Group-based Fraud Detection Network on e-Commerce Platforms

Jianke Yu
Zhejiang Gongshang University
Alibaba Group
jiankey.zjgsu@gmail.com

Hanchen Wang(✉)
Zhejiang Gongshang University
University of Technology Sydney
hanchen.wang@uts.edu.au

Xiaoyang Wang
University of New South Wales
xiaoyang.wang1@unsw.edu.au

Zhao Li
Zhejiang University
Hangzhou Link2Do Technology
lzjoey@gmail.com

Lu Qin
University of Technology Sydney
lu.qin@uts.edu.au

Wenjie Zhang
University of New South Wales
wenjie.zhang@unsw.edu.au

Jian Liao
Alibaba Group
jian.liaoj@alibaba-inc.com

Ying Zhang
Zhejiang Gongshang University
University of Technology Sydney
ying.zhang@uts.edu.au

ABSTRACT

Along with the rapid technological and commercial innovation on the e-commerce platforms, there are an increasing number of frauds that bring great harm to these platforms. Many frauds are conducted by organized groups of fraudsters for higher efficiency and lower costs, which are also known as group-based frauds. Despite the high concealment and strong destructiveness of group-based fraud, there is no existing research work that can thoroughly exploit the information within the transaction networks of e-commerce platforms for group-based fraud detection. In this work, we analyze and summarize the characteristics of group-based frauds, based on which we propose a novel end-to-end semi-supervised Group-based Fraud Detection Network (GFDN) to support such fraud detection in real-world applications. Experimental results on large-scale e-commerce datasets from Taobao and Bitcoin trading datasets show the superior effectiveness and efficiency of our proposed model for group-based fraud detection on bipartite graphs.

CCS CONCEPTS

• **Information systems** → **Data mining**.

KEYWORDS

Fraud Detection, Graph Neural Network, Bipartite Graph

ACM Reference Format:
Jianke Yu, Hanchen Wang(✉), Xiaoyang Wang, Zhao Li, Lu Qin, Wenjie Zhang, Jian Liao, and Ying Zhang. 2023. Group-based Fraud Detection Network on e-Commerce Platforms. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3580305.3599836

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599836

1 INTRODUCTION

Along with the increasing popularity, e-commerce platforms become more and more susceptible to fraudulent attacks, especially group-based frauds. These fraudulent attacks are usually conducted by groups of fraudsters (crowd workers) on the e-commerce platforms by creating fake links for efficiency and effectiveness purposes. The fraudulent attacks not only affect the platform's reputation but also influence the user experience and even lead to the loss of platform users.

One common fraud approach in Taobao platform is the "Ride Item's Coattails" attack [51] which creates fake clicks by groups of fraudsters to establish the deceptive correlation between popular products and low-quality products, and hence promotes the recommendation of the low-quality products to other customers. Figure 1 illustrates an example of the "Ride Item's Coattails" attack. Fraudsters (denoted as u_5-u_7) intentionally click the popular products (denoted as v_1-v_3) and the target low-quality product (denoted as v_4) at the same time in order to boost the sale of product v_4. Therefore, in Figure 1, the edges in red (solid lines) are regarded as fraudulent clicks. Another fraudulent attack approach is Sockpuppet-based Targeted Attack on Reviewing Systems (STARS) [57] attack. Similarly, the STARS attack aiming at the review systems of platforms is usually conducted by groups of fraudsters, which initiates fake ratings of target products, thus changing (usually improving) the rating of the target products and fraudulently promoting the products to other legitimate users. The fraudsters also rate normal products to imitate the behaviors of legitimate users, which increases the difficulty of detecting STARS attacks.

The research works [33, 37, 47, 51, 85] on fraud detection usually model the relationships between customers and products (*e.g.,* clicks, purchases and reviews) in e-commerce platform as an attributed bipartite graph. We have observed the following characteristics of group-based frauds on attributed bipartite graphs: (1) the subgraphs containing fraudsters and targets usually have high cohesion; (2) the fraudsters usually organize communities to

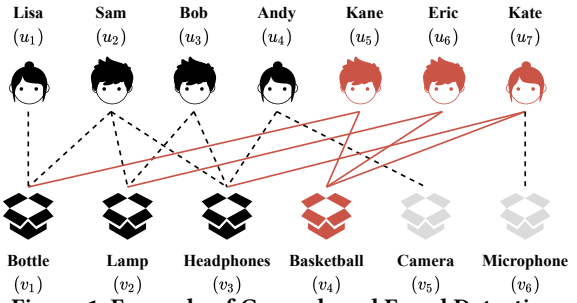


Figure 1: Example of Group-based Fraud Detection

conduct the frauds; (3) the attributes in the graph, such as the number of purchases, and the ground-truth labels of fraudulent links and fraudsters are of great help for fraud detection. Several methods [33, 37, 38, 47, 51, 79, 85] have been proposed for group-based fraud detection based on these characteristics, but none of them fully explores all these characteristics. For instance, in database literature, the solutions [51] are usually based on the cohesive subgraph mining [14, 35, 48, 56, 61, 69, 77, 90], such as biclique and (α, β)-core detection. However, these methods cannot utilize the attribute and label information, and some of them suffer from NP-completeness. Several fraud detection methods [33, 37, 47, 79, 85] in machine learning and data mining literature are proposed for leveraging the attribute and label information. Nevertheless, the heavy reliance on label information [33, 38, 47, 79] and the requirement of manual parameter setting [32, 85] limit the applicability of these methods in real datasets where only partial label information is available. Many other algorithms [10, 46, 53, 57, 64, 73, 74] utilize iterative learning, belief propagation and vertex ranking techniques and try to preserve graph topology information to uncover fraudsters. Other algorithms detect fraudulent attacks based on user behavior [31, 39, 63, 78, 80–82] or location information [44, 59, 75, 87] of vertices. However, due to the insufficient utilization of global topological and attribute information, these methods generally have limited performance. In addition to the aforementioned existing algorithms, community detection methods [23, 26, 65, 67, 68] are potential solutions that do not require label information. Unfortunately, community information is generally overlooked by existing works for fraud detection.

Motivated by the limitations of the existing methods, in this paper, we propose an end-to-end semi-supervised model Group-based Fraud Detection Network, namely GFDN, for group-based fraud detection on attributed bipartite graphs. Specifically, our model consists of two main parts: a structural feature generation module and a community-aware fraud detection network. With the carefully designed feature generation module, GFDN adaptively exploits the structural and attribute information of the bipartite graphs with the database techniques. A novel community-aware *Bipartite Deep Clustering Network* is proposed in our model to capture the group fraud behavior based on the attribute and high-order structural information. In this network, the community detection part can find potential fraudster communities and assist the model in fraud detection. Fraud detection can be modeled as an edge or vertex classification problem depending on the type of fraud. A multi-task learning mechanism is designed to train GFDN with joint objectives of fraud and fraudster detection for better group-based fraud detection capability. Besides, the carefully designed framework and training objectives also enable GFDN to be trained with partially available and imbalanced labels. The contribution of this paper can be summarized as follows:

- The proposed GFDN is a novel end-to-end model which adaptively utilizes the cohesive subgraph distribution information, structural information, attribute information and community information in the attributed bipartite graph for group-based fraud detection based on techniques in both database and machine learning literature.
- Extensive experiments are conducted for the fraud detection of "Ride Item's Coattails" attack and STARS attack on real-life datasets. The results of these experiments demonstrate the significant performance improvement (at least 13.83% and 3.09% improvement *w.r.t.* F1-score in the previous task, respectively) of GFDN compared with the existing methods on group-based fraud detection. We also conduct an in-depth analysis to evaluate the effectiveness of each component in GFDN.

2 BACKGROUND AND RELATED WORK

In this section, we introduce the important definitions and the problem statement we investigated. Then we present related works.

2.1 Preliminaries

We aim at the group-based fraud detection on e-commerce platforms which are modeled as attributed bipartite graphs in this paper. The definition of an attributed bipartite graph is as follows:

Definition 2.1 (Attributed Bipartite Graph). An attributed bipartite graph is denoted as $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \boldsymbol{X}_{\mathcal{U}}, \boldsymbol{X}_{\mathcal{V}})$, where $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ and $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ are two disjoint vertex sets; $\mathcal{E} \in \mathcal{U} \times \mathcal{V}$ is the edge set, $\mathcal{E}$ can be weighted in some types of graphs (*e.g.*, evaluation system graph); $\boldsymbol{X}_{\mathcal{U}}$ and $\boldsymbol{X}_{\mathcal{V}}$ are the attribute feature matrices for the vertex sets $\mathcal{U}$ and $\mathcal{V}$.

(α, β)-core in a bipartite graph is important for many fraud detection approaches, which is defined as follows:

Definition 2.2 ((α, β)-core). Given a bipartite graph $\mathcal{G}$ and integers $\alpha, \beta \in \mathbb{Z}^+$, (α, β)-core of $\mathcal{G}$ is denoted as $\mathcal{G}'$ which consists of two vertex sets $\mathcal{U}' \subseteq \mathcal{U}$ and $\mathcal{V}' \subseteq \mathcal{V}$. The (α, β)-core $\mathcal{G}'$ is a maximal bipartite subgraph induced by $\mathcal{U}' \cup \mathcal{V}'$ from $\mathcal{G}$ in which all the vertices in $\mathcal{U}'$ have degrees at least α and all the vertices in $\mathcal{V}'$ have degrees at least β.

Please note that the corresponding attribute feature matrices in (α, β)-core $\mathcal{G}'$ are denoted as $\boldsymbol{X}'_{\mathcal{U}}$ and $\boldsymbol{X}'_{\mathcal{V}}$ respectively.

Problem Statement. In this paper, we aim to design an end-to-end learning-based model for group-based fraud detection on attributed bipartite graphs. Specifically, depending on the type of fraud attacks, we aim to find the fraudulent clicks or fraudulent users in the e-commerce platforms, *i.e.,* to detect the fake links $\mathcal{E}_{att} \subset \mathcal{E}$ created by the groups of fraudsters or detect the fraudulent vertices (users) $\mathcal{U}_{att} \subset \mathcal{U}$ in the attributed bipartite graphs.

When there is no ambiguity, in this paper, we use the customer-product graph as an example of the attributed bipartite graph for ease of presentation. It should be noted that fraud detection of "Ride Item's Coattails" attack and STARS attack can be modeled as an edge classification problem and a vertex classification problem, respectively.

2.2 Related Works

In this section, we introduce the closely related works of group-based fraud detection. Specifically, we introduce the works proposed for classification algorithms, cohesive subgraph mining techniques and fraud detection methods.

Classification Algorithms. The group-based fraud detection can be modeled as an edge or vertex classification problem. Some research applies balance theory [21, 34] and matrix decomposition [9, 21] to predict edge signs for bipartite graphs but have difficulty handling imbalanced labeled vertices. Works on knowledge graph [49] and recommender system [30, 54] can solve edge sign prediction. Other methods include graph neural network [27, 42, 72, 88] and graph embedding [25, 66] for edge sign prediction on unipartite [60] or bipartite graphs [17, 18, 29, 89], but cannot utilize community information. Existing vertex classification methods [19, 43, 50, 62, 86] focus on analyzing vertex features and sharing neighborhood information to solve the problem, but cannot identify fraudster behavior and have limited performance in detecting savvy frauds.

Cohesive Subgraph Mining. Finding cohesive subgraphs on a bipartite graph, such as biclique [11, 90], k-bitruss [76], bi-triangle [83], (α, β)-core [56], δ-quasi-biclique [58] and k-biplex [84], *etc*, is widely used for community detection. However, it is challenging to use attribute and label information in these algorithms. In addition, there are some learning-based models for community mining [15, 41, 70]. However, they cannot be trivially applied for fraud detection.

Fraud Detection. To the best of our knowledge, RICD [51] is the state-of-the-art method for "Ride Item's Coattails" attack. RICD employs near-biclique to determine the group of fraudsters and then detect the frauds. However, RICD totally ignores attribute information and requires manual adjustments. As for STARS attack detection, RTV [57] is the state-of-the-art method. Thanks to the full use of rating information, this method can effectively detect fraudsters. However, the supervised variant of this method, named RTV-SUP, fails to utilize label information well, but directly employs the results of unsupervised learning as features and performs a simple supervised learning method (*e.g.,* logistic regression and random forest) to detect fraudsters. Click farming is another type of group-based fraud, which aims to generate enormous fake traffic for the target by groups of fraudsters. There have been many algorithms [24, 36, 37, 52, 85, 91] proposed for click farming detection. However, they mainly focus on feature engineering and ignore abundant structural information in graphs.

3 MODEL

In this section, we introduce details of our model GFDN. The framework of GFDN is illustrated in Figure 2. GFDN is developed in an end-to-end fashion. We first utilize the (α, β)-core distribution to initialize the structural features. A novel community-aware *Bipartite Deep Clustering Network* (BDCN) is proposed to capture the characteristics of the group-based frauds. Furthermore, a multi-task learning mechanism is designed to train GFDN with joint objectives of fraud and fraudster detection for better group-based fraud detection capability and generalisability.

3.1 Structural Feature Initialization

The initial features contain the inherent attribute information and the structural information in the attributed bipartite graph. In this section, we introduce how the structural features are initialized.

As observed in our experiments and [51], the fraudsters' behavior is closely related to their degree. Intuitively, fraudsters have a relatively high degree due to their creation of fake links (clicks or reviews). In this work, we choose (α, β)-core distribution to obtain the structural information. As defined in Section 2.1, α limits the minimum degree of one vertex set, *e.g.,* customer vertex set, and β limits the minimum degree of the other vertex set, *e.g.,* product vertex set. By varying values of α and β, (α, β)-core is utilized to obtain subgraphs with different sparsity effectively. These subgraphs are used to generate expressive structural features of the whole graph.

Due to the characteristics of group-based fraud, for example, in the customer-product graph, the degrees of fraudsters and target products are relatively high. With the increasing values of α and β, fewer vertices are retained in (α, β)-core. Consequently, we set the upper thresholds α_τ^+ and β_τ^+ to put more attention to the vertices with relatively high degrees. Meanwhile, when the values of α and β are relatively low, the size of (α, β)-core changes remarkably with varying α and β. Therefore, we set lower thresholds α_τ^- and β_τ^- to put more attention to keeping the discrimination of structural features for each vertex. (α, β)-core varying α and β can reveal the activeness of customers and the popularity of products, which would be of great benefit for group-based fraud detection. As a result, we query all (α, β)-cores with $\alpha_\tau^- \le \alpha \le \alpha_\tau^+$ and $\beta_\tau^- \le \beta \le \beta_\tau^+$. For each vertex, the structural feature $\boldsymbol{x}_s \in \{0,1\}^{d_0}$ is generated with dimension $d_0 = (\alpha_\tau^+ - \alpha_\tau^- + 1) \times (\beta_\tau^+ - \beta_\tau^- + 1)$ equals to the number of (α, β)-cores queried. Each Boolean entry in $\boldsymbol{x}_s$ indicates whether the vertex belongs to the corresponding (α, β)-core. Finally, the structural features $\boldsymbol{X}_{(\mathcal{U},s)} \in \{0,1\}^{|\mathcal{U}| \times d_0}$ and $\boldsymbol{X}_{(\mathcal{V},s)} \in \{0,1\}^{|\mathcal{V}| \times d_0}$ are obtained for both vertex sets in the attributed bipartite graph, where $|\mathcal{U}|$ and $|\mathcal{V}|$ are the number of vertices in both sets respectively.

For the obtained structural features, we utilize the learnable weights to allow autonomous adjustment of the importance of the (α, β)-core distribution. Specifically, the weights $\boldsymbol{W}_{(\mathcal{U},s)} \in \mathbb{R}^{1 \times d_0}$ and $\boldsymbol{W}_{(\mathcal{V},s)} \in \mathbb{R}^{1 \times d_0}$ are used to generate $\hat{\boldsymbol{X}}_{(\mathcal{U},s)}$ and $\hat{\boldsymbol{X}}_{(\mathcal{V},s)}$:

$$\hat{\boldsymbol{X}}_{(\mathcal{U},s)} = \boldsymbol{X}_{(\mathcal{U},s)} \odot (\boldsymbol{I}_{\mathcal{U}} \boldsymbol{W}_{(\mathcal{U},s)}),\ \hat{\boldsymbol{X}}_{(\mathcal{V},s)} = \boldsymbol{X}_{(\mathcal{V},s)} \odot (\boldsymbol{I}_{\mathcal{V}} \boldsymbol{W}_{(\mathcal{V},s)}), \quad (1)$$

where $\odot$ denotes the Hadamard (element-wise) product, $\boldsymbol{I}_{\mathcal{U}} = \boldsymbol{1}^{|\mathcal{U}| \times 1}$ and $\boldsymbol{I}_{\mathcal{V}} = \boldsymbol{1}^{|\mathcal{V}| \times 1}$ are the matrices whose elements are all 1 with dimensions $|\mathcal{U}| \times 1$ and $|\mathcal{V}| \times 1$ respectively.

3.2 Fraudster Community Detection

To conduct fraudulent attacks, fraudsters need to create a great number of fake links (clicks or reviews) in a relatively short time window. To reduce the cost and improve efficiency, fraudsters usually organize communities or register a great number of accounts to carry out such links. Identifying these communities can significantly assist the group-based fraud detection, which is overlooked in previous works [36, 37]. However, community information is usually unavailable in real-life data. Therefore, unsupervised techniques are exploited for community detection.

Inspired by SDCN [15], we propose a community-aware graph neural network for bipartite graphs, named *Bipartite Deep Clustering Network* (BDCN). SDCN achieves SOTA performance for clustering on the unipartite graphs. However, SDCN cannot be trivially adapted to the bipartite graphs, and it cannot exploit the available label information. Besides, since there is usually no strong

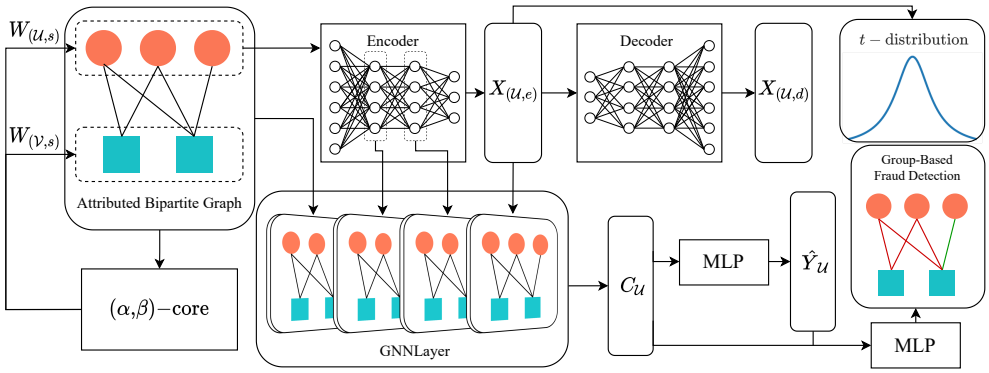


Figure 2: The Framework of GFDN

correlation between the products involved in the fraud, clustering is only required for customers in the graph. Motivated by these limitations, we propose BDCN in this work, which can detect the communities based on both structural and attribute information.

BDCN has two major components: autoencoder and graph neural network (GNN). With autoencoder [13], BDCN can be trained in a self-supervised fashion while preserving the information from the input features. The autoencoder uses the concatenation of weighted structural features and the attribute features of the customer vertices, *i.e.,* $\tilde{\boldsymbol{X}}_{\mathcal{U}} = \hat{\boldsymbol{X}}_{(\mathcal{U},s)}||\boldsymbol{X}_{\mathcal{U}}$ as input. The encoder is modeled as a multilayer perceptron (MLP) with input $\tilde{\boldsymbol{X}}_{\mathcal{U}}$. At each neural layer, the computation is as follows:

$$\boldsymbol{X}^{(l+1)}_{(\mathcal{U},a)} = \sigma(\boldsymbol{X}^{(l)}_{(\mathcal{U},a)}\boldsymbol{W}^{(l)}_{e} + \boldsymbol{b}^{(l)}_{e}), \tag{2}$$

where σ is an activation function, $\boldsymbol{W}^{(l)}_{e}$ and $\boldsymbol{b}^{(l)}_{e}$ are weight matrix and bias at l-th layer of encoder, $\boldsymbol{X}^{(l)}_{(\mathcal{U},a)}$ is the output of l-th layer of the encoder, and $\boldsymbol{X}^{(0)}_{(\mathcal{U},a)} = \tilde{\boldsymbol{X}}_{\mathcal{U}}$. The output of the last encoder layer is considered the encoded customer feature matrix, *i.e.,* $\boldsymbol{X}_{(\mathcal{U},e)} = \boldsymbol{X}^{(L_e)}_{(\mathcal{U},a)}$, where L_e denotes the number of layers of the encoder; $\boldsymbol{X}_{(\mathcal{U},e)}$ is the encoded features of $\tilde{\boldsymbol{X}}_{\mathcal{U}}$.

Similarly, the decoder is also modeled as an MLP. The intermediate process of the decoder can be expressed as:

$$\boldsymbol{X}^{'(l+1)}_{(\mathcal{U},a)} = \sigma(\boldsymbol{X}^{'(l)}_{(\mathcal{U},a)}\boldsymbol{W}^{(l)}_{d} + \boldsymbol{b}^{(l)}_{d}), \tag{3}$$

where $\boldsymbol{X}^{'(0)}_{(\mathcal{U},a)} = \boldsymbol{X}_{(\mathcal{U},e)}$; $\boldsymbol{W}^{(l)}_{d}$ and $\boldsymbol{b}^{(l)}_{d}$ are weight matrix and bias for l-th layer in the decoder. The final decoding result is the output of the last layer of the decoder, *i.e.,* $\boldsymbol{X}_{(\mathcal{U},d)} = \boldsymbol{X}^{'(L_d)}_{(\mathcal{U},a)}$. The autoencoder is designed to extract the expressive low-dimension representations of vertices that contains valuable attributes for downstream fraud detection. The objective of the autoencoder is to minimize the difference between $\boldsymbol{X}_{(\mathcal{U},d)}$ and $\tilde{\boldsymbol{X}}_{\mathcal{U}}$. Mean square error (MSE) [12] is used as the self-supervised loss function for the autoencoder:

$$\mathcal{L}_{ae} = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}||\boldsymbol{x}_{ud} - \tilde{\boldsymbol{x}}_{u}||^2_2, \tag{4}$$

where $\boldsymbol{x}_{ud}$ and $\tilde{\boldsymbol{x}}_{u}$ are representations of vertex u from the matrices $\boldsymbol{X}_{(\mathcal{U},d)}$ and $\tilde{\boldsymbol{X}}_{\mathcal{U}}$.

With the obtained encoded representations $\boldsymbol{X}_{(\mathcal{U},e)}$, we would like to get the community information of the customer vertices. $\boldsymbol{X}_{(\mathcal{U},e)}$ is fed into a specific clustering algorithm such as K-means [28], mean shift [20], *etc*, to detect the clusters in $\mathcal{U}$. It is worth noting that since the encoded representations $\boldsymbol{X}_{(\mathcal{U},e)}$ contains both structural and attribute information, the clustering is based on these two types of information. We denote the cluster center vectors, *i.e.,* representations of the center vertices in each cluster, as $\boldsymbol{C}_{\mathcal{K}}$, where $\mathcal{K}$ is the number of clusters. Please note that $\boldsymbol{C}_{\mathcal{K}}$ consists of the encoded representations for cluster centers and is trainable. The performance of our model is not sensitive to the choice of clustering algorithms, and K-means is chosen for clustering in this work. For the i-th customer vertex and the j-th cluster, the Student's t-distribution [71] is selected as the kernel for measuring the similarity between representation $\boldsymbol{x}_{ue,i}$ and the cluster center vector $\boldsymbol{c}_j$. It can be calculated by the following equation:

$$q_{ij} = \frac{(1+||\boldsymbol{x}_{ue,i} - \boldsymbol{c}_j||^2/\omega)^{-\frac{\omega+1}{2}}}{\sum_{j'}(1+||\boldsymbol{x}_{ue,i} - \boldsymbol{c}_{j'}||^2/\omega)^{-\frac{\omega+1}{2}}}, \tag{5}$$

where ω is the degree of freedom of the Student's t-distribution; $\boldsymbol{x}_{ue,i}$ is the i-th customer vertex representation in $\boldsymbol{X}_{(\mathcal{U},e)}$; $\boldsymbol{c}_j \in \boldsymbol{C}_{\mathcal{K}}$ is j-th cluster center representation; q_{ij} denotes the similarity between i-th customer and j-th cluster center which can also be considered as the probability of assigning i-th customer to j-th cluster. We define $\boldsymbol{Q} = [q_{ij}] \in \mathbb{R}^{|\mathcal{U}|\times\mathcal{K}}$ as the matrix of these similarities. We aim to make the customer vertices closer to the cluster centers, *i.e.,* the cluster assignment with high confidence, thus improving the cluster cohesion. For this purpose, we compute the target similarity distributions with the following equation:

$$p_{ij} = \frac{q^2_{ij}/\sum_i q_{ij}}{\sum_{j'} q^2_{ij'}/\sum_i q_{ij'}}. \tag{6}$$

We define $\boldsymbol{P} = [p_{ij}] \in \mathbb{R}^{|\mathcal{U}|\times\mathcal{K}}$ as the normalized matrix of $\boldsymbol{Q}$, *i.e.,* enlarging the similarities between similar instances and reduce the similarities between dissimilar instances, using squared pairs. We use Kullback-Leibler (KL) divergence [45] as the loss function to minimize the difference between $\boldsymbol{P}$ and $\boldsymbol{Q}$:

$$\mathcal{L}_c = KL(\boldsymbol{P}||\boldsymbol{Q}) = \frac{1}{|\mathcal{U}|\mathcal{K}}\sum_{i}^{|\mathcal{U}|}\sum_{j}^{\mathcal{K}} p_{ij}\log\frac{p_{ij}}{q_{ij}}. \tag{7}$$

By minimizing $\mathcal{L}_c$, the customer representations become more distinguishable for clustering. As a result, optimized by $\mathcal{L}_{ae}$ and $\mathcal{L}_c$, the output of the encoder $\boldsymbol{X}_{(\mathcal{U},e)}$ is regarded as community representations of customer vertices.

Autoencoder can generate high-quality community representations for the customer vertices. However, since only MLPs are utilized, the autoencoder preserves the structural and attribute

information only from the initial features but loses adjacency information in the bipartite graph. Besides, the information on the product vertices is ignored in the task. Therefore, we employ the graph neural network (GNN) along with the autoencoder in BDCN to further capture the abundant information within the attributed bipartite graph.

GNNs have been successfully used in various applications because of their powerful capability of preserving information within graphs. Existing graph neural networks usually adopt an aggregate and combine scheme as follows:

$$\boldsymbol{X}_u^{(l)} = \mathcal{COM}^{(l)}(\boldsymbol{X}_u^{(l-1)}, \mathcal{AGG}^{(l)}\{\boldsymbol{X}_{u'}^{(l-1)}; u' \in N(u)\}), \tag{8}$$

where $\boldsymbol{X}_u^{(l)}$ is the representation of vertex u at l-th layer of the graph neural network, $N(u)$ is the set of neighbors of vertex u, $\mathcal{AGG}$ is the aggregation operation that iteratively updates the representation of a vertex by aggregating the representations of its neighbors, and $\mathcal{COM}$ is the combine operation that updates the representation of vertex u by the aggregated representations and its own representation $\boldsymbol{X}_u^{(l-1)}$ from the previous layer.

However, popular GNNs, such as GCN [42], GAT [72], GraphSAGE [27], *etc*, are designed for unipartite graphs rather than specifically designed for bipartite graphs, and hence cannot be employed directly in this task. Besides, the initial features for vertices in different vertex sets, *i.e.,* customers and products, contain different attributes and have different dimensions, which is not considered in the popular graph neural network architectures.

To address these issues, we design a novel GNN-based model to allow aggregation on the attributed bipartite graph while preserving the attribute and structural information. The input features for product vertices are as follows:

$$\tilde{\boldsymbol{X}}_{\mathcal{V}} = \hat{\boldsymbol{X}}_{(\mathcal{V},s)} || (\boldsymbol{X}_{\mathcal{V}} \boldsymbol{W}_{\mathcal{V}} + \boldsymbol{b}_{\mathcal{V}}), \tag{9}$$

where $\hat{\boldsymbol{X}}_{(\mathcal{V},s)}$ is the structural features introduced in Section 3.1; $\boldsymbol{X}_{\mathcal{V}}$ is the attribute features for products; $\boldsymbol{W}_{\mathcal{V}}$ and $\boldsymbol{b}_{\mathcal{V}}$ are weight matrix and bias used to map $\boldsymbol{X}_{\mathcal{V}}$ in order to match the dimension of $\tilde{\boldsymbol{X}}_{\mathcal{V}}$ with that of $\tilde{\boldsymbol{X}}_{\mathcal{U}}$ to allow the aggregation between two sets of vertices.

In each GNN hidden layer, we superimpose the customer representation from each encoder layer onto that from the previous GNN hidden layer as the input. More specifically, at the first GNN layer, we pass $\tilde{\boldsymbol{X}}_{\mathcal{U}}$ and $\tilde{\boldsymbol{X}}_{\mathcal{V}}$ through the GNN network:

$$\boldsymbol{H}^{(1)} = \sigma(\mathrm{g}^{(1)}(\tilde{\boldsymbol{X}}_{\mathcal{U}}, \tilde{\boldsymbol{X}}_{\mathcal{V}}, \boldsymbol{A})), \tag{10}$$

where $\mathrm{g}^{(l)}(\cdot)$ is l-th layer GNN, $\boldsymbol{A} \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{V}|}$ is the adjacency matrix; $\boldsymbol{H}^{(1)}$ is the output of first GNN layer. Then, the output of the corresponding encoder layer is superimposed onto the customer features before passing to the next GNN hidden layer:

$$\boldsymbol{H}^{(l+1)} = \sigma(\mathrm{g}^{(l+1)}((\boldsymbol{X}_{(\mathcal{U},a)}^{(l)} \oplus \boldsymbol{H}_{\mathcal{U}}^{(l)}), \boldsymbol{H}_{\mathcal{V}}^{(l)}, \boldsymbol{A})), \tag{11}$$

where $\oplus$ denotes the element-wise summation, $\boldsymbol{H}_{\mathcal{U}}^{(l)}$ and $\boldsymbol{H}_{\mathcal{V}}^{(l)}$ are hidden representations for $\mathcal{U}$ and $\mathcal{V}$ at l-th layer respectively. Finally, the customer features obtained from the last hidden layer are superimposed onto the output of encoder $\boldsymbol{X}_{(\mathcal{U},e)}$, *i.e.,* customer community representations, which are then fed into the output GNN layer to obtain community affiliation results:

$$\boldsymbol{C}_{\mathcal{U}} = sigmoid(\mathrm{g}^{(L_g)}((\boldsymbol{X}_{(\mathcal{U},e)} \oplus \boldsymbol{H}_{\mathcal{U}}^{(L_g-1)}), \boldsymbol{H}_{\mathcal{V}}^{(L_g-1)}, \boldsymbol{A})\boldsymbol{M}_{\mathcal{U}}), \tag{12}$$

where L_g is the number of GNN layers, $\boldsymbol{M}_{\mathcal{U}}$ is the mask to select the customer vertices; $\boldsymbol{C}_{\mathcal{U}} \in \mathbb{R}^{|\mathcal{U}|\times\mathcal{K}}$ is community affiliation representation, each entry of which is the probability that a customer belongs to a community. It is common for one customer to be in multiple communities simultaneously, *i.e.,* one customer can have similar behavior with multiple groups of customers. Therefore, we choose Sigmoid as the output layer activation function.

The performance of GFDN is not sensitive to the choice of GNN backbone. In this paper, we choose the architecture of GraphSAGE [27] to ensure efficiency on large-scale graphs. Specifically, each layer can be symbolized as follow:

$$\boldsymbol{X}_u^{(l)} = \boldsymbol{W}_1^{(l)} \boldsymbol{X}_u^{(l-1)} + \boldsymbol{W}_2^{(l)} \cdot \mathrm{mean}_{u' \in N(u)} \boldsymbol{X}_{u'}^{(l-1)}, \tag{13}$$

where $\boldsymbol{W}_1^{(l)}$ and $\boldsymbol{W}_2^{(l)}$ are weight matrices at l-th layer.

3.3 Training Objective

As introduced in Section 2.2, group-based fraud detection can be modeled as either edge classification or vertex classification problem depending on the type of fraud. To optimize GFDN for both types of frauds, we design a multi-task training objective for edge and vertex classification.

Specifically, for the fraudulent operation detection tasks, such as "Ride Item's Coattails" attack detection, the objective is to perform the edge classification in attributed bipartite graphs. Since all frauds are conducted by fraudsters, training the model for fraudster detection, *i.e.,* vertex classification, can boost the performance of the main task. Please note that although all fraud links come from fraudsters, not all links generated by the fraudsters are fraudulent. Thus the vertex classification results cannot be trivially used for fraudulent operation detection. Meanwhile, for the fraudster detection tasks, such as STARS attack detection, the main objective of GFDN is the vertex classification, *i.e.,* classifying the customers into fraudulent and legitimate groups. In this case, edge classification is regarded as an auxiliary task to improve the optimization since the fraudulent links are only created by the fraudsters. As a result, in this work, we utilize multi-task learning with a joint training objective for both edge classification (fraud detection) and vertex classification (fraudster detection).

We first introduce how we perform the vertex classification. Based on the structural and attribute information, the community affiliation representation of customers $\boldsymbol{C}_{\mathcal{U}}$ has been obtained by Equation 12. $\boldsymbol{C}_{\mathcal{U}}$ is then used to predict the fraudsters by a two-layer MLP as follows:

$$\hat{\boldsymbol{Y}}_{\mathcal{U}} = \delta(\mathrm{MLP}(\boldsymbol{C}_{\mathcal{U}})), \tag{14}$$

where $\hat{\boldsymbol{Y}}_{\mathcal{U}} \in \mathbb{R}^{|\mathcal{U}|\times 2}$ denotes the predicted probability of whether a customer is a legitimate user or a fraudster, and δ is Softmax function. If the predicted probability $\hat{\boldsymbol{y}}_u$ of vertex u is above a threshold $\tau_{\mathcal{U}}$, u will be considered a fraudster.

Similarly, we introduce how we perform edge classification. The representation of the edge e that connects customer u and product v is constructed as follows:

$$\boldsymbol{x}_{e,uv} = \hat{\boldsymbol{y}}_u || \boldsymbol{c}_u || \hat{\boldsymbol{x}}_{us} || \hat{\boldsymbol{x}}_{vs} || \boldsymbol{x}_u || \boldsymbol{x}_v, \tag{15}$$

where $\hat{\boldsymbol{y}}_u \in \hat{\boldsymbol{Y}}_{\mathcal{U}}$ is the predicted fraudster probability of u calculated by Equation 14; $\boldsymbol{c}_u \in \boldsymbol{C}_{\mathcal{U}}$ is community affiliation representation of u; $\hat{\boldsymbol{x}}_{us} \in \hat{\boldsymbol{X}}_{(\mathcal{U},s)}$ and $\hat{\boldsymbol{x}}_{vs} \in \hat{\boldsymbol{X}}_{(\mathcal{V},s)}$ are structural features of customer u and product v respectively; $\boldsymbol{x}_u \in \boldsymbol{X}_{\mathcal{U}}$ and $\boldsymbol{x}_v \in \boldsymbol{X}_{\mathcal{V}}$

are the attribute features of u and v. Specifically, for the fraudster detection tasks, *e.g.,* STARS attack, the fraudulent edges are only created by the fraudsters, and the predicted fraudster probability $\hat{\boldsymbol{y}}_u$ is highly related to the prediction of fraudulent edges and reduce the effectiveness for optimization of the auxiliary edge classification task. Therefore, for these tasks, $\boldsymbol{x}_{e,uv}$ is not concatenated with $\hat{\boldsymbol{y}}_u$ while the rest of the representation remains unchanged. With the concatenation in Equation 15, we obtain an edge feature matrix: $\boldsymbol{X}_{\mathcal{E}}$ for all edges in $\mathcal{G}$. Similarly, we also use two fully connected layers followed by the Softmax function to obtain the edge classification results:

$$\hat{Y}_{\mathcal{E}} = \delta(\text{MLP}(\boldsymbol{X}_{\mathcal{E}})). \tag{16}$$

Given an edge e, if the predicted probability $\hat{\boldsymbol{y}}_e$ is above a threshold $\tau_{\mathcal{E}}$, e will be regarded as a fraudulent edge.

With the vertex classification and edge classification results $\hat{Y}_{\mathcal{U}}$ and $\hat{Y}_{\mathcal{E}}$, and the ground-truth labels for vertices $Y_{\mathcal{U}}$ and edges $Y_{\mathcal{E}}$, one immediate loss function for these two tasks is the cross-entropy loss. However, since the vertex and edge labels are usually imbalanced, *e.g.,* most vertices and edges are labeled as legitimate ones, the direct utilization of cross-entropy will lead to a deteriorated performance. To increase the emphasis on the fraudsters and the fraudulent edges during the training phase, we employ focal loss [55] as the loss function for both tasks. Specifically, the loss function for vertex classification is as follows:

$$\begin{aligned}\mathcal{L}_l = \sum_{u\in\mathcal{U}} -[&\lambda_{lt}\boldsymbol{y}_u(1-(\hat{\boldsymbol{y}}_u))^{\gamma_l}\log(\hat{\boldsymbol{y}}_u)+\\ &\lambda_{lf}(1-\boldsymbol{y}_u)(1-(1-\hat{\boldsymbol{y}}_u))^{\gamma_l}\log(1-\hat{\boldsymbol{y}}_u)],\end{aligned} \tag{17}$$

where $\boldsymbol{y}_u$ denotes ground truth label of u, and $\hat{\boldsymbol{y}}_u$ denotes the predicted probability of whether u is a fraudster, λ_{lt}, λ_{lf} and γ_l are adjustable weighting parameters for training focus.

The loss function for edge classification is as follows:

$$\begin{aligned}\mathcal{L}_e = \sum_{e\in\mathcal{E}} -[&\lambda_{et}\boldsymbol{y}_e(1-(\hat{\boldsymbol{y}}_e))^{\gamma_e}\log(\hat{\boldsymbol{y}}_e)+\\ &\lambda_{ef}(1-\boldsymbol{y}_e)(1-(1-\hat{\boldsymbol{y}}_e))^{\gamma_e}\log(1-\hat{\boldsymbol{y}}_e)],\end{aligned} \tag{18}$$

where $\boldsymbol{y}_e \in Y_{\mathcal{E}}$ is the ground-truth label of edge e, $\hat{\boldsymbol{y}}_e \in \hat{Y}_{\mathcal{E}}$ is the predicted probability of edge e as a fraudulent edge, λ_{et}, λ_{ef}, and γ_e are parameters to adjust the training weights.

The parameters of GFDN are jointly optimized in one optimizer with the following unified loss function:

$$\mathcal{L} = \omega_{ae}\mathcal{L}_{ae} + \omega_c\mathcal{L}_c + \omega_l\mathcal{L}_l + \omega_e\mathcal{L}_e, \tag{19}$$

where ω_{ae}, ω_c, ω_l and ω_e are coefficients of $\mathcal{L}_{ae}$, $\mathcal{L}_c$, $\mathcal{L}_l$ and $\mathcal{L}_e$. Therefore, our proposed GFDN can be trained to focus on the vertex or edge classification by adjusting these weighting parameters.

4 EXPERIMENT

In this section, we first introduce the settings of the experiments, *i.e.,* the details of datasets, and the baseline algorithms. To evaluate the accuracy, efficiency, and importance of each component in GFDN, we report the results of the comparison with state-of-the-art baselines, ablation study, and parameter sensitivity analysis. In detail, two group-based fraud detection tasks, *i.e.,* "Ride Item's Coattails" and STARS attack detection, are selected as the representatives of two major types of fraud detection, *i.e.,* fraudulent links detection and fraudster detection in the experiment. We also conduct an in-depth analysis to demonstrate how the structural feature generation and clustering module influence the performance of GFDN.

Table 1: Datasets for "Ride Item's Coattails" Attack Detection

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{V}\rvert$	% Fraudulent	% Legitimate
TB	3,085,653	996,090	381,611	0.62%	3.53%
TC	1,050,000	532,345	239,840	2.86%	11.43%

Table 2: Datasets for STARS Attack Detection

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{V}\rvert$	% Fraudulent	% Legitimate
Alpha	24,186	3,286	3,754	3.10%	4.20%
OTC	35,592	4,814	5,858	3.70%	2.80%

We concentrate on verifying the accuracy of GFDN. The implementation, hardware details, and other setting details are summarized in Appendix A. The code is available at [7]. Additional experiments and analyses can be found in Appendix B. Five evaluation metrics are used: F1-score, accuracy, AUC, precision and recall. The specific definitions of these metrics can be found in Appendix A.4.

4.1 Experimental Setup

Dataset for "Ride Item's Coattails" attack detection. We aim to perform group-based fraud detection to detect the "Ride Item's Coattails" attack [51], *i.e.,* the groups of fraudsters create fake clicks (*i.e.,* edges) with popular and low-quality target products to boost the sales of target products, in the experiments. The experiments are conducted on two real-life customer-product datasets, **TC** [4] and **TB**. **TC** is an open source dataset used for Tianchi "Ride Item's Coattails" attack prediction competition [5]. **TB** is a large-scale attributed bipartite customer-product graph on the Alibaba e-commerce platform Taobao. The fraudulent labels in these two datasets are obtained basically by expert labelling, *i.e.,* labeled manually by experts. The details are reported in Appendix A.5.

The expert labeling in Appendix A.5 can be very accurate in finding frauds, but it also has significant drawbacks, *e.g.,* it requires a lot of labor costs and low labeling efficiency, which makes it challenging to achieve a large number of labels; due to the additional consideration of more feature information, feature filtering in dense subgraphs approach does not significantly improve labeling efficiency; red team attack simulation approach is not cost-effective due to its high consumption of money and resources. Therefore, GFDN dramatically helps the Taobao platform find more fraudulent attacks with lower consumption.

The specific statistics of both datasets are presented in Table 1 where % Fraudulent and % Legitimate denote the percentage of labeled fraudulent and legitimate edges respectively. Please note that only partial edges have labels in both datasets. Edges linked to the vertices with only one degree are labeled as normal edges. The evaluation is conducted in a transductive setting, and all structural and attribute information is available during both training and test process. Among the labeled edges, we apply the stratified sampling to select 10% edges in the test set and the rest labeled edges in the training set.

Dataset for STARS attack detection. STARS attack is also a typical group-based fraud on attributed bipartite graphs. Fraudsters in the e-commerce system create a great number of fake accounts to rate targets with numerous fake ratings. To test the performance of GFDN and the baseline methods, we use Bitcoin Alpha [46] and Bitcoin OTC [46] with the pre-processing introduced in RTV [57] as the datasets for STARS attack detection in our experiments.

These two datasets are user-to-user trust networks of Bitcoin users trading using Alpha platform and OTC platform, and they are made bipartite by splitting each user into a 'rater' with all its

Table 3: Effectiveness Evaluation Results for "Ride Item's Coattails" Detection

	TB Data					TC Data				
	F1	Acc	AUC	Pre	Recall	F1	Acc	AUC	Pre	Recall
LPA	0.2737	0.4627	0.5517	0.1715	0.6785	0.2056	0.4284	0.5276	0.1219	0.6557
SBGNN	0.4789	0.8228	0.7947	0.4279	0.5438	0.3676	0.8074	0.7666	0.2900	0.5018
BiGI	0.5359	0.8540	0.8491	0.5097	0.5649	0.4039	0.8292	0.8044	0.3331	0.5129
SIHG	0.6449	0.8709	0.8692	0.5470	0.7853	0.5947	0.8771	0.8985	0.4735	0.7992
Tianchi	0.6446	0.8752	0.9342	0.5606	0.7581	0.5364	0.8717	0.9107	0.4527	0.6583
RICD	0.6518	0.8405	0.9063	0.4834	**1.0000**	0.4784	0.8482	0.7474	0.3906	0.6171
(α, β)-core	0.8081	0.9449	0.8757	0.8417	0.7770	0.6348	0.8907	0.8696	0.5093	0.8423
FRAUDAR	0.2580	0.1481	0.4963	0.1483	0.9927	0.2020	0.1124	0.4981	0.1124	**0.9961**
CF1	0.2407	0.7698	0.5532	0.2371	0.2445	0.1620	0.7981	0.5253	0.1523	0.1731
CF2	0.4675	0.7603	0.7376	0.3497	0.7052	0.3588	0.6837	0.7277	0.2326	0.7844
Naive	0.8109	0.9473	0.9844	0.8736	0.7565	0.6397	0.9090	0.9516	**0.7816**	0.5414
GFDN-S	0.6867	0.9202	0.9653	0.8284	0.5864	0.6122	0.8783	0.9342	0.4780	0.8514
GFDN-F	0.9212	0.9754	0.9886	0.8821	0.9639	0.6401	0.8976	0.9287	0.5302	0.8076
GFDN-L	0.9398	0.9813	0.9964	0.9050	0.9775	0.7015	0.9192	0.9654	0.6014	0.8417
GFDN-C	0.9423	0.9821	0.9967	0.9086	0.9785	0.7048	0.9226	0.9646	0.6181	0.8198
GFDN	**0.9522**	**0.9853**	**0.9974**	**0.9254**	0.9806	**0.7226**	**0.9242**	**0.9713**	0.6154	0.8752

outgoing edges and each 'product' with all incoming edges. The specific statistics of both datasets are presented in Table 2, where % Fraudulent and % Legitimate represent the percentage of known fraudsters and normal raters among all raters in the original dataset, respectively. We refer to the work of RTV and simulate STARS attack on these two datasets. More specifically, among the parameters chosen in RTV's work, we select the percentage of sockpuppet accounts as 30%, choose the number of fake ratings per sockpuppet account as 10, and fix the number of target products to 100. All these sockpuppet accounts are considered fraudsters, and these fraudsters' initial fairness and rating reliability are randomly distributed. These fraudsters rate the target products with the highest score, and for other products, their ratings are randomly obtained by the normal distribution of existing ratings. Then, to maximize the benefits of RTV, we also provide trusted raters and verified raters. We generate 100 trusted raters and mark 500 existing raters as verified raters to ensure that datasets in our experiments are identical to those in RTV's work. Trusted raters have the highest fairness, and their ratings are also randomly obtained by the normal distribution of the existing ratings. We set a fairness value of 0.5 for rater vertices without initial fairness, a goodness value of 0 for product vertices without initial goodness, and a reliability value of 1 for ratings without initial reliability. For attribute features, we consider vertex type, degree, and initial fairness/goodness as vertex features and ratings as additional edge features. After completing the above pre-processing, we apply the stratified sampling to select 10% of the vertices in the test set and the other 90% of the vertices in the training set. The further setting of RTV can be found in Appendix A.1.

Compared Methods. To demonstrate the performance of our proposed model, we compare GFDN with the state-of-the-art baseline methods. Generally, the baseline methods can be classified into two major categories: learning-based and pattern-based methods.

Learning-based methods utilize machine learning techniques to perform fraud detection. We compare the following state-of-the-art methods:

- **Label Propagation Algorithm (LPA) [68].** LPA is a fast semi-supervised algorithm to assign labels to a graph.
- **Signed Infomax Hyperbolic Graph (SIHG) [60].** SIHG is a signed link prediction method based on hyperbolic graph neural network.
- **BiGI [18].** BiGI is a novel bipartite graph embedding method based on the local-global infomax for recommendation and link prediction on bipartite graphs.
- **Signed Bipartite Graph Neural Networks (SBGNN) [34].** SBGNN is a representation learning algorithm for vertices in signed bipartite graphs based on the balance theory.
- **Tianchi [6].** The algorithm with the best performance on Tianchi "Ride Item's Coattails" attack prediction competition. The algorithm predicts the fraud edges by the semi-supervised model with MLPs and batch normalization, whose inputs are the attribute features of vertices.

The pattern-based methods aim to utilize the structural information in the graph to detect potential frauds. Due to the characteristics of the group-based fraud detection, *i.e.,* the fraud edges usually form cohesive subgraphs, and the cohesive subgraph detection methods are used for the fraud detection. In this work, the following pattern-based methods are compared:

- **(α, β)-core.** Given α and β, the edges in the computed (α, β)-core are regarded as frauds. In our experiments, we enumerate the choices of α and β and report the best result.
- **RICD [51].** RICD ((α, k_1, k_2)-biclique) is proposed in [51] to detect the "Ride Item's Coattails" attack. Following the settings in [51], we set $\alpha = 1$ for this method. The code used is from the public project [2].

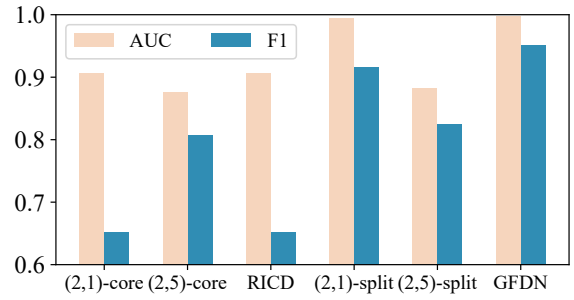


Figure 3: Comparison with Pattern-based Algorithms on TB

Please note that, following the setting in [51], all edges in cohesive subgraphs found by (α, β)-core and RICD are regarded as fraudulent links.

Besides, we also compare GFDN with the following methods, which achieve state-of-the-art performance in fraud detection:

- **FRAUDAR [1, 32].** FRAUDAR is designed to detect frauds in bipartite graphs in a camouflage-resistant way.
- **CF1 [85].** CF1 is a state-of-the-art algorithm for click farming detection. CF1 utilizes LOF [16] to filter data and self-supervised k-means to perform the prediction.
- **CF2 [37].** CF2 is also designed for click farming detection. The algorithm labels the vertices with a label propagation method and then combines SVM and neural networks for training and prediction.
- **RTV-SUP [57].** RTV is the state-of-the-art STARS attack detection algorithm. This algorithm can find users with abnormal behavior by fully utilizing the credibility of vertices and ratings. RTV-SUP is a supervised variant of RTV.

To evaluate the effectiveness of each component in our model, the following variants of GFDN are also compared:

- **Naive.** The naive algorithm is a variant that directly uses initial features $\hat{\boldsymbol{x}}_{us}||\hat{\boldsymbol{x}}_{vs}||\boldsymbol{x}_u||\boldsymbol{x}_v$ as input and conducts the fraud detection with two fully connected neural layers.
- **GFDN-S.** Compared to GFDN, GFDN-S ignores $\hat{X}_{(\mathcal{U},s)}$ and $\hat{X}_{(\mathcal{V},s)}$ for fraud detection to test the effectiveness of structural feature generation module in our model.
- **GFDN-F.** Compared to GFDN, $\boldsymbol{X}_{\mathcal{U}}$ and $\boldsymbol{X}_{\mathcal{V}}$ are not involved in fraud detection to test the importance of attribute features.
- **GFDN-L:** Compared to GFDN, this variant removes the $\hat{\boldsymbol{Y}}_{\mathcal{U}}$ from the edge representation $\boldsymbol{X}_{\mathcal{E}}$ to test the effectiveness of multi-task learning, specifically vertex classification.
- **GFDN-C.** Compared to GFDN, the community affiliation representations $\boldsymbol{C}_{\mathcal{U}}$ are not included in the edge representation $\boldsymbol{X}_{\mathcal{E}}$ to test the effectiveness of community information.

We keep the parameter settings of SIHG, BiGI and SBGNN the same as in the original papers. In addition, since none of the three sets of baselines can handle unlabeled data (*i.e.,* support semi-supervised learning), only the edges with labels are used in the train and test sets. For the pattern-based methods, we tune the parameters and report their best performance. We also keep the hyperparameters of Naive model, GFDN-S, GFDN-F, GFDN-L, and GFDN-C the same as that of GFDN, except for the number of epochs to ensure convergence.

4.2 Prediction Accuracy Evaluation on "Ride Item's Coattails" Attack Detection

In this section, we report the performance of compared methods for "Ride Item's Coattails" attack detection.

The experimental results of all models are presented in Table 3. Overall, our proposed GFDN outperforms all the other compared methods on both datasets. Specifically, compared with the second best-performed baseline, GFDN achieves an improvement over 17.83% and 13.83% on TB and TC in the F1-score metric.

Compared with learning-based methods. LPA has the worst performance because it totally ignores the attribute information. Although the structural information of the attributed bipartite graph is considered in BiGI and SBGNN, BiGI ignores the attribute information of products during aggregation, while SBGNN cannot handle label and graph sparsity, which leads to their non-competitive performance. Meanwhile, SIHG and Tianchi, which pay more attention to attribute feature utilization and processing, perform much better. However, these two algorithms cannot utilize the graph structural and community information which are crucial for group-based fraud detection. Thus they still have a significant performance gap with GFDN, *e.g.,* more than 47.65% improvement *w.r.t.* F1-score on TB dataset.

Compared with pattern-based methods. The pattern-based methods have the best performance among the existing compared methods, in spite of the fact that only the graph structural information is used. RICD detects the frauds by finding near-bicliques, which is too strict and thus cannot detect a great number of fraud edges that are linked to unpopular products, *i.e.,* vertices with relatively small degree. Therefore, RICD has a high AUC but a low F1-score and accuracy. Meanwhile, with appropriate α and β, (α, β)-core can achieve good performance by excluding most of the normal edges in the prediction. However, (α, β)-core still cannot detect frauds on unpopular products. On the other hand, because of the inability to utilize edge label information and attribute information, there is a considerable performance gap between (α, β)-core and GFDN.

Due to the superior performance of the pattern-based methods, we provide more comparison with the pattern-based algorithms on TB dataset for detailed analysis. Specifically, GFDN is compared with pattern-based methods varying values of their parameters, *e.g.,* values of α, β in (α, β)-core. Furthermore, we compare two methods named (α, β)-split. We first filter out the edges outside the (α, β)-core and regard them as legitimate edges. Then, the consequent predictions are made for edges inside the (α, β)-core using the prediction network in GFDN. The comparison results are presented in Figure 3.

From Figure 3, we can find that (2, 5)-core achieves better performance than (2, 1)-core. It indicates that the products which have a higher degree also have a higher probability of being involved in fraud. Moreover, RICD achieves the best performance with $k_1 = 1$, $k_2 = 2$, *i.e.,* the biclique with at least 1 customer vertex and 2 product vertices. With larger k_1, k_2 values, the accuracy of RICD decreases due to its strict filtering. Its optimal performance is achieved when it degenerates to (2, 1)-core. Thanks to GFDN's capability to utilize the information from structure, attributes and labels, it still outperforms the above methods.

Table 4: Effectiveness Evaluation Results for STARS Detection

	Alpha					OTC				
	F1	Acc	AUC	Pre	Recall	F1	Acc	AUC	Pre	Recall
FRAUDAR	0.3800	0.2626	0.5236	0.2346	**1.0000**	0.3780	0.2547	0.5183	0.2330	**1.0000**
RTV-SUP	0.8652	**0.9452**	0.8859	**0.9747**	0.7778	0.7010	0.8082	0.8736	0.5417	0.9931
(α, β)-core	0.7857	0.8767	0.9204	0.6471	**1.0000**	0.7784	0.8711	0.9167	0.6372	**1.0000**
Naive	0.8089	0.9018	0.9789	0.7222	0.9192	0.7937	0.8978	0.9508	0.7310	0.8681
GFDN	**0.8919**	**0.9452**	**0.9913**	0.8049	**1.0000**	**0.9231**	**0.9623**	**0.9746**	**0.8571**	**1.0000**

In (α, β)-split, since the graph has been segmented using structural information, the structural features $\hat{\boldsymbol{X}}_{(\mathcal{U},s)}$ and $\hat{\boldsymbol{X}}_{(\mathcal{V},s)}$ are not available for the network in (α, β)-split. Specifically, (2,1)-core and (2,5)-core are tested. It can be observed from Figure 3 that (2,5)-split is outperformed by (2,1)-split because (2,5)-split regards many fraudulent links as legitimate ones, hence limits the benefits from the prediction network. These two methods are outperformed by GFDN since they cannot fully use the structural information, *i.e.,* (α, β)-core distribution, for fraud detection.

Compared with fraud detection methods. FRAUDAR has poor performance, *e.g.,* 25.80% F1-score on TB, because it predicts the frauds based on subgraph density in which the abundant attribute and structural information are overlooked. CF1 and CF2 are the latest click farming detection models. However, CF1 cannot fully utilize edge label information, and CF2 does not leverage the graph structural information, which results in the worse performance of these two algorithms.

Compared with variants of GFDN. We conduct ablation experiments to illustrate the effectiveness of each part of GFDN. By mining structural, attribute and edge label information, the Naive model can achieve excellent accuracy, even though it mainly consists of simple neural networks. The accuracy of GFDN-S is only marginally higher than SIHG and Tianchi, and significantly lower than (α, β)-core. This result demonstrates that $\hat{\boldsymbol{X}}_{(\mathcal{U},s)}$ and $\hat{\boldsymbol{X}}_{(\mathcal{V},s)}$ can provide significant improvement for the task. The accuracy of GFDN-F is close to Naive but far behind GFDN, which shows the importance of attribute features. When $\hat{\boldsymbol{Y}}_{\mathcal{U}}$ or $\boldsymbol{C}_{\mathcal{U}}$ is unavailable, GFDN-L and GFDN-C suffer from slight decrease of accuracy compared with GFDN. It can be concluded that both the vertex classification learning objective and the community affiliation representations can improve the effectiveness of this kind of attack detection. Still, this improvement is not as remarkable as that brought by the utilization of structural and attribute features.

4.3 Performance of STARS Attack Detection

In this section, we report the evaluation results of GFDN on STARS attack detection task. In addition to the baseline method FRAUDAR, (α, β)-core and Naive, we also compare GFDN with the state-of-the-art fraudster detection method RTV [57]. Since GFDN is a semi-supervised model, we compare it with the supervised variant of RTV, named RTV-SUP for fairness. The detailed settings of this method are reported in Appendix A.1.

The experimental results are presented in Table 4. GFDN outperforms the baseline methods with a significant margin on this fraudster detection task.

Compared with pattern-based methods. (α, β)-core achieves a high recall score but a low precision score. This result indicates that the fraudster detection method that is only based on structural information may result in a great number of false positive predictions. In comparison, our proposed GFDN achieves a much better precision score which leads to superior performance in terms of F1 score, accuracy and AUC. This improvement is brought by the exploitation of the label and attribute information in GFDN.

Compared with fraudster detection methods. Specifically, RTV-SUP outperforms all other models except GFDN on Alpha dataset. Compared to RTV-SUP, GFDN achieves about 3% improvement in the F1-score metric, and about 11% improvement *w.r.t.* AUC. The advancement of GFDN mainly comes from the high recall score. Our proposed model can achieve 1 as the recall score, which means there is no false negative in the prediction of GFDN. This is an important characteristic of the fraudster detection methods, which cannot be satisfied in RTV-SUP. On OTC dataset, GFDN achieves an improvement over 30% *w.r.t.* F1-score metric compared with RTV-SUP. GFDN has a great margin over RTV-SUP on the precision value, which indicates the better capacity of GFDN in reducing the number of false positives compared with RTV-SUP.

Therefore, we can conclude that our proposed GFDN not only achieves state-of-the-art performance but is also applicable in real-life fraudster detection tasks.

4.4 Further Analyses

To further demonstrate the superiority of GFDN, we conduct query time comparison, in-depth effectiveness analysis and parameter sensitivity analysis (results reported in Appendix B). In the query time comparison part, we report the query time costs of the compared methods for fraud and fraudster detection. The in-depth effectiveness analysis illustrates how (α, β)-core works in our model. The parameter sensitivity analysis tests the performance of the model with different parameters. These experiments demonstrate the efficiency of GFDN, the importance of (α, β)-core in the model, and how sensitive the model is to its parameters.

5 CONCLUSION

In this paper, we investigate group-based fraud detection on e-commerce platforms. Based on the characteristics of such frauds, we carefully design a model, named GFDN, for group-based fraud detection. GFDN adaptively utilizes the structural, attribute and available label information in fraud detection. The experimental results on large-scale e-commerce data and Bitcoin trading data show a significant improvement in accuracy and the efficiency achieved by GFDN.

REFERENCES

[1] 2016. http://bhooi.github.io/code/camo.zip.
[2] 2018. https://github.com/hadisfr/biclique.
[3] 2020. https://github.com/boge-liu/alpha-beta-core.
[4] 2021. https://tianchi.aliyun.com/dataset/dataDetail?dataId=123862.
[5] 2021. https://tianchi.aliyun.com/competition/entrance/531925/introduction.
[6] 2021. https://tianchi.aliyun.com/forum/postDetail?spm=5176.12586969.1002.15.333a7671eOe7Cf&postId=316896.
[7] 2022. https://github.com/yujianke100/GFDN.
[8] 2022. https://github.com/swig/swig.
[9] Charu C Aggarwal, Yao Li, S Yu Philip, and Yuchen Zhao. 2017. On edge classification in networks with structure and content. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 187–190.
[10] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. 2013. Opinion Fraud Detection in Online Reviews by Network Effects. In *Proceedings of the Seventh International Conference on Weblogs and Social Media, ICWSM 2013, Cambridge, Massachusetts, USA, July 8-11, 2013*, Emre Kiciman, Nicole B. Ellison, Bernie Hogan, Paul Resnick, and Ian Soboroff (Eds.). The AAAI Press. http://www.aaai.org/ocs/index.php/ICWSM/ICWSM13/paper/view/5981
[11] Gabriela Alexe, Sorin Alexe, Yves Crama, Stephan Foldes, Peter L Hammer, and Bruno Simeone. 2004. Consensus algorithms for the generation of all maximal bicliques. *Discrete Applied Mathematics* 145, 1 (2004), 11–21.
[12] David M Allen. 1971. Mean square error of prediction as a criterion for selecting variables. *Technometrics* 13, 3 (1971), 469–475.
[13] Pierre Baldi and Zhiqin Lu. 2012. Complex-valued autoencoders. *Neural Networks* 33 (2012), 136–147.
[14] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. 2013. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *Proceedings of the 22nd international conference on World Wide Web*. 119–130.
[15] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. 2020. Structural deep clustering network. In *Proceedings of The Web Conference 2020*. 1400–1410.
[16] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 93–104.
[17] Jonathan Bryan and Pablo Moriano. 2021. Graph-Based Machine Learning Improves Just-in-Time Defect Prediction. *arXiv preprint arXiv:2110.05371* (2021).
[18] Jiangxia Cao, Xixun Lin, Shu Guo, Luchen Liu, Tingwen Liu, and Bin Wang. 2021. Bipartite graph embedding via mutual information maximization. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 635–643.
[19] Hao Chen, Zhong Huang, Yue Xu, Zengde Deng, Feiran Huang, Peng He, and Zhoujun Li. 2022. Neighbor enhanced graph convolutional networks for node classification and recommendation. *Knowl. Based Syst.* 246 (2022), 108594. https://doi.org/10.1016/j.knosys.2022.108594
[20] Yizong Cheng. 1995. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence* 17, 8 (1995), 790–799.
[21] Tyler Derr, Cassidy Johnson, Yi Chang, and Jiliang Tang. 2019. Balance in signed bipartite networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1221–1230.
[22] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).
[23] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.
[24] Saptarshi Ghosh, Bimal Viswanath, Farshad Kooti, Naveen Kumar Sharma, Gautam Korlam, Fabrício Benevenuto, Niloy Ganguly, and P. Krishna Gummadi. 2012. Understanding and combating link farming in the twitter social network. In *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012*, Alain Mille, Fabien Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab (Eds.). ACM, 61–70. https://doi.org/10.1145/2187836.2187846
[25] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.
[26] Roger Guimera, Marta Sales-Pardo, and Luís A Nunes Amaral. 2007. Module identification in bipartite and directed networks. *Physical Review E* 76, 3 (2007), 036102.
[27] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).
[28] John A Hartigan and Manchek A Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)* 28, 1 (1979), 100–108.
[29] Fang He, Feiping Nie, Rong Wang, Haojie Hu, Weimin Jia, and Xuelong Li. 2020. Fast semi-supervised learning with optimal bipartite graph. *IEEE Transactions on Knowledge and Data Engineering* 33, 9 (2020), 3245–3257.
[30] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.
[31] Bryan Hooi, Neil Shah, Alex Beutel, Stephan Günnemann, Leman Akoglu, Mohit Kumar, Disha Makhija, and Christos Faloutsos. 2016. BIRDNEST: Bayesian Inference for Ratings-Fraud Detection. In *Proceedings of the 2016 SIAM International Conference on Data Mining, Miami, Florida, USA, May 5-7, 2016*, Sanjay Chawla Venkatasubramanian and Wagner Meira Jr. (Eds.). SIAM, 495–503. https://doi.org/10.1137/1.9781611974348.56
[32] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. 2016. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 895–904.
[33] Benjamin Horne and Sibel Adali. 2017. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. In *Proceedings of the international AAAI conference on web and social media*, Vol. 11. 759–766.
[34] Junjie Huang, Huawei Shen, Qi Cao, Shuchang Tao, and Xueqi Cheng. 2021. Signed Bipartite Graph Neural Networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 740–749.
[35] Dmitry I Ignatov, Polina Ivanova, and Albina Zamaletdinova. 2018. Mixed integer programming for searching maximum quasi-bicliques. In *International Conference on Network Analysis*. Springer, 19–35.
[36] Cuixia Jiang, Jun Zhu, and Qifa Xu. 2020. Dissecting click farming on the Taobao platform in China via PU learning and weighted logistic regression. *Electronic Commerce Research* (2020), 1–20.
[37] Cuixia Jiang, Jun Zhu, and Qifa Xu. 2021. Which goods are most likely to be subject to click farming? An evidence from the Taobao platform. *Electronic Commerce Research and Applications* 50 (2021), 101107.
[38] Fang Jin, Edward Dougherty, Parang Saraf, Yang Cao, and Naren Ramakrishnan. 2013. Epidemiological modeling of news and rumors on twitter. In *Proceedings of the 7th workshop on social network mining and analysis*. 1–9.
[39] Parisa Kaghazgaran, James Caverlee, and Majid Alfifi. 2017. Behavioral Analysis of Review Fraud: Linking Malicious Crowdsourcing to Amazon and Beyond. In *Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017*. AAAI Press, 560–563. https://aaai.org/ocs/index.php/ICWSM/ICWSM17/paper/view/15659
[40] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980
[41] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).
[42] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl
[43] Emmanouil Krasanakis, Symeon Papadopoulos, and Ioannis Kompatsiaris. 2022. p2pGNN: A Decentralized Graph Neural Network for Node Classification in Peer-to-Peer Networks. *IEEE Access* 10 (2022), 34755–34765. https://doi.org/10.1109/ACCESS.2022.3159688
[44] Wataru Kudo, Mao Nishiguchi, and Fujio Toriumi. 2020. GCNEXT: graph convolutional network with expanded balance theory for fraudulent user detection. *Soc. Netw. Anal. Min.* 10, 1 (2020), 85. https://doi.org/10.1007/s13278-020-00697-w
[45] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.
[46] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and V. S. Subrahmanian. 2018. REV2: Fraudulent User Prediction in Rating Platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, Yi Chang, Chengxiang Zhai, Yan Liu, and Yoelle Maarek (Eds.). ACM, 333–341. https://doi.org/10.1145/3159652.3159729
[47] Srijan Kumar, Robert West, and Jure Leskovec. 2016. Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes. In *Proceedings of the 25th international conference on World Wide Web*. 591–602.
[48] Sergei O Kuznetsov. 2001. On computing the size of a lattice and related decision problems. *Order* 18, 4 (2001), 313–321.
[49] Geon Lee, Seonggoo Kang, and Joyce Jiyoung Whang. 2019. Hyperlink classification via structured graph embedding. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1017–1020.
[50] Fangyuan Lei, Xun Liu, Jianjian Jiang, Liping Liao, Jun Cai, and Huimin Zhao. 2022. Graph convolutional networks with higher-order pooling for semisupervised node classification. *Concurr. Comput. Pract. Exp.* 34, 16 (2022). https://doi.org/10.1002/cpe.5695
[51] Jingdong Li, Zhao Li, Jiaming Huang, Ji Zhang, Xiaoling Wang, Xingjian Lu, and Jingren Zhou. 2021. Large-scale Fake Click Detection for E-commerce

Recommendation Systems. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2595–2606.

[52] Neng Li, Suguo Du, Haizhong Zheng, Minhui Xue, and Haojin Zhu. 2018. Fake reviews tell no tales? dissecting click farming in content-generated social networks. *China Communications* 15, 4 (2018), 98–109.

[53] Rong-Hua Li, Jeffrey Xu Yu, Xin Huang, and Hong Cheng. 2012. Robust Reputation-Based Ranking on Bipartite Rating Networks. In *Proceedings of the Twelfth SIAM International Conference on Data Mining, Anaheim, California, USA, April 26-28, 2012*. SIAM / Omnipress, 612–623. https://doi.org/10.1137/1.9781611972825.53

[54] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*. 689–698.

[55] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.

[56] Boge Liu, Long Yuan, Xuemin Lin, Lu Qin, Wenjie Zhang, and Jingren Zhou. 2019. Efficient (α, β)-core computation: An index-based approach. In *The World Wide Web Conference*. 1130–1141.

[57] Rui Liu, Runze Liu, Andrea Pugliese, and V. S. Subrahmanian. 2020. STARS: Defending against Sockpuppet-Based Targeted Attacks on Reviewing Systems. *ACM Trans. Intell. Syst. Technol.* 11, 5 (2020), 56:1–56:25. https://doi.org/10.1145/3397463

[58] Xiaowen Liu, Jinyan Li, and Lusheng Wang. 2008. Quasi-bicliques: Complexity and binding pairs. In *International Computing and Combinatorics Conference*. Springer, 255–264.

[59] Yajing Liu, Zhengya Sun, and Wensheng Zhang. 2022. Improving Fraud Detection via Hierarchical Attention-based Graph Neural Network. *CoRR* abs/2202.06096 (2022). arXiv:2202.06096 https://arxiv.org/abs/2202.06096

[60] Yadan Luo, Zi Huang, Hongxu Chen, Yang Yang, Hongzhi Yin, and Mahsa Baktashmotlagh. 2021. Interpretable signed link prediction with signed infomax hyperbolic graph. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[61] Bingqing Lyu, Lu Qin, Xuemin Lin, Ying Zhang, Zhengping Qian, and Jingren Zhou. 2020. Maximum biclique search at billion scale. *Proceedings of the VLDB Endowment* (2020).

[62] Arpit Merchant, Ananth Mahadevan, and Michael Mathioudakis. 2022. Scalably Using Node Attributes and Graph Structure for Node Classification. *Entropy* 24, 7 (2022), 906. https://doi.org/10.3390/e24070906

[63] Amanda J. Minnich, Nikan Chavoshi, Abdullah Mueen, Shuang Luan, and Michalis Faloutsos. 2015. TrueView: Harnessing the Power of Multiple Review Sites. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi (Eds.). ACM, 787–797. https://doi.org/10.1145/2736277.2741655

[64] Abhinav Mishra and Arnab Bhattacharya. 2011. Finding the bias and prestige of nodes in networks based on trust scores. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*, Sadagopan Srinivasan, Krithi Ramamritham, Arun Kumar, M. P. Ravindra, Elisa Bertino, and Ravi Kumar (Eds.). ACM, 567–576. https://doi.org/10.1145/1963405.1963485

[65] Mark EJ Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical review E* 69, 2 (2004), 026113.

[66] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[67] B Aditya Prakash, Ashwin Sridharan, Mukund Seshadri, Sridhar Machiraju, and Christos Faloutsos. 2010. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 435–448.

[68] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E* 76, 3 (2007), 036106.

[69] Yuxiang Ren, Hao Zhu, Jiawei Zhang, Peng Dai, and Liefeng Bo. 2021. EnsemFDet: An Ensemble Approach to Fraud Detection based on Bipartite Graph. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 2039–2044. https://doi.org/10.1109/ICDE51399.2021.00197

[70] Chonggang Song, Qian Lin, Guohui Ling, Zongyi Zhang, Hongzhao Chen, Jun Liao, and Chuan Chen. 2020. LoCEC: local community-based edge classification in large online social networks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1689–1700.

[71] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[72] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. (2018). https://openreview.net/forum?id=rJXMpikCZ

[73] Guan Wang, Sihong Xie, Bing Liu, and Philip S. Yu. 2011. Review Graph Based Online Store Review Spammer Detection. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, Diane J. Cook, Jian Pei, Wei Wang, Osmar R. Zaïane, and Xindong Wu (Eds.). IEEE Computer Society, 1242–1247. https://doi.org/10.1109/ICDM.2011.124

[74] Guan Wang, Sihong Xie, Bing Liu, and Philip S. Yu. 2012. Identify Online Store Review Spammers via Social Review Graph. *ACM Trans. Intell. Syst. Technol.* 3, 4 (2012), 61:1–61:21. https://doi.org/10.1145/2337542.2337546

[75] Jianyu Wang, Rui Wen, Chunming Wu, Yu Huang, and Jian Xiong. 2019. FdGars: Fraudster Detection via Graph Convolutional Networks in Online App Review System. In *Companion of The 2019 World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, Sihem Amer-Yahia, Mohammad Mahdian, Ashish Goel, Geert-Jan Houben, Kristina Lerman, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (Eds.). ACM, 310–316. https://doi.org/10.1145/3308560.3316586

[76] Kai Wang, Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang. 2020. Efficient bitruss decomposition for large-scale bipartite graphs. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 661–672.

[77] Lusheng Wang. 2013. Near optimal solutions for maximum quasi-bicliques. *Journal of Combinatorial Optimization* 25, 3 (2013), 481–497.

[78] Guangyu Wu, Derek Greene, and Padraig Cunningham. 2010. Merging multiple criteria to identify suspicious reviews. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, Xavier Amatriain, Marc Torrens, Paul Resnick, and Markus Zanker (Eds.). ACM, 241–244. https://doi.org/10.1145/1864708.1864757

[79] Ke Wu, Song Yang, and Kenny Q Zhu. 2015. False rumors detection on sina weibo by propagation structures. In *2015 IEEE 31st international conference on data engineering*. IEEE, 651–662.

[80] Sihong Xie, Guan Wang, Shuyang Lin, and Philip S. Yu. 2012. Review spam detection via temporal pattern discovery. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, Qiang Yang, Deepak Agarwal, and Jian Pei (Eds.). ACM, 823–831. https://doi.org/10.1145/2339530.2339662

[81] Chang Xu and Jie Zhang. 2015. Towards Collusive Fraud Detection in Online Reviews. In *2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015*, Charu C. Aggarwal, Zhi-Hua Zhou, Alexander Tuzhilin, Hui Xiong, and Xindong Wu (Eds.). IEEE Computer Society, 1051–1056. https://doi.org/10.1109/ICDM.2015.62

[82] Chang Xu and Jie Zhang. 2017. Collusive Opinion Fraud Detection in Online Reviews: A Probabilistic Modeling Approach. *ACM Trans. Web* 11, 4 (2017), 25:1–25:28. https://doi.org/10.1145/3098859

[83] Yixing Yang, Yixiang Fang, Maria E Orlowska, Wenjie Zhang, and Xuemin Lin. 2021. Efficient bi-triangle counting for large bipartite networks. *Proceedings of the VLDB Endowment* 14, 6 (2021), 984–996.

[84] Kaiqiang Yu, Cheng Long, Shengxin Liu, and Da Yan. 2022. Efficient Algorithms for Maximal k-Biplex Enumeration. In *Proceedings of the 2022 International Conference on Management of Data*. 860–873.

[85] Mengxi Yu, Ziyu Liu, Yuhang Tang, and Jianfeng Jiang. 2021. Recognition algorithm of e-commerce click farming based on K-means technology. In *2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)*. IEEE, 103–106.

[86] Baoliang Zhang, Xiaoxin Guo, Zhenchuan Tu, and Jia Zhang. 2022. Graph alternate learning for robust graph neural networks in node classification. *Neural Comput. Appl.* 34, 11 (2022), 8723–8735. https://doi.org/10.1007/s00521-021-06863-1

[87] Shijie Zhang, Hongzhi Yin, Tong Chen, Quoc Viet Hung Nguyen, Zi Huang, and Lizhen Cui. 2020. GCN-Based User Representation Learning for Unifying Robust Recommendation and Fraudster Detection. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 689–698. https://doi.org/10.1145/3397271.3401165

[88] Xianhang Zhang, Hanchen Wang, Jianke Yu, Chen Chen, Xiaoyang Wang, and Wenjie Zhang. 2022. Bipartite graph capsule network. *World Wide Web* (2022), 1–20.

[89] Xianhang Zhang, Hanchen Wang, Jianke Yu, Chen Chen, Xiaoyang Wang, and Wenjie Zhang. 2022. Polarity-based graph neural network for sign prediction in signed bipartite graphs. *World Wide Web* 25, 2 (2022), 471–487. https://doi.org/10.1007/s11280-022-01015-4

[90] Yun Zhang, Charles A Phillips, Gary L Rogers, Erich J Baker, Elissa J Chesler, and Michael A Langston. 2014. On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types. *BMC bioinformatics* 15, 1 (2014), 1–18.

[91] Jie Zhao, Raymond YK Lau, Wenping Zhang, Kaihang Zhang, Xu Chen, and Deyu Tang. 2016. Extracting and reasoning about implicit behavioral evidences for detecting fraudulent online transactions in e-Commerce. *Decision support systems* 86 (2016), 109–121.

A EXPERIMENTAL SETUP AND ENVIRONMENT

A.1 Parameter Settings

Parameter Settings of GFDN. In our model, by default, we set the number of GNN layers $L_g = 4$, the number of encoder layers and decoder layers $L_e = L_d = 3$, learning rate $lr = 0.001$, hidden dimension as 128. For "Ride Item's Coattails" attack detection, we set the number of communities $\mathcal{K} = 32$, number of epochs as 100, loss coefficients $\lambda_{lt} = \lambda_{lf} = \lambda_{et} = \lambda_{ef} = 1$, $\gamma_l = \gamma_e = 2$, $\omega_{ae} = 0.2$, $\omega_c = 0.1$, $\omega_l = 0.1$ and $\omega_e = 0.6$. We set $\alpha_\tau^- = \alpha_\tau^+ = 2$, $\beta_\tau^- = 1$ and $\beta_\tau^+ = 11$ for both TB and TC. The fraud detection threshold $\tau_\mathcal{E}$ is set to 0.5 in TB and 0.45 in TC. For the STARS attack detection, we set the number of epochs as 75, and the fraudster detection threshold $\tau_\mathcal{U}$ is set to 0.4. Due to the distribution of datasets, we set $\alpha_\tau^- = 2$, $\alpha_\tau^+ = 15$, $\beta_\tau^- = \beta_\tau^+ = 1$ for Alpha and OTC datasets. For loss coefficients, we change ω_l to 0.6, ω_e to 0.1. The remaining parameters are consistent with those for the "Ride Item's Coattails" attack detection. Adam [40] is chosen as the optimizer. Due to the huge size of TB and TC datasets, we adopt the minibatch sampling method [27] to incorporate them in GPU during training.

Parameter Settings of RTV. RTV has eight pre-set weights. We use the settings of these parameters as indicated in the original work [57]. We choose logistic regression as the supervised learning part of RTV. In [57], the tuning strategy of RTV-SUP's supervised learning part is not provided. Therefore, we use the strategy in favor of its experimental results, to obtain the best experimental results of RTV-SUP.

A.2 Implementation Details

During the feature generation phase, (α, β)-core is computed by QueryOPT [3, 56] implemented with C++. The neural network of GFDN is implemented by Pytorch framework and PyTorch Geometric [22] package in Python. The swig [8] compiler is used to integrate QueryOPT into Python environment. The code of GFDN is publicly available on Github [7].

A.3 Experiment Environment

The experiments are conducted on a server that is running Ubuntu 18.04.6 LTS system with Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz, NVIDIA Tesla V100 16G GPU and 30GB RAM.

A.4 Evaluation Metrics

Five popular metrics are used for the evaluation of group-based fraud detection: F1-score, accuracy, AUC, precision and recall. TP, TN, FP and FN are used to denote the true positive, true negative, false positive and false negative, respectively. These five metrics are defined as follows: AUC is the area under the receiver operating characteristic (ROC) curve. Accuracy is defined as $\text{Acc} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$. F1-score is computed by the precision and recall values, which is defined as $\text{F1} = \frac{2\times\text{Pre}\times\text{Recall}}{\text{Pre+Recall}} = \frac{2\times\text{TP}}{2\times\text{TP+FP+FN}}$. Precision is defined as $\text{Pre} = \frac{\text{TP}}{\text{TP+FP}}$. Recall is defined as $\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$. For these five metrics, higher values indicate better performance.

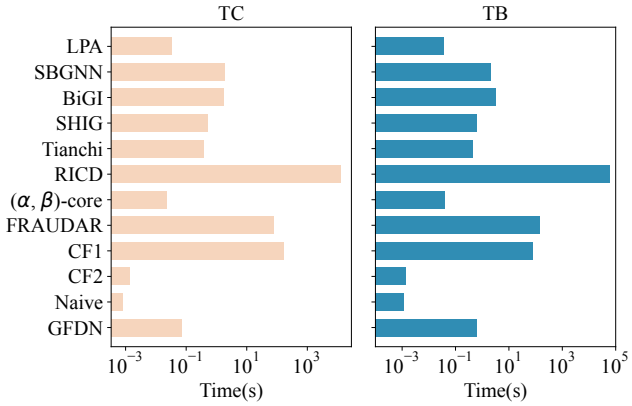


Figure 4: Query Time Evaluation of "Ride Item's Coattails"

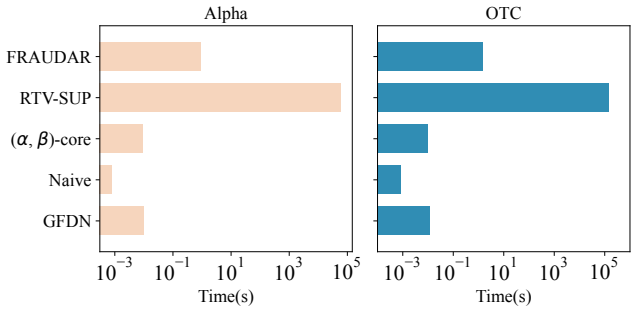


Figure 5: Query Time Evaluation of STARS

A.5 Label Acquisition for "Ride Item's Coattails" Attack

Datasets TC and TB contain the labels of "Ride Item's Coattails" attack. These labels are obtained in the following ways:

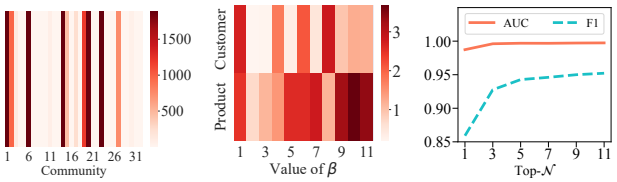
- **Expert Labeling.** Business experts in the Taobao platform provide significantly accurate label information, along with enormous labor costs.
- **Feature Filtering in Dense Subgraphs.** We clean up the vertices with low data concentration. Then we utilize the temporal and attribute information to support the business experts in the labeling task. In this case, we add attack labels to these clicks even if business experts are not completely convinced that the customer is attacking.
- **Red Team Attack Simulation.** We hire buyers and sellers on the Taobao platform to comply with our rules and conduct the attacks to obtain real and accurate attack data.

B APPROACH ANALYSES

B.1 Query Time Comparison

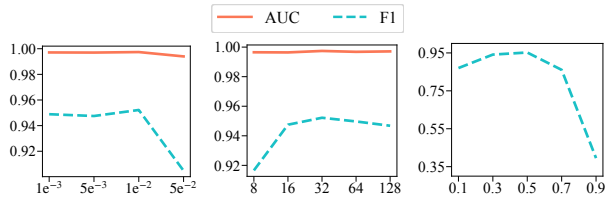
In this section, we report the query time costs, *i.e.,* time for distinguishing the frauds in the test set, of the compared methods for "Ride Item's Coattails" attack detection and STARS attack detection.

The efficiency results for "Ride Item's Coattails" attack detection are shown in Figure 4. Compared with learning-based methods,



(a) Heatmap of $C_{\mathcal{U}}$ (b) Heatmap of $W_{(\mathcal{U},s)}$, $W_{(\mathcal{V},s)}$ (c) Varying number of β

Figure 6: In-Depth Effectiveness Analysis of GFDN



(a) Varying lr (b) Varying $\mathcal{K}$ (c) Varying $\tau_{\mathcal{E}}$

Figure 8: Parameter Analysis Results in GFDN

LPA is the most efficient method, but it shows the lowest accuracy among these methods. Meanwhile, GFDN is around an order of magnitude faster than other learning-based methods on TC. Compared with fraud detection methods, despite the best efficiency, CF2 is significantly outperformed by GFDN, while the other methods are much less efficient and accurate than GFDN. RICD, as the SOTA method, requires around five orders of magnitude more query time compared to GFDN. On TC and TB datasets, the efficiency of (α, β)-core is close to or slightly faster than GFDN, but GFDN achieves 11.03% and 17.83% improvement than (α, β)-core *w.r.t.* F1-score metric. Consequently, GFDN can achieve remarkably high accuracy with competitive efficiency.

Efficiency test results of the STARS attack detection are shown in Figure 5. These results show that GFDN is remarkably efficient. In more detail, Naive is the most efficient algorithm among all the models, but not much different from GFDN; (α, β)-core ranks second in efficiency, but GFDN is almost as fast as it. With efficiency close to theirs, the effectiveness of GFDN is much higher than theirs. Compared to RTV-SUP, GFDN is 6 to 7 orders of magnitude faster on both datasets. These experimental results prove that GFDN outperforms the state-of-the-art model RTV in completing the STARS attack detection, both in terms of effectiveness and efficiency.

In conclusion, GFDN offers remarkable efficiency in the task of group-based fraud detection while achieving high effectiveness. With the help of GFDN, group-based fraud on attributed bipartite graphs can be detected efficiently and effectively.

B.2 In-Depth Effectiveness Analysis

In this section, we provide an in-depth analysis of the effectiveness of the structural information in the initial features and the clustering module on TB dataset. Specifically, we analyze the result of community affiliation prediction $C_{\mathcal{U}}$, two trained parameters in our model: $W_{(\mathcal{U},s)}$ and $W_{(\mathcal{V},s)}$. To analyze the improvement brought by structural information, we also test the influence caused by varying numbers and values of β.

The heatmap in Figure 6(a) shows the summation of probabilities for all fraudsters to be allocated to each cluster, *i.e.,* $\sum_{u \text{ is a fraudster}} \boldsymbol{c}_u$. Specifically, each bar represents a community. The darker red bars indicate that there is a higher probability for the fraudsters to belong to the corresponding communities. We can see that the fraudsters are almost allocated in a few communities by our model rather than evenly scattered among them. It is evident that our model effectively mines the fraudster communities.

Figure 6(b) shows the values of learned weights of structural features for customers and products, *i.e.,* $W_{(\mathcal{U},s)}$ and $W_{(\mathcal{V},s)}$, respectively. In this figure, the horizontal coordinates represent the values of β in the (2, β)-core. The darker red represents the greater absolute value of the corresponding weight. We can see that the model gives high importance to the features obtained by $\beta = 1$ for both customer and product vertices, which can be verified by the truth that all fraudsters should be in (2, 1)-core. Varying the value of β, we can find that the weights are greater on the feature entries generated by larger β for products. It can be concluded that the model pays more attention to popular products.

To further demonstrate the effectiveness of structural information , we generate the structural features with the β values that with top-$\mathcal{N}$ greatest weight values reported in the previous paragraph,*i.e.,* weights shown in the Figure 6(b). The experimental results are presented in Figure 6(c), where the x-axis shows values of $\mathcal{N}$. The model's performance is significantly improved when top-3 structural features are used compared to the case where only one feature is used. The performance continues to improve with the growth of $\mathcal{N}$, which becomes stable when $\mathcal{N} > 5$.

B.3 Parameter Sensitivity Analysis

We report the results for parameter sensitivity analysis on TB dataset in this section, which are shown in Figure 8.

Figure 8(a) shows the influence of varying learning rate (lr) on the model. The performance of GFDN notably drops when $lr = 5 \times 10^{-2}$. We found that the model is hard to converge when $lr > 5 \times 10^{-2}$ and $lr < 1 \times 10^{-3}$. The salient point is reached when $lr = 1 \times 10^{-2}$, which is selected as our default setting.

Figure 8(b) shows the influence of the varying number of clusters $\mathcal{K}$ on the experimental results. The experimental results show the stability of F1-score of our model when $\mathcal{K} \geq 8$. This result indicates that our model is not sensitive to the selection of the number of clusters $\mathcal{K}$. Meanwhile, the small value of $\mathcal{K}$ would lead to deteriorated performance due to the consequent low dimension of $C_{\mathcal{U}}$. In our experiments, we choose $\mathcal{K} = 32$ to save computational resources while ensuring fraud detection accuracy.

Figure 8(c) shows the effect of the fraud detection threshold $\tau_{\mathcal{E}}$ on the model. Particularly, an edge is predicted to be fraudulent if its predicted probability is greater than the threshold $\tau_{\mathcal{E}}$. We only report F1-score since AUC metric is not influenced by varying $\tau_{\mathcal{E}}$. Because of focal loss, GFDN maintains good performance for threshold values between 0.3 and 0.6. Once $\tau_{\mathcal{E}} \geq 0.7$, the model's accuracy is greatly decreased due to the label imbalance in the dataset. We choose $\tau_{\mathcal{E}} = 0.5$ for TB in our experiments to ensure that our model can achieve reasonable accuracy.