Tutor 23(2):1078–1124
10. Wang R, Lai J, Zhang Z, Li X, Vijayakumar P, Karuppiah M (2022) Privacy-preserving federated learning for internet of medical things under edge computing. IEEE J Biomed Health Inform 27(2):854–865.
11. Zhou X, Liang W, Yan K, Li W, Kevin I, Wang K, Ma J, Jin Q (2022) Edge-enabled two-stage scheduling based on deep reinforcement learning for internet of everything. IEEE Internet Things J 10(4):3295–3304
12. Yuan L, He Q, Chen F, Zhang J, Qi L, Xu X, Xiang Y, Yang Y (2022) Csedge: Enabling collaborative edge storage for multi-access edge computing based on blockchain. IEEE Trans Parallel Distrib Syst 33(8):1873–1887
13. Xu G, Dong J, Ma C, Liu J, Cliff UGO (2022) A certificateless signcryption mechanism based on blockchain for edge computing. IEEE Internet Things J https://doi.org/10.1109/JIOT.2022.3151359
14. Qi L, Liu Y, Zhang Y, Xu X, Bilal M, Song H (2022) Privacy-aware point-of-interest category recommendation in internet of things. IEEE Internet Things J 9(21):21398–21408
15. Liang W, Hu Y, Zhou X, Pan Y, Kevin I, Wang K (2021) Variational few-shot learning for microservice-oriented intrusion detection in distributed industrial IoT. IEEE Trans Ind Inform 18(8):5087–5095
16. Wang G, Li C, Huang Y, Wang X, Luo Y (2022) Smart contract-based caching and data transaction optimization in mobile edge computing. Knowl-Based Syst 252(109):344
17. Boo E, Kim J, Ko J (2021) Litezkp: Lightening zero-knowledge proof-based blockchains for IoT and edge platforms. IEEE Syst J 16(1):112–123
18. Nawaz A, Peña Queralta J, Guan J, Awais M, Gia TN, Bashir AK, Kan H, Westerlund T (2020) Edge computing to secure IoT data ownership and trade with the Ethereum blockchain. Sensors 20(14):3965
19. Dasgupta D, Shrein JM, Gupta KD (2019) A survey of blockchain from security perspective. J Bank Financ Technol 3:1–17
20. Berdik D, Otoum S, Schmidt N, Porter D, Jararweh Y (2021) A survey on blockchain for information systems management and security. Inf Process Manag 58(1):102397
21. Feng J, Yang LT, Gati NJ, Xie X, Gavuna BS (2020) Privacy-preserving computation in cyber-physical-social systems: A survey of the state-of-the-art and perspectives. Inf Sci 527:341–355
22. Liu Z, Yang D, Wang Y, Lu M, Li R (2023) Egnn: graph structure learning based on evolutionary computation helps more in graph neural networks. Appl Soft Comput 135:110040
23. Zhou X, Liang W, Li W, Yan K, Shimizu S, Kevin I, Wang K (2021) Hierarchical adversarial attacks against graph-neural-network-based IoT network intrusion detection system. IEEE Internet Things J 9(12):9310–9319
24. Khan A (2022) Graph analysis of the ethereum blockchain data: A survey of datasets, methods, and future work. In: 2022 IEEE International Conference on Blockchain (Blockchain), IEEE, pp 250–257
25. Farrugia S, Ellul J, Azzopardi G (2020) Detection of illicit accounts over the Ethereum blockchain. Expert Syst Appl 150:1–11
26. Poursafaei F, Hamad GB, Zilic Z (2020) Detecting malicious ethereum entities via application of machine learning classification. In: Proceedings 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS). IEEE, pp 120–127
27. Zhang Y, Yu W, Li Z, Raza S, Cao H (2022) Detecting Ethereum Ponzi schemes based on improved lightGBM algorithm. IEEE Trans Comput Soc Syst 9(2):624–637
28. Chen W, Guo X, Chen Z, Zheng Z, Lu Y (2020) Phishing scam detection on ethereum: Towards financial security for blockchain ecosystem. In: Proceedings 29th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, pp 4506–4512
29. Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY (2020) A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst 32(1):4–24
30. Liu M, Gao H, Ji S (2020) Towards deeper graph neural networks. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 338–348
31. Liu X, Tang Z, Li P, Guo S, Fan X, Zhang J (2022) A graph learning based approach for identity inference in dapp platform blockchain. IEEE Trans Emerg Top Comput 10(1):438–449
32. Weber M, Domeniconi G, Chen J, Weidele DKI, Bellei C, Robinson T, Leiserson CE Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. 2019. arXiv:1908.02591
33. Cai H, Zheng VW, Chang KCC (2018) A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE Trans Knowl Data Eng 30(9):1616–1637
34. Wang X, Bo D, Shi C, Fan S, Ye Y, Yu PS (2023) A survey on heterogeneous graph embedding: Methods, techniques, applications and sources. IEEE Trans Big Data 9(2):415–436
35. Yuan Q, Huang B, Zhang J, Wu J, Zhang H, Zhang X (2020) Detecting phishing scams on ethereum based on transaction records. In: Proceedings 2020 IEEE International Symposium on Circuits and Systems. IEEE, pp 1–5
36. Wu J, Yuan Q, Lin D, You W, Chen W, Chen C, Zheng Z (2022) Who are the phishers? phishing scam detection on Ethereum via network embedding. IEEE Trans Syst Man Cybern: Syst 52(2):1156–1166
37. Yuan Z, Yuan Q, Wu J (2020) Phishing detection on Ethereum via learning representation of transaction subgraphs. In: Proceedings International Conference on Blockchain and Trustworthy Systems. Springer, pp 178–191
38. Lin D, Wu J, Yuan Q, Zheng Z (2020) Modeling and understanding Ethereum transaction records via a complex network approach. IEEE Trans Circ Syst II: Express Briefs 67(11):2737–2741
39. Qi L, Yang Y, Zhou X, Rafique W, Ma J (2022) Fast anomaly identification based on multiaspect data streams for intelligent intrusion detection toward secure industry 4.0. IEEE Trans Ind Inf 18(9):6503–6511
40. Liu Z, Yang D, Wang S, Su H (2022) Adaptive multi-channel bayesian graph attention network for IoT transaction security. Digit Commun Netw. https://doi.org/10.1016/j.dcan.2022.11.018
41. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 701–710
42. Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 855–864
43. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) Line: Large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web. ACM, pp 1067–1077
44. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: Proceedings International Conference on Learning Representations. ICLR, pp 1-14

45. Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph Attention Networks. In: Proceedings International Conference on Learning Representations. ICLR, pp 1–12
46. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Proceedings Neural Information Processing Systems. MIT, pp 1025–1035
47. Cen Y, Hou Z, Wang Y, Chen Q, Luo Y, Yu Z, Zhang H, Yao X, Zeng A, Guo S, Dong Y, Yang Y, Zhang P, Dai G, Wang Y, Zhou C, Yang H, Tang J (2023) Cogdl: A comprehensive library for graph deep learning. In: Proceedings of the ACM Web Conference 2023 (WWW'23). ACM, pp 747–758

Publisher's Note