Supplementary Material: Understanding the Influence of Extreme High-degree Nodes on Graph Anomaly Detection

Authors

Institutes

1 Details of Dataset Construction

Leveraging blockchain data, we have devised a pipeline that transforms raw data into a graph.

1.1 Raw Data

The blockchain is essentially a ledger, where each transaction is documented. In particular, our attention is directed towards the Ethereum blockchain, recognized as one of the most substantial and influential blockchain networks. Generally, transactions on Ethereum can be categorized into two primary segments: one pertains to transactions involving fungible tokens (ERC-20 Tokens), such as `Ether` and `USDC`, while the other concerns non-fungible tokens. Furthermore, the latter can be subcategorized into two distinct types, namely, ERC-721 NFTs and ERC-1155 NFTs, based on the contract standard governing their creation. In contrast to ERC-721, ERC-1155 emerges as a novel blockchain contract standard, exuding substantial promise, and poised to potentially surpass ERC-721, courtesy of the cost-efficiency and reduced transaction fees inherent in ERC-1155 [8]. We choose to establish a dataset based on transactions involving ERC-1155 NFTs. The motivation behind this choice lies in the recent introduction of ERC-1155, which is two years newer than ERC-721 and four years newer than ERC-20. The early stage of ERC-1155 increases the likelihood of generating node inequality and extreme high-degree nodes. Over time, the emergence of more high-degree nodes, particularly from entities like cryptocurrency exchanges, may contribute to diminishing the initially observed extreme inequality.

First, we request all ERC-1155 NFT token addresses from the Etherscan website [2] (These token addresses can be found in the GitHub repository [4]). Subsequently, we gather all transaction hashes linked to these NFT token addresses through the official `txlist` API function provided by Etherscan. That is, the token address corresponds to the `from` or `to` fields in a transaction. Lastly, using these transaction hashes as keys and indexes, we proceed to retrieve specific transaction details from the TokenView website [3] through an automated web driver in the Python selenium package. The Python script for this process is available in the GitHub repository [4]). We extract certain fields to compose the format of raw data (Table 1). In total, we obtained 11,009 NFT token addresses, and NFT transactions up to July 31, 2022, with a total number of 8,668,231.

Table 1: Format of raw NFT transaction data. Each row represents a transfer of a certain *Amount* of *Token* worth *Value* between the *From* and *To* addresses at a specific *Timestamp*, spending a transaction fee (*TxFee*), as recorded in the transaction (*TxHash*).

TxHash	*From*	*To*	*Token*	*Timestamp*	*Amount*	*Value*($)	*TxFee*($)
0xb5...b420	0x94...7293	0x6e...b7d3	0xd0...2430	20220730055230	1	78.52	2.23
0xa5...aeea	0x00...0000	0xd8...ac95	0xd0...2430	20220730055230	14	0.0	0.98
0xa2...bdf1	0x5b...1abb	0x4f...6580	0xd0...2430	20220730055138	1	0.0	0.33
...	...	...	...	...	...	...	...

1.2 Graph Construction

Upon acquiring the raw data, our goal is to construct ERC1155-NFTGraph (NFTGraph for short). To emphasize the unconcealment of NFTGraph, we utilize the information from transactions directly without any masking or transformation. Specifically, the *From* and *To* addresses, acting as the sending and receiving parties of a transaction, serve as the source and target nodes in the graph. An edge is established between the source and target nodes if tokens are transferred between them in a transaction. Interactions between nodes within the NFTGraph encompass multiple occurrences, often involving the transfer of tokens from source nodes to target nodes. However, for compatibility with the majority of GNN models, directed multi-edges between any two nodes are merged into a singular edge.

Utilizing the transactions linked to nodes, we emphasize several node features: *Address, OutAmount, OutValue, OutTxFee, InAmount, InValue*, and *InTxFee*. Aside from *Address*, which serves as the unique identifier for each node, the other features represent cumulants from transactions. Specifically, *OutAmount* and *InAmount* represent the cumulative tokens transferred from and to the current node, respectively. Similarly, *OutValue* and *InValue* indicate the cumulative transaction value when the current node is a source and target node separately and *OutTxFee* and *InTxFee* are the cumulative transaction fees paid by the current node as a sender and receiver, respectively.

Formally, the NFTGraph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes, and $\mathcal{E}$ is the set of all edges. $\mathcal{G}$ possesses a node attribute matrix $\mathbf{X} \in \mathcal{R}^{|\mathcal{V}| \times 6}$ and an adjacency matrix $\mathbf{A} \in \mathcal{R}^{|\mathcal{V}| \times |\mathcal{V}|}$. $\mathbf{A}_{uv} = 1$ if there exists an edge e in $\mathcal{E}$ between u and v, where $u, v \in \mathcal{V}$. Otherwise, $\mathbf{A}_{uv} = 0$.

Moreover, we present a variant of NFTGraph achieved through extraction. By extracting approximately 20,000 of the most active nodes while excluding isolated nodes, we form NFTGraph-Tiny, leading to a substantial reduction in the size of the original graph. This is executed with the recognition that certain GNNs may encounter challenges in handling extensive graphs within resource-constrained environments, a scenario often encountered in certain anomaly detection methods.

1.3 Labeling Suspicious Nodes

First, we define *ground-truth fraudulent nodes* that are the account addresses involved in a variety of fraudulent schemes published by previous researchers, encompassing Ponzi schemes [7] and phishing scams [5]. We collect an approximate total of 6,000 ground-truth fraudulent account addresses (Data sources and all ground-truth fraudulent account addresses can be found in the GitHub repository [4]), then we download all the transactions associated with these accounts through the Etherscan's official API [1]. Drawing inspiration from the methodologies delineated in prior works [6], we label nodes that exhibit interactions with the fraudulent nodes exceeding a count of three instances as *suspicious nodes*. The reason behind this approach is based on the idea that these nodes warrant examination due to their extensive participation in transactions with the previously mentioned fraudulent nodes. This recurring pattern is uncommon for legitimate entities, suggesting a potential connection to the same entity controlling these nodes. Suspicious nodes aim to alleviate the notable imbalance, arising from the limitation of ground-truth fraudulent nodes aligning with the NFTGraph's node set.

2 Other Experimental Settings for Section 4.1

The dataset is divided into training, validation, and test sets in proportions of 40% / 20% / 40% (Questions dataset follows the original split: 50% / 25% / 25%). Training utilizes node labels from the training set (except for unsupervised models). GNN aggregations may utilize nodes from the validation and testing sets, but their labels are not leaked during the training phase. The validation set is used for hyperparameter selection, with 100 rounds of random hyperparameter search conducted to obtain the best AUROC on the testing set. The models run in the transductive settings. AUROC on the test sets are calculated for model comparison. The search space for hyperparameters and the optimal hyperparameters can be found in the GitHub repository [4].

3 Over-smoothing Metrics

Instance Information Gain (G_{Ins}): The features of a node to some extent determine its class label. Thus, by independently processing each node instance, the instance information gain G_{Ins} is defined as the amount of input feature information contained in the final representation. A smaller G_{Ins} indicates that the final representation is more independent of the input features, reflecting over-smoothing to some extent. The formal expression is as follows (Equation 1):

$$G_{Ins} = I(\mathcal{X}; \mathcal{H}) = \sum_{x_v \in \mathcal{X}, h_v \in \mathcal{H}} P_{\mathcal{X}\mathcal{H}}(x_v, h_v) \log \frac{P_{\mathcal{X}\mathcal{H}}(x_v, h_v)}{P_{\mathcal{X}}(x_v) P_{\mathcal{H}}(h_v)}, \tag{1}$$

where $\mathcal{X}$ and $\mathcal{H}$ represent the random variables of input features and final representation vectors, $P_{\mathcal{X}}$ and $P_{\mathcal{H}}$ represent their probability distributions, and $P_{\mathcal{XH}}$ represents the joint probability distribution.

Group Distance Ratio (R_{Group}): Generally, vectors representing nodes of the same category should be closer together, while vectors representing nodes of different categories should be farther apart. Therefore, the group distance ratio is defined as the ratio of inter-group distance to intra-group distance under Euclidean distance. The formal expression is as follows (Equation 2):

$$R_{Group} = \frac{\frac{1}{(C-1)^2}\sum_{i\neq j}\left(\frac{1}{|L_i||L_j|}\sum_{h_{iv}\in L_i}\sum_{h_{jv'}\in L_j}||h_{iv}-h_{jv'}||_2\right)}{\frac{1}{C}\sum_{i}\left(\frac{1}{|L_i|^2}\sum_{h_{iv},h_{iv'}\in L_i}||h_{iv}-h_{iv'}||_2\right)}, \tag{2}$$

where C represents the number of node categories, nodes of the same category are defined as the same group $L_i = \{h_{iv}\}$, h_{iv} represents the vector representation of node v with category label i. $||\cdot||_2$ denotes the L2 norm of a vector, and $|\cdot|$ denotes the cardinality of a set (i.e., the number of elements in the set). The numerator (denominator) of R_{Group} represents the average distance between any two nodes of different groups (within the same group). Typically, smaller intra-group distances and larger inter-group distances (R_{Group} large) indicate that over-smoothing is alleviated. A smaller R_{Group} means that the representations of all groups are mixed together, hindering the classification of nodes, and over-smoothing is more likely to occur.

4 Other Experimental Settings for Section 5.2

Hyperparameter Setting: To ensure a fair comparison between SNGNN and other GAD models, we adopt a uniform approach of employing 2-layer GNN models, with fixed dimensions for the hidden layers (h_dim). Specifically, based on the original dimensions of the datasets, the model is set with h_dim = 32 for NFTGraph and Reddit, and h_dim = 128 for Weibo and Questions. The remaining hyperparameters are optimized through 100 random searches.

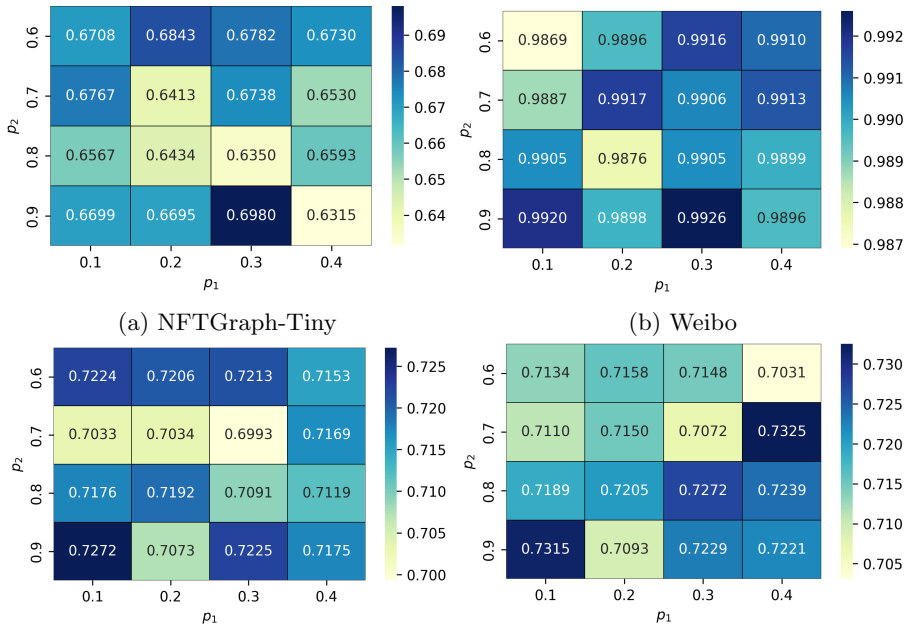
The remaining experimental settings not explicitly stated here follow those outlined in Section 4.1.

5 Parameter Sensitivity Analysis of SNGNN

The hyperparameters p_1 and p_2 represent the p_1 and p_2 percentiles of the LP output probabilities, determining the removal and addition of edges of SN in each training iteration. p_1 and p_2 indicate the strength and confidence of edge removal (addition), thus it is necessary to analyze the sensitivity to p_1 and p_2.

Figure 1 shows the detection AUROC of the SNGNN model under different values of p_1 and p_2 on four datasets. The optimal values of p_1 and p_2 for NFTGraph-Tiny and Weibo are (0.3, 0.9), for Reddit is (0.1, 0.9), and for Questions is (0.4, 0.7). The optimal values for these four datasets are not located in

the central area (i.e., $0.2 \leqslant p_1 \leqslant 0.3$, $0.7 \leqslant p_2 \leqslant 0.8$), indicating that for the edges of SN, the best practice is to ensure that the number of edge removals and additions is either relatively high or relatively low, or one is high while the other is low. If the number of edge removals and additions in each iteration is kept at a moderate level, it is difficult to achieve the optimal state.



(a) NFTGraph-Tiny (b) Weibo

(c) Reddit (d) Questions

Fig. 1: Parameter sensitivity.

References

1. Etherscan api document (2015), https://docs.etherscan.io.
2. Etherscan website (2015), https://etherscan.io.
3. Tokenview website (2017), https://tokenview.io/.
4. Anonymousdatacodehub (2023), https://github.com/AnonymousDataCodeHub/ERC1155-NFTGraph.
5. Chen, L., Peng, J., Liu, Y., Li, J., Xie, F., Zheng, Z.: Phishing scams detection in ethereum transaction network. ACM transactions on internet technology(TOIT) **21**(1), 1–16 (2020)
6. Chen, W., Wu, J., Zheng, Z., Chen, C., Zhou, Y.: Market manipulation of bitcoin: Evidence from mining the mt. gox transaction network. In: Proceedings of Ieee conference on computer communications(INFOCOM). pp. 964–972. IEEE (2019)

7. 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 world wide web conference(WWW). pp. 1409–1418 (2018)
8. Tan, Y., Wu, Z., Liu, J., Wu, J., Zheng, Z., Chen, T.: Bubble or not: Measurements, analyses, and findings on the ethereum erc721 and erc1155 non-fungible token ecosystem. arXiv preprint (2023)