Identity Inference on Blockchain Using Graph Neural Network

Jie Shen[1,2], Jiajun Zhou[1,2], Yunyi Xie[1,2], Shanqing Yu[1,2(✉)], and Qi Xuan[1,2,3]

[1] Institute of Cyberspace Security, Zhejiang University of Technology,
Hangzhou 310023, China
yushanqing@zjut.edu.cn
[2] College of Information Engineering, Zhejiang University of Technology,
Hangzhou 310023, China
[3] PCL Research Center of Networks and Communications, Peng Cheng Laboratory,
Shenzhen 518000, China

Abstract. The anonymity of blockchain has accelerated the growth of illegal activities and criminal behaviors on cryptocurrency platforms. Although decentralization is one of the typical characteristics of blockchain, we urgently call for effective regulation to detect these illegal behaviors to ensure the safety and stability of user transactions. Identity inference, which aims to make a preliminary inference about account identity, plays a significant role in blockchain security. As a common tool, graph mining technique can effectively represent the interactive information between accounts and be used for identity inference. However, existing methods cannot balance scalability and end-to-end architecture, resulting high computational consumption and weak feature representation. In this paper, we present a novel approach to analyze user's behavior from the perspective of the transaction subgraph, which naturally transforms the identity inference task into a graph classification pattern and effectively avoids computation in large-scale graph. Furthermore, we propose a generic end-to-end graph neural network model, named I^{2}BGNN, which can accept subgraph as input and learn a function mapping the transaction subgraph pattern to account identity, achieving de-anonymization. Extensive experiments on EOSG and ETHG datasets demonstrate that the proposed method achieve the state-of-the-art performance in identity inference.

Keywords: Blockchain · Identity inference · Graph classification · Graph neural network

1 Introduction

As a distributed database technology, blockchain achieves the function of decentralization, encryption, and tamper-proof. Benefiting from its anonymity, the past few years have witnessed the growing prevalence of cryptocurrencies. As of the first quarter of 2021, there are more than 8,700 kinds of cryptocurrencies

H.-N. Dai et al. (Eds.): BlockSys 2021, CCIS 1490, pp. 3–17, 2021.
https://doi.org/10.1007/978-981-16-7993-3_1

with a total market cap of 1,721 billion dollars. People only need to create a pseudonymous account (synonymous with address in this paper), and they can implement transaction at almost no cost. However, as the volume of transactions surged, the blockchain system of cryptocurrencies has also become a hotbed of illegal and criminal behavior, such as various scams [1–3] (Ponzi schemes, mining scams, scam wallets, fraudulent exchanges, etc.), money laundering [4,5], abusing bot accounts [6] and vulnerability attack [7].

As an open technique, blockchain provides public and tamper-proof transaction records, which creates the condition for data mining and analysis. Recently, the emergence of related research has helped to analyze the transaction pattern and account behavior on the blockchain system, and most of them leverage graph modeling methods. Such as evolution analysis of market via the on-chain transaction graph [8–12], transaction patterns recognition via graph topology and motifs [13,14], detection of abnormal users or transactions via graph embedding or graph neural network [15,16], etc. Among them, identity inference, which can be regarded as a de-anonymization process, is particularly important in blockchain data mining. Generally, identity inference aims to make a preliminary inference about account identity by capturing the characteristics of the transaction pattern of the accounts. For this task, common researches mainly concentrate on manual feature engineering including transaction features [17], graph features [18] and external features [14]. These features are mainly intuitive information, and share the same drawback like weak representation ability for classification. Further, several methods based on random-walk [19] and graph motif [20] capture higher-order network features that are more representational. With the development of graph deep learning, graph convolution network (GCN) has attracted considerable attention and been applied in identity inference gradually, achieving outstanding results [21,22].

After reviewing the above various methods, we summarize two conflicting issues: scalability and end-to-end. On the one hand, real-world transaction data on blockchain systems are generally extremely huge. Although these methods based on feature engineering, especially manual features, show good scalability because of the independence of feature extraction, they cannot achieve end-to-end architecture. End-to-end can reduce the reliance on expertise which is the core of feature engineering, and optimize target task in a complete form rather than multi-flows. On the other hand, although the graph convolution network is commonly achieved via end-to-end, most of them have poor scalability. Because the training of graph convolution network is usually performed on the whole transaction graph, where the loading and computing are not realistic.

Motivated by the subgraph perspective [15], we propose a framework to reconcile the scalability and end-to-end solution for identity inference. Benefiting from previous work, we collect two kinds of on-chain transaction data including Ethereum and EOSIO, to infer the “phisher” and “bot” accounts, respectively. Firstly, we extract the transaction subgraph for each labeled accounts by a sampling mechanism. Through that, each account is transformed into an independent transaction subgraph. The sampling mechanism constrains the scale of transaction subgraph, which can effectively reduce the occupation of resources. Secondly, we propose an end-to-end model, to achieve **I**dentity **I**nference on **B**lockchain using **G**raph **N**eural **N**etwork (named I^2BGNN).

The rest of paper is organized as follows. In Sect. 2, we introduce the related work about identity inference. In Sect. 3, we describe the details of our framework, including subgraph extraction and the architecture of I^2BGNN. Section 4 presents the experiment settings and the comparison of experimental results with discussion. Finally, we conclude the paper in Sect. 5.

2 Related Work

Identity inference, which aims to detect abnormal and illegal accounts, has become an effective means to monitor accounts for platform and measure transaction risks for users. For identity inference on blockchain, related works concentrate on manual feature, graph embedding, graph neural network, and the others.

Manual Feature. Manual feature is a kind of feature engineering that relies on the experience of experts relatively. Normally, the more expert experience involved, the more reliable the feature vectors are. Lin et al. [23] designed various features of transaction timestamps to express the transaction history about the accounts, and constructed a classifier against abnormal bitcoin addresses. Li et al. [17] considered three kinds of features: the basic account feature, the topological feature which is related to transaction patterns, and temporal feature which is captured from the distributions of transaction timestamp. In addition to transaction information, Huang et al. [6] also considered the calling information of smart contract to expand the feature space, and finally realized the identification of bot accounts in EOSIO.

Graph Embedding. Graph embedding aims to learn low-dimensional node representations that capture the graph structure and drive downstream graph mining task such as node classification to identify illicit accounts. Up to now, a series of methods based on DeepWalk (DW) [24] have been used to detect accounts. Yuan et al. [16] used the Node2Vec algorithm which is a variant of DW to extract the potential features of the accounts and classified the phishers by Support-Vector-Machine (SVM). Wu et al. [19] redesigned the walking strategy by using transaction volume, timestamps, and multi-edges features to make their embedding framework more suitable for this task. Subsequently, Yuan et al. [15] extracted the subgraphs for each target account and embedded their transaction topology into feature vector via an embedding method named Graph2Vec [25]. Besides, they introduced the line graph [26] to further enhance the network structure embedding. Chen et al. [27] also used subgraph mechanism and got the embeddings by a graph convolution layer combining graph auto-encoder in an unsupervised way, and achieved phisher classification by LightGBM [28].

Graph Neural Network. This part mainly about the graph neural networks with end-to-end architecture. In [22], the whole transaction graph was sliced into small graphs by timestamp. This operation reduced the computational complexity and memory consumption which alleviate the scalability problem. Subsequently, graph convolution network was used for inductive learning to realize account identity inference. Tam et al. [21] used the mechanism of sampling transaction neighbors which is similar to the subgraph extraction. They characterized edges by embedding the temporal features from the time-series of transactions and incorporating them into the graph convolution network.

Others. Besides the aforementioned methods, there are other frameworks to achieve this identity inference. Phetsouvanh [29] proposed a graph mining technology to detect the suspicious bitcoin flow and account by analyzing the path length and confluence account of the directed subgraph. Zhang [30] introduced the concept of meta-path from the heterogeneous network and constructed multi-constrained meta-path based on time, attribution and topology, which is an effective way to capture behavior pattern features in a complex network.

3 Method

In this section, we first define the identity inference problem on blockchain, then present the details of subgraph extraction for constructing graph classification dataset. Finally, we review the knowledge of using graph neural network (GNN) for learning node and graph representations, and represent the details of proposed I^{2}BGCN model for identity inference.

3.1 Problem Definition

From the perspective of graph mining, identity inference can be regarded as a node classification task. During node classification, the blockchain data will be modeled as a user network with million nodes, which results in unaffordable time and memory consumption for most practical algorithms. Inspired by the core of "neighborhood aggregation" in graph neural network, we transform the node classification problem into a graph classification pattern in return for less time and memory consumption.

Given a set of n_{A} labeled accounts $\mathsf{A} = \{(a_i, y_i) \mid i = 1, 2, \cdots, n_{\mathsf{A}}\}$, we can extract the transaction subgraph centered on each target account. Specifically, we extract the transaction subgraph of account a_i: $G_{a_i} = (V, E_v, E_t, X, y_i)$, where V represents the set of accounts in this subgraph, E_v and E_t represent the directed edge sets that contain information about transaction volume and transaction frequency respectively, X represents the calling information of smart contract, y_i is the label of subgraph G_{a_i}. Note that we assign the label of account a_i to the transaction subgraph centered on it, and transform the node classification problem into a graph classification task: $f_{nc}(a_i) \Rightarrow f_{gc}(G_{a_i})$. The final goal is to learn the transaction patterns of subgraphs and classify centered account into phishing or non-phishing via graph neural networks. The workflow of our framework is shown in Fig. 1, and the details of subgraph extraction and classifier design will be introduced in Subsects. 3.2 and 3.3, respectively.

3.2 Subgraph Extraction

For each account a_i in A, we check the number of its transaction partners (i.e. neighbor nodes) first. Here, an upper limit of neighbor size, denoted as n_u, will be set to control the scale of transaction subgraph. If the neighbor size of a_i is less than the threshold, all neighbors and all transactions between them will be extracted. Otherwise, we calculate and sort the total transaction volume between target account and its neighbors, and select the top-n_u neighbors. We assume that the larger the transaction volume, the higher the correlation between the two accounts. The above extraction mechanism can also be used to sample k-hop transaction neighbors from the $(k-1)$-hop neighbors. Therefore, the scale of one-order/two-order subgraph of target account will not exceed $n_u/(n_u)^2$ normally.

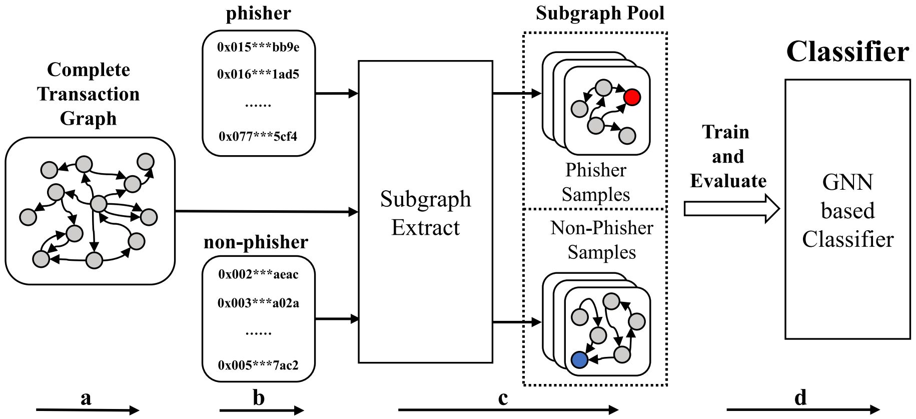

Fig. 1. The schematic depiction of our framework. The complete workflow proceeds as follows: a) modeling the transaction network; b) sampling the labeled accounts; c) extracting the subgraphs centered on target accounts; d) training and evaluating using GNNs.

The above process constructs the account (node) sets V, the transaction (edge) volume set E_v and the transaction (edge) frequency set E_t. Next, we briefly introduce two datasets which are named ETHG and EOSG and construct feature matrix X for them.

- **ETHG** It is from Xblock[1] which is a blockchain data platform for academic research. There is an account list that contains 1660 phisher accounts and

[1] http://xblock.pro/.

1700 non-phisher accounts with their 2-hop transaction records in Ethereum. Based on this, we filter the Contract-Account (CA) which will be considered as the contract calling feature of Externally-Owned-Accounts (EOA). And according to the span of the block where the transaction record is located, we collect all CAs from 0 to 10,000,000 blocks, filter them via the calling amount, and retain the top 14885 finally. After that, we construct the feature matrix of contract calling (cc) $X_{cc} \in \mathbb{R}^{n \times 14885}$, and each EOA has a 14885 dimension vector to represent their calling situation about those CAs.

- **EOSG** It is collected by [6]. They integrate and model the on-chain data of EOSIO: Enhanced Money Flow Graph (EMFG) which contains the transactions between accounts including timestamps and volume, Enhanced Account Creation Graph (EACG) which contains account creation tree data, Enhanced Contract Invocation Graph (ECIG) which contains smart contract calling data, and a list of labeled accounts which contains 229,907 normal accounts and 63863 bot-like accounts. Similarly, we extract the subgraph graph and contract calling features from EMFG and ECIG respectively, and construct the feature matrix of contract calling (cc) $X_{cc} \in \mathbb{R}^{n \times 1213}$. Further, we consider the account name restriction mechanism of EOSIO and add three kinds of node labels to expand features, since that the type of neighbors can also express the transaction pattern of the account. The three labels are the general account which consists of 12 characters, the auction account which is less than 12 characters but does not contain the character '.', and the sub-account of auction account which combines '.' with auction account name as the suffix. On the other hand, the neighbor extraction will stop at the system account whose name begins with 'EOSIO.'. Because the behavior pattern of the current center account has nothing to do with the transactions between other further accounts and system accounts. We construct the feature matrix of node label (nl) $X_{nl} \in \mathbb{R}^{n \times 3}$.

In summary, the feature matrix is $X = X_{cc} \in \mathbb{R}^{n \times 14885}$ for ETHG and $X = X_{cc} \oplus X_{nl} \in \mathbb{R}^{n \times 1216}$ for EOSG where $\oplus$ is concatenation operation.

3.3 Graph Neural Networks

By viewing the accounts interaction as graph data (i.e., transaction graph), recent deep learning methods for graph structural data, such as graph neural network (GNNs) [31–33], can be utilized to learn transaction pattern representation that can be fed to downstream machine learning models for phishing account detection. In this section, we will present the details of employing GNNs to obtain transaction pattern representation.

GNNs learn the representations of nodes by leveraging both the graph structure and node/edge features. This is done by a neighborhood aggregation function that iteratively takes the representation of all neighbors together with the graph structure as input, and outputs the aggregate representation of target node. The aggregation function can be defined using Graph Convolution layer

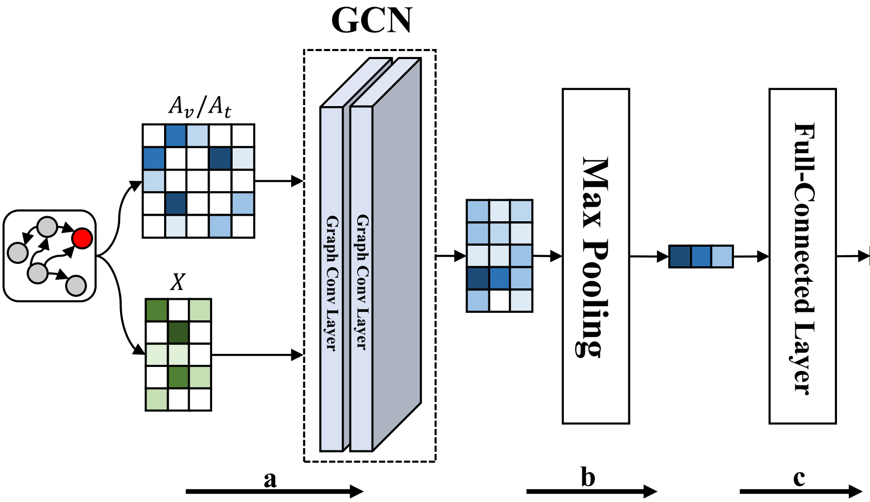

Fig. 2. The architecture of I^{2}BGNN model. a) A_v (A_t) and X are captured from subgraph extraction and sent to graph convolution network; b) the max-pooling layer is used to compress the aggregated node representations to obtain the whole graph representation; c) the graph representation is used to predict the subgraph (account) label.

[31], Graph Attention layer [32], or any general message passing layer [33]. Formally, a graph convolution network (GCN) model follows the following rule to aggregate the feature of neighbors:

$$H^{(l)} = \sigma(\hat{A}H^{(l-1)}W^{(l-1)}), \tag{1}$$

where $H^{(l-1)} \in \mathbb{R}^{n\times k}$ is a matrix containing the k-dimensional representation of n nodes in the $(l-1)$-th layer, σ is the activation function (typically ReLU), $\hat{A}$ is a symmetric normalization of A and can be defined as:

$$\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}, \quad \tilde{A} = A + I_n, \quad \tilde{D} = \mathrm{diag}(\sum_{j=0}^{n}\tilde{A}_{ij}), \tag{2}$$

where $\tilde{A}$ is an $n \times n$ adjacency matrix of the graph with self connections added, $\tilde{D}$ is a degree diagonal matrix. After l layer of computation, the node representations $H^{(l)}$ is able to capture the information within their l-hop neighborhoods.

Generally, GCN model is used to learn the node representations in semi-supervised node classification. A 2-layer GCN model with softmax function can be formulated as:

$$Z = \mathrm{softmax}(\hat{A} \cdot \mathrm{ReLU}(\hat{A}XW^{(0)})W^{(1)}) \tag{3}$$

where $Z \in \mathbb{R}^{n\times y}$ is the prediction probability distribution and y is the dimension of node labels. $W^{(0)}$ and $W^{(1)}$ are the input-to-hidden and hidden-to-output weights, respectively.

3.4 I^{2}BGNN

We now present the details of proposed I^{2}BGNN for identity inference on blockchain. For graph classification, the pooling operations aggregate node representations from the final iteration to obtain the whole graph's representation. By stacking the pooling layer and fully-connected layer after 2-layer GCN, the basic graph classification model for identity inference can be constructed as follows:

$$Z = \text{softmax}(\text{MaxPooling}(ReLU(\hat{A} \cdot \text{ReLU}(\hat{A}XW^{(0)})W^{(1)}))W^{(2)} + b) \qquad (4)$$

Note that we use the max pooling to obtain the whole graph's representation. The model architecture of I^{2}BGNN is shown in Fig. 2. In a transaction subgraph, each node represents an account and each directed edge represents transaction flow that contains information about transaction volume and frequency. For the input layer of GCN, we first initialize the node representations using their attributions in transaction subgraph. Specifically, the node attributions include contract calling information (cc) and distinctive node-label (nl), as mentioned in Sect. 3.2, and we initialize node representation as $H^{(0)} = X$.

4 Experiment

4.1 Dataset

For EOSG dataset, we filter the labeled account list in term of subgraph size and obtain over 20,000 available accounts. Then, 1000 accounts per label are selected randomly for the follow-up experiments. The detailed dataset properties are given in Table 1. Finally, each dataset is split into training and testing sets with a proportion of 1:1, and they will be resplit 3 times using different random seeds. We report the average accuracy across all trials.

Table 1. Dataset properties. $|G|$ is the number of subgraphs in dataset, $Avg.|V|$ is the average number of nodes per graph, $Avg.|E_{di}|$ is the average number of edges per directed graph, $Avg.|E_{ud}|$ is the average number of edges per undirected graph which is transformed from corresponding directed graph, $|F|$ is the dimension of node features, $|Y|$ is the number of classes for labels.

Dataset	$\lvert G\rvert$	$Avg.\lvert V\rvert$	$Avg.\lvert E_{di}\rvert$	$Avg.\lvert E_{ud}\rvert$	$\lvert F\rvert$	$\lvert Y\rvert$	**Label bias**
ETHG	3266	80	239	222	14885	2	0.99
EOSG	2000	260	4250	3212	1216	2	1

4.2 Baseline

Since we implement identity inference with a graph classification pattern, we compare our framework with several SOTA graph classification algorithms including SF [34], Graph2vec [25], Netlsd [35] and FGSD [36]. The first two are graph embedding methods and the last two are graph kernel methods. Graph2vec extends the document embedding methods to graph classification and learns a distributed representation of the whole graph via document embedding neural networks. SF performs graph classification by spectral decomposition of the graph Laplacian, i.e., it relies on spectral features of the graph. Netlsd performs graph classification by extracting compact graph signatures that inherit the formal properties of the Laplacian spectrum. FGSD calculates the Moore-Penrose spectrum of the normalized laplacian and uses the histogram of the spectral features of this spectrum to represent the whole graph.

4.3 Experiment Setting

During subgraph extraction, the direction of edges in subgraph is determined by the transaction flow. However, during the experiments, we find that symmetric adjacency matrix of subgraph usually outperforms directed adjacency matrix. Therefore, we transform the directed adjacency matrix into a symmetric adjacency matrix by adding its transpose to itself. Other settings for models and datasets are as follows:

Method Settings. For all the four baseline methods, we set the embedding dimension to 128, and use default settings for other parameters. Further, we implement graph classification by using the following machine learning classifiers: Support Vector Machine (SVM) with radial basis kernel, k-Nearest Neighbors classifier (KNN) and Random Forest classifier (RF). As for I^{2}BGNN, we apply two layers of GCNs with output dimensions both equal to 128, and set the maximum number of eopchs to be 50, the batch size to be 30 and dropout to be 0.3.

Metric Settings. Both the datasets have two classes, so we evaluate the results of binary classification by precision, recall and F1-Score.

Table 2. Results of identity inference. The top-2 best results are highlighted in bold.

Method		Dataset					
		EOSG			ETHG		
		F1	Precision	Recall	F1	Precision	Recall
Graph2vec	SVM	0.8223	0.8132	0.8317	0.6487	0.7564	0.5678
	KNN	0.6171	0.9820	0.4499	0.5705	0.5709	0.5701
	RF	0.7637	0.8155	0.7180	0.6104	0.7355	0.5216
SF	SVM	0.9428	0.9222	0.9643	0.6287	0.6405	0.6173
	KNN	0.9089	0.9081	0.9098	0.6238	0.6401	0.6083
	RF	0.9333	0.9166	0.9507	0.6908	0.7056	0.6766
Netlsd	SVM	0.8730	0.8574	0.8891	0.7067	0.6869	0.7276
	KNN	0.8406	0.8417	0.8396	0.6774	0.6884	0.6667
	RF	0.8845	0.8625	0.9077	0.6702	0.6782	0.6623
FGSD	SVM	0.9617	0.9534	0.9701	0.7206	0.6810	0.7650
	KNN	0.9469	0.9404	0.9534	0.7161	0.6750	0.7625
	RF	0.9578	0.9579	0.9578	0.7372	0.7448	0.7297
I^{2}BGNN-v		**0.9940**	**0.9894**	**0.9986**	**0.8587**	**0.8190**	**0.9024**
I^{2}BGNN-t		**0.9950**	**0.9917**	**0.9983**	**0.8600**	**0.8697**	**0.8505**

4.4 Result and Discussion

Inference Performance. Table 2 reports the performance comparison between I^{2}BGNN and baselines, from which we can observe that I^{2}BGNN significantly outperforms other methods across the two datasets. Specifically, compared with baselines, our I^{2}BGNN achieves average improvement of 12%/19% in term of F1 on EOSG/ETHG. This may be due to the excellent expression ability of the graph convolution layer and the effectiveness of the features which are constructed by the contract calling information. In addition, these graph embedding and kernel methods which are based on spectral analysis are significantly better than Graph2vec. Their advantages are also reflected in the efficiency of model operation in experiments.

Furthermore, we investigate the influence of neighborhood depth and data division on the experimental results under various settings.

The Influence of Neighborhood Depth. Normally, the subgraph containing 3-hop neighbors will have a large scale, which leads to difficulties in feature learning. To further analyze the influence of different depth for subgraph extraction, we extract 1-hop and 2-hop neighbors to construct the subgraphs for each target account. Table 3 shows the properties of 1-order and 2-order subgraphs for two datasets. And Table 4 reports the performance comparison between 1-order and 2-order subgraphs using I^{2}BGNN. For EOSG, I^{2}BGNN with 1-order subgraph performs slightly better than that with 2-order subgraph. Actually, the 1-order transaction subgraph contains sufficient and effective characteristics of transaction behavior, while the larger scale of the 2-order subgraph leads to the redundancy of information. As for ETHG, the situation is just the opposite, I^{2}BGNN with 2-order subgraph outperforms that with 1-order subgraph. Obviously, the 1-order subgraph contains sparse transaction information, which is not conducive to inference, while the denser interactions in the 2-order subgraph facilitate behavior analysis and identity inference.

Table 3. The properties of 1-hop and 2-hop subgraphs.

Dataset	**Subgraph**	$Avg.\|V\|$	$Avg.\|E_{di}\|$	$Avg.\|E_{ud}\|$
EOSG	1-order	17	65	48
	2-order	260	4250	3212
ETHG	1-order	10	13	12
	2-order	80	239	222

Table 4. Results of I^{2}BGNN with different neighborhood depth.

Dataset	Method	1-order	2-order
EOSG	I^{2}BGNN-v	**0.9960**	0.9940
	I^{2}BGNN-t	**0.9980**	0.9950
ETHG	I^{2}BGNN-v	0.8356	**0.8587**
	I^{2}BGNN-t	0.8366	**0.8600**

The Influence of Data Split. Next, we analyze the sensitivity of models to different ratios of data split. Specifically, we vary the ratio of training set to testing set in {1:9, 1:7, 1:5, 1:3, 1:1, 3:1}. Figure 3 reports the inference results (F1) of different models with various proportion of training set. Obviously, for different ratios, our I^{2}BGNN holds the best performance compared with other graph classification models. In addition, with the increase of training data, the performances of all models are naturally improved.

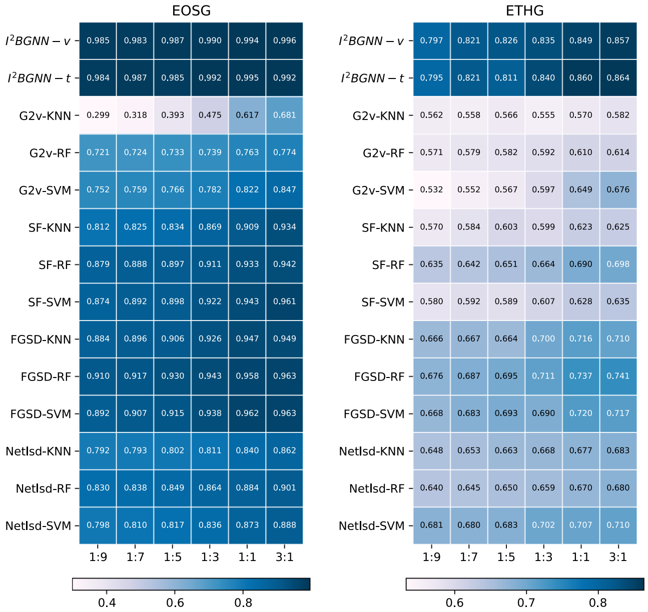

Fig. 3. Experimental results of different division ratios of the dataset on EOSG (left) and ETHG (right)

5 Conclusion

Traditional graph mining methods for identity inference are stuck in a dilemma where it is difficult to integrate scalability and end-to-end architecture into one model. In this work, we balance scalability and end-to-end architecture in model design. Specifically, we propose to learn the transaction subgraph centered on target account and transform the identity inference task on blockchain into graph classification pattern, resulting in a great reduction in resource consumption. Moreover, we design an end-to-end I^{2}BGNN model, which is capable of learning an effective graph representation. Finally, we conduct extensive experiments on two real blockchain datasets (EOSG and ETHG) to demonstrate the effectiveness of our proposed I^{2}BGNN. Experimental results show that the transaction pattern hidden in subgraph can actually reveal the account behavior, and our I^{2}BGNN achieves the outstanding performance in identity inference.

Acknowledgements. The authors would like to thank all the members in the IVSN Research Group, Zhejiang University of Technology for the valuable discussions about the ideas and technical details presented in this paper. This work was partially supported by the National Key R&D Program of China under Grant No. 2020YFB1006104, by the National Natural Science Foundation of China under Grant No. 61973273, by the Zhejiang Provincial Natural Science Foundation of China under Grant No. LR19F030001, by the Ministry of Public Security's Research Project "Research and Demonstration Application of Key Technologies of Criminal Social Network Model", and by the Special Scientific Research Fund of Basic Public Welfare Profession of Zhejiang Province under Grant LGF20F020016.

References

1. Vasek, M., Moore, T.: There's no free lunch, even using bitcoin: tracking the popularity and profits of virtual currency scams. In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 44–61. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47854-7_4
2. Wu, J., Lin, D., Zheng, Z., Yuan, Q.: T-edge: temporal weighted multidigraph embedding for ethereum transaction network analysis. arXiv preprint arXiv:1905.08038 (2019)
3. Chen, W., Zheng, Z., Cui, J., Ngai, E., Zheng, P., Zhou, Y.: Detecting ponzi schemes on ethereum: towards healthier blockchain technology. In: Proceedings of the 2018 World Wide Web Conference, pp. 1409–1418 (2018)
4. Bryans, D.: Bitcoin and money laundering: mining for an effective solution. Ind. LJ **89**, 441 (2014)
5. Fanusie, Y., Robinson, T.: Bitcoin laundering: an analysis of illicit flows into digital currency services. Center on Sanctions and Illicit Finance memorandum, January 2018
6. Huang, Y., et al.: Understanding (mis) behavior on the eosio blockchain. Proc. ACM Meas. Anal. Comput. Syst. **4**(2), 1–28 (2020)
7. Di Francesco, D., Maesa, A.M., Ricci, L.: Detecting artificial behaviours in the bitcoin users graph. Online Soc. Networks Media **3**, 63–74 (2017)
8. Kondor, D., Pósfai, M., Csabai, I., Vattay, G.: Do the rich get richer? an empirical analysis of the bitcoin transaction network. PloS One **9**(2), e86197 (2014)
9. Alqassem, I., Rahwan, I., Svetinovic, D.: The anti-social system properties: bitcoin network data analysis. IEEE Trans. Syst. Man Cybern. Syst. **50**(1), 21–31 (2018)
10. Tasca, P., Hayes, A., Liu, S.: The evolution of the bitcoin economy: extracting and analyzing the network of payment relationships. J. Risk Financ. (2018)
11. Bai, Q., Zhang, C., Xu, Y., Chen, X., Wang, X.: Evolution of ethereum: a temporal graph perspective. arXiv preprint arXiv:2001.05251 (2020)
12. Ferretti, S., D'Angelo, G.: On the ethereum blockchain structure: a complex networks theory perspective. Concurrency Comput. Practice Exp. **32**(12), e5493 (2020)
13. Huang, B., Liu, Z., Chen, J., Liu, A., Liu, Q., He, Q.: Behavior pattern clustering in blockchain networks. Multimed. Tools Appl. **76**(19), 20099–20110 (2017). https://doi.org/10.1007/s11042-017-4396-4
14. Ranshous, S., et al.: Exchange pattern mining in the bitcoin transaction directed hypergraph. In: Brenner, M. (ed.) FC 2017. LNCS, vol. 10323, pp. 248–263. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70278-0_16
15. Yuan, Z., Yuan, Q., Wu, J.: Phishing detection on ethereum via learning representation of transaction subgraphs. In: Zheng, Z., Dai, H.-N., Fu, X., Chen, B. (eds.) BlockSys 2020. CCIS, vol. 1267, pp. 178–191. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-9213-3_14
16. Yuan, Q., Huang, B., Zhang, J., Wu, J., Zhang, H., Zhang, X.: Detecting phishing scams on ethereum based on transaction records. In: 2020 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5. IEEE (2020)

17. Li, Y., Cai, Y., Tian, H., Xue, G., Zheng, Z.: Identifying illicit addresses in bitcoin network. In: Zheng, Z., Dai, H.-N., Fu, X., Chen, B. (eds.) BlockSys 2020. CCIS, vol. 1267, pp. 99–111. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-9213-3_8
18. Pham, T., Lee, S.: Anomaly detection in the bitcoin system-a network perspective. arXiv preprint arXiv:1611.03942 (2016)
19. Wu, J., et al.: Who are the phishers? phishing scam detection on ethereum via network embedding. IEEE Trans. Syst. Man Cybern. Syst. (2020)
20. Wu, J., Liu, J., Chen, W., Huang, H., Zheng, Z., Zhang, Y.: Detecting mixing services via mining bitcoin transaction network with hybrid motifs. IEEE Trans. Syst. Man Cybern. Syst. (2021)
21. Tam, D.S.H., Lau, W.C., Hu, B., Ying, Q.F., Chiu, D.M., Liu, H.: Identifying illicit accounts in large scale e-payment networks-a graph representation learning approach. arXiv preprint arXiv:1906.05546 (2019)
22. Weber, M., et al.: Anti-money laundering in bitcoin: experimenting with graph convolutional networks for financial forensics. arXiv preprint arXiv:1908.02591 (2019)
23. Lin, Y.-J., Wu, P.-W., Hsu, C.-H., Tu, I.-P., Liao, S.: An evaluation of bitcoin address classification based on transaction history summarization. In: 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pp. 302–310. IEEE (2019)
24. 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, pp. 701–710 (2014)
25. Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., Jaiswal, S.: graph2vec: learning distributed representations of graphs. arXiv preprint arXiv:1707.05005 (2017)
26. Xuan, Q., et al.: Subgraph networks with application to structural feature space expansion. IEEE Trans. Knowl. Data Eng. (2019)
27. Chen, L., Peng, J., Liu, Y., Li, J., Xie, F., Zheng, Z.: Phishing scams detection in ethereum transaction network. ACM Trans. Internet Technol. (TOIT) **21**(1), 1–16 (2020)
28. Ke, G., et al.: Lightgbm: a highly efficient gradient boosting decision tree. Adv. Neural Inf. Process. Syst. **30**, 3146–3154 (2017)
29. Phetsouvanh, S., Oggier, F., Datta, A.: Egret: extortion graph exploration techniques in the bitcoin network. In: 2018 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 244–251. IEEE (2018)
30. Zhang, R., Zhang, G., Liu, L., Wang, C., Wan, S.: Anomaly detection in bitcoin information networks with multi-constrained meta path. J. Syst. Archit. **110**, 101829 (2020)
31. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
32. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018). Accepted as poster
33. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: International Conference on Machine Learning, pp. 1263–1272. PMLR (2017)
34. de Lara, N., Pineau, E.: A simple baseline algorithm for graph classification. arXiv preprint arXiv:1810.09155 (2018)

35. Tsitsulin, A., Mottin, D., Karras, P., Bronstein, A., Müller, E.: Netlsd: hearing the shape of a graph. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2347–2356 (2018)
36. Verma, S., Zhang, Z.-L.: Hunt for the unique, stable, sparse and fast feature learning on graphs. In: NIPS, pp. 88–98 (2017)