Unveiling the Threat of Fraud Gangs to Graph Neural Networks: Multi-Target Graph Injection Attacks Against GNN-Based Fraud Detectors

Jinhyeok Choi, Heehyeon Kim, Joyce Jiyoung Whang*

School of Computing, KAIST
{cjh0507, heehyeon, jjwhang}@kaist.ac.kr

Abstract

Graph neural networks (GNNs) have emerged as an effective tool for fraud detection, identifying fraudulent users, and uncovering malicious behaviors. However, attacks against GNN-based fraud detectors and their risks have rarely been studied, thereby leaving potential threats unaddressed. Recent findings suggest that frauds are increasingly organized as gangs or groups. In this work, we design attack scenarios where fraud gangs aim to make their fraud nodes misclassified as benign by camouflaging their illicit activities in collusion. Based on these scenarios, we study adversarial attacks against GNN-based fraud detectors by simulating attacks of fraud gangs in three real-world fraud cases: spam reviews, fake news, and medical insurance frauds. We define these attacks as multi-target graph injection attacks and propose MonTi, a transformer-based Multi-target one-Time graph injection attack model. MonTi simultaneously generates attributes and edges of all attack nodes with a transformer encoder, capturing interdependencies between attributes and edges more effectively than most existing graph injection attack methods that generate these elements sequentially. Additionally, MonTi adaptively allocates the degree budget for each attack node to explore diverse injection structures involving target, candidate, and attack nodes, unlike existing methods that fix the degree budget across all attack nodes. Experiments show that MonTi outperforms the state-of-the-art graph injection attack methods on five real-world graphs.

Introduction

Recent endeavors have highlighted the power of Graph Neural Networks (GNNs) for fraud detection (Dou et al. 2021; Xu et al. 2022). In fraud detection tasks, complex interactions of fraudsters can be effectively modeled using graphs, and frauds are typically represented as nodes corresponding to individuals with malicious intentions. GNN-based fraud detection methods aim to determine whether the nodes are fraudulent or benign. Meanwhile, it has been reported that adversarial attacks can cause GNNs to malfunction across various domains, including recommender systems and social network analysis (Chen et al. 2022a; You et al. 2023; Huang and Li 2023; Shao et al. 2023; Wang et al. 2023a). Those works consider vanilla GNNs as victim models,

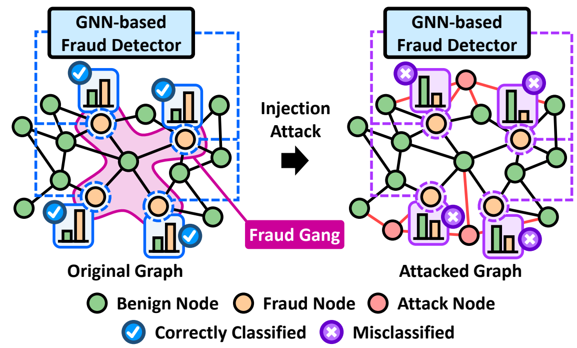

Figure 1: An example of a multi-target graph injection attack against a GNN-based fraud detector: a fraud gang injects attack nodes to induce misclassification of their fraud nodes.

e.g., GCN (Kipf and Welling 2017), GraphSAGE (Hamilton, Ying, and Leskovec 2017), and GAT (Veličković et al. 2018). On the other hand, various tailored GNNs for fraud detection have recently been developed to filter the camouflaged fraudsters, such as CARE-GNN (Dou et al. 2020), PC-GNN (Liu et al. 2021), and GAGA (Wang et al. 2023c). However, vulnerabilities of these GNN-based fraud detectors to adversarial attacks remain unexplored.

It has been recently observed that frauds are increasingly organized into gangs or groups exhibiting collusive patterns to carry out fraudulent activities more effectively with reduced risk (Wang et al. 2023b; Yu et al. 2023; Ma et al. 2023). For example, in the medical insurance domain, fraudsters may collaborate with doctors or insurance agents to obtain fake diagnoses. On online review platforms, fraudsters could create multiple fake reviews using different IDs. On social media platforms, fraudsters can spread misinformation by using multiple fake accounts. We design these three attack scenarios where fraud gangs attack GNN-based fraud detectors to make them misclassify the fraud nodes as benign. Furthermore, to simulate the scenarios, we create datasets and target sets that consist of fraud nodes grouped based on metadata or relations in real-world graphs.

In this work, the adversarial attack on GNN-based fraud detectors by fraud gangs is defined as a multi-target graph injection attack. We adopt a graph injection attack, as it is more feasible than a graph modification attack, which re-

*Corresponding author.

quires privileged access to alter existing structures (Chen et al. 2022b). As illustrated in Figure 1, in our attack scenarios, a fraud gang attempts to deceive the detector by injecting malicious nodes (i.e., attack nodes) into the original graph, causing their fraud nodes to be misclassified by increasing the scores of being benign. We consider a black-box evasion attack, where the attacker can access only the original graph, partial labels, and a surrogate model, and the attack occurs during the victim model's inference phase (Sun et al. 2023).

Existing methods (Tao et al. 2021; Zou et al. 2021; Wang et al. 2022) have focused on adversarial attacks for randomly grouped target nodes. However, fraud nodes are often organized into gangs to camouflage their illicit activities. While the existing methods have investigated attacks where vanilla GNNs are employed as victim models, GNN-based fraud detectors are tailored to address heterophily induced by fraud nodes, making them more difficult to attack than vanilla GNNs. To overcome the limitations of these existing methods in our scenario, we propose MonTi, a transformer-based Multi-target one-Time graph injection attack model.

MonTi significantly differs from the existing graph injection attack methods in several aspects, as summarized in Table 1. The existing methods, such as TDGIA (Zou et al. 2021), Cluster Attack (Wang et al. 2022), and G^2A2C (Ju et al. 2023), inject multiple attack nodes sequentially, fixing the graph structure at each step. Those approaches fix the degree budget across all attack nodes due to the lack of information about future steps, limiting their flexibility and efficiency. In contrast, MonTi injects all attack nodes at once, adaptively assigning the degree budget for each attack node to explore diverse structures among target, candidate, and attack nodes. Furthermore, the existing methods, including G-NIA (Tao et al. 2021), overlook interactions within target nodes and among attack nodes, which are crucial for modeling the intricate relationships within a fraud gang and attack nodes. Additionally, the existing methods sequentially generate attributes and edges of attack nodes, only considering a one-way dependency, which fails to model collusive patterns of fraud gangs. On the other hand, MonTi employs a transformer encoder to effectively capture interdependencies within target and attack nodes, and between attributes and edges. Our contributions are summarized as follows:

- To the best of our knowledge, our work is the first study to investigate the vulnerabilities of GNN-based fraud detectors and also the first study on graph injection attacks for multiple target nodes organized by groups.
- We propose a graph injection attack method MonTi, which generates attributes and edges of all attack nodes at once via the adversarial structure encoding transformer.
- MonTi can explore adversarial injection structures comprehensively by generating adversarial edges with adaptive degree budget allocation for each attack node.
- Experimental results demonstrate that MonTi substantially outperforms the state-of-the-art graph injection attack methods on five real-world graphs.[1]

[1]Our datasets and codes are available at https://github.com/bdi-lab/MonTi ; the full paper is available at https://bdi-lab.kaist.ac.kr.

	Victim		Multi-Tar.		Multi-Inj.		Budget		Attr-Edge
	Van	Frd	Rnd	Frd	Seq	One	Fix	Ada	Interdep.
G-NIA	✔		✔				✔		
TDGIA	✔		✔		✔		✔		
Cluster Atk.	✔		✔		✔		✔		
G^2A2C	✔				✔		✔		
MonTi	✔	✔	✔	✔		✔		✔	✔

Table 1: Comparison of graph injection attack methods. We mark whether victim models are vanilla GNNs (Van) or GNN-based fraud detectors (Frd); target nodes are randomly grouped (Rnd) or fraud gangs (Frd); multiple attack nodes are injected sequentially (Seq) or at once (One); degree budget per attack node is fixed (Fix) or adaptively assigned (Ada); attribute-edge interdependency is considered or not.

Related Work

Graph-based Fraud Detection The majority of graph-based fraud detection methods adopt GNNs and define this task as a problem of classifying nodes into two categories: fraud or benign (Huang et al. 2022; Shi et al. 2022; Chai et al. 2022; Li et al. 2022b,a). In fraud graphs, fraud nodes are much fewer than benign nodes, resulting in a class imbalance problem. In addition, fraud nodes tend to connect with or imitate the attributes of benign nodes to hide their malicious activities. Such behavior introduces heterophily (Zhu et al. 2020), which causes vanilla GNNs to fail in detecting fraud nodes. To address the class imbalance and heterophily in fraud graphs, recent approaches have introduced learnable neighborhood samplers (Dou et al. 2020; Liu et al. 2021), utilized Beta wavelet transform (Tang et al. 2022; Wu et al. 2023) and proposed group aggregation strategies (Wang et al. 2023c; Duan et al. 2024). Despite the recent advancements, vulnerabilities of GNN-based fraud detectors to adversarial attacks have not yet been studied.

Adversarial Attacks on Graphs It has been shown that even a few adversarial modifications on graphs can significantly degrade the performance of GNNs (Zügner, Akbarnejad, and Günnemann 2018; Bojchevski and Günnemann 2019; Zügner and Günnemann 2019; Ma et al. 2021; Ma, Deng, and Mei 2022; Lin, Blaser, and Wang 2022). Graph injection attacks have emerged focusing on more practical settings, where attackers inject new malicious nodes instead of modifying existing nodes and edges (Wang et al. 2020; Sun et al. 2020). Recent developments include learning attribute and edge generators to target unseen nodes (Tao et al. 2021), introducing defective edge selection strategies (Zou et al. 2021), regarding graph injection attacks as clustering tasks (Wang et al. 2022), and employing the advantage actor-critic framework (Ju et al. 2023). More detailed discussions on related work are provided in Appendix A.

Problem Definition

An undirected attributed graph is defined as $G = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ where $\mathcal{V}$ is a set of n nodes, $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is a set of edges, and $\mathcal{X}$ is a set of D-dimensional node attribute vectors. For a node $v \in \mathcal{V}$, we represent its attribute vector as $\mathbf{x}_v \in \mathcal{X}$ and

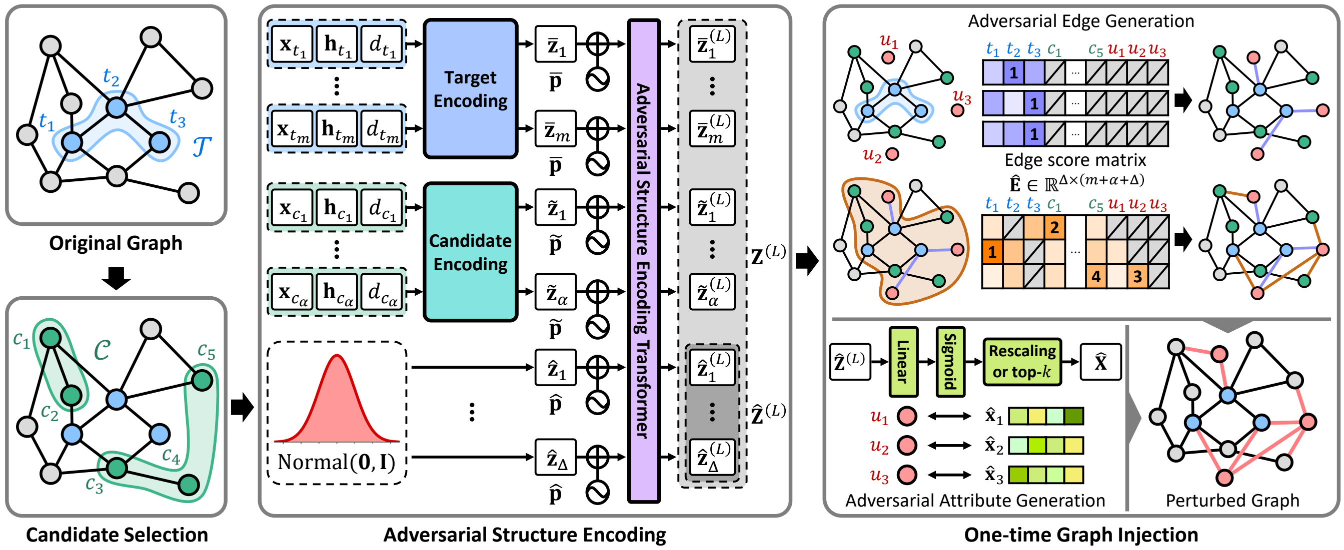

Figure 2: Overview of MonTi. Given the original graph and the target set, MonTi selects candidate nodes among K-hop neighbors of target nodes utilizing a learnable scoring function. Then, MonTi calculates the intermediate node representations of target, candidate, and attack nodes by the adversarial structure encoding transformer. Based on those intermediate representations, adversarial attributes and edges for injection are simultaneously generated.

its label as $y_v \in \{0, 1\}$ where $y_v = 1$ indicates v is a fraud and $y_v = 0$ indicates v is benign. $\mathcal{Y} = \{y_v \mid v \in \mathcal{V}\}$ denotes a set of node labels. We represent a GNN-based fraud detector as $f_\theta(\cdot)$ where θ indicates learnable parameters. The fraud score vector $\mathbf{s}_v \in \mathbb{R}^2$ and the predicted label $\hat{y}_v \in \{0, 1\}$ are calculated by $\mathbf{s}_v = f_\theta(G, v) = \text{MLP}\,(\Phi(G, v))$ and $\hat{y}_v = \arg\max_i\ s_{v,i}$ where MLP is a multi-layer perceptron, $\Phi(\cdot)$ denotes a GNN encoder and $s_{v,i}$ denotes the i-th element of $\mathbf{s}_v$. We formulate the objective function of the fraud detector as $\max_\theta \sum_{v\in\mathcal{V}} \mathbb{I}\,(\hat{y}_v = y_v)$ where $\mathbb{I}\,(\cdot)$ is an indicator function. Most GNN-based fraud detectors utilize label information based on domain-specific knowledge, employing (semi-)supervised approaches (Dou et al. 2020; Liu et al. 2022). As the first study of adversarial attacks against GNN-based fraud detectors, we assume that graphs are single-relational, in line with existing studies of adversarial attacks on graphs (Tao et al. 2021; Zou et al. 2021; Wang et al. 2022; Ju et al. 2023).

A graph injection attack injects attack nodes $\mathcal{V}_{\text{in}}$ with attributes $\mathcal{X}_{\text{in}}$ and adversarial edges $\mathcal{E}_{\text{in}} \subset (\mathcal{V}' \times \mathcal{V}') \setminus (\mathcal{V} \times \mathcal{V})$ into the original graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ where $\mathcal{V}' = \mathcal{V} \cup \mathcal{V}_{\text{in}}$. We define the perturbed graph as $G' = (\mathcal{V}', \mathcal{E}', \mathcal{X}')$ where $\mathcal{E}' = \mathcal{E} \cup \mathcal{E}_{\text{in}}$ and $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}_{\text{in}}$. The multi-target graph injection attack against a GNN-based fraud detector aims to make fraud nodes in the target set $\mathcal{T} \subset \mathcal{V}$ misclassified. For a node $v \in \mathcal{V}$, let $\mathbf{s}'_v = f_{\theta^*}(G', v)$ where $\theta^* = \arg\min_\theta\ \mathcal{L}_{\text{train}}(f_\theta, G, \mathcal{D})$, $\mathcal{L}_{\text{train}}$ is a training loss of $f_\theta(\cdot)$, and $\mathcal{D} \subset \mathcal{Y}$ is a node label set for training. The objective function of the multi-target graph injection attack is:

$$\min_{G'} \sum_{t\in\mathcal{T}} \mathbb{I}\left(\arg\max_i s'_{t,i} = y_t\right) \text{ st. } |\mathcal{V}_{\text{in}}| \le \Delta, |\mathcal{E}_{\text{in}}| \le \eta \quad (1)$$

where Δ and η denote node and edge budgets, respectively.

Multi-target One-time Graph Injection Attack

We propose MonTi (Figure 2), a graph injection attack model with a transformer-based architecture. MonTi utilizes self-attention to capture interactions among target, candidate, and attack nodes, simultaneously generating attributes and edges of attack nodes to reflect their interdependencies. In addition, by generating edges of all attack nodes at once, MonTi enables flexible budget allocation for each node.

Adversarial Structure Encoding

Three types of contexts can affect multi-target graph injection attacks: target nodes $\mathcal{T} = \{t_1, \cdots, t_m\}$, attack nodes $\mathcal{V}_{\text{in}} = \{u_1, \cdots, u_\Delta\}$, and candidate nodes. We define $\mathcal{N}^{(K)} \subset \mathcal{V}$ as a set of K-hop neighbors of the target nodes, excluding the target nodes themselves. From $\mathcal{N}^{(K)}$, we select α nodes to form a set of candidate nodes $\mathcal{C} = \{c_1, \cdots, c_\alpha\} \subset \mathcal{N}^{(K)}$. The neighborhood of target nodes provides information on how to propagate malicious messages within a GNN-based architecture. MonTi employs multi-head self-attention to learn intermediate representations for nodes in $\mathcal{T}$, $\mathcal{V}_{\text{in}}$, and $\mathcal{C}$ to capture interactions among these contexts, distinguishing their respective roles.

Candidate Selection The size of $\mathcal{N}^{(K)}$ can drastically increase depending on the target nodes, making it computationally inefficient and challenging to find the optimal G'. Therefore, when the number of possible candidate nodes exceeds the threshold n_c, MonTi selects candidate nodes to narrow the search space with a learnable scoring function $\mathcal{J}$; otherwise, all nodes are considered as candidate nodes. To incorporate the class distribution learned by GNN architectures, MonTi utilizes a surrogate GNN model pretrained on the original graph. MonTi calculates candidate scores for all $v \in \mathcal{N}^{(K)}$ by integrating node attributes and graph topology with those of the target nodes using the surrogate GNN:

$$\mathcal{J}(G, v) = \text{MLP}\left(\sigma\left([\mathbf{q}_v \,||\, \mathbf{m}_v \,||\, \mathbf{h}_v \,||\, \mathbf{h}_{\mathcal{T}}]\right)\right) \quad (2)$$

where $\mathbf{q}_v = \text{MLP}(\mathbf{x}_v) \in \mathbb{R}^{D_H}$, D_H is the dimension of node representation, $\mathbf{m}_v = \text{MLP}([d_v || \beta_v]) \in \mathbb{R}^{D_H}$, d_v is the degree of node v, β_v is the number of target nodes directly connected to v, $\mathbf{h}_v = \Phi^*(G, v) \in \mathbb{R}^{D_H}$, Φ^* is a pretrained GNN encoder of the surrogate model, $\mathbf{h}_{\mathcal{T}} = \text{READOUT}(\{\mathbf{h}_t \,|\, t \in \mathcal{T}\}) \in \mathbb{R}^{D_H}$, READOUT$(\cdot)$ is a pooling function, $||$ is a horizontal concatenation, and σ is the activation function. We adopt the mean operation as a pooling function. Based on the calculated scores, the top-n_c nodes are selected as candidate nodes. To solve the optimization problems of discrete selection in MonTi, we adopt the Gumbel-Top-k technique (Jang, Gu, and Poole 2017; Kool, Hoof, and Welling 2019) coupled with the straight-through estimator (Bengio, Léonard, and Courville 2013).

Target and Candidate Encoding To exploit the structural information of target and candidate nodes, we introduce learnable encodings with three different factors of each node v related to the attack: raw attributes $\mathbf{x}_v$, degree d_v, and the GNN representation $\mathbf{h}_v$. We employ the raw attributes to explicitly take into account the node itself. The node degree is utilized to indirectly measure the susceptibility of the node to the change of its neighbors. We use the GNN representation to capture an inherent class distribution learned by the surrogate GNN and topological information of the node. With a learnable target encoding function $\overline{\mathcal{P}}$, a target encoding $\overline{\mathbf{z}}_i = \overline{\mathcal{P}}(\mathbf{x}_{t_i}, d_{t_i}, \mathbf{h}_{t_i})$ for a target node $t_i \in \mathcal{T}$ is computed by projecting and integrating all three factors:

$$\overline{\mathbf{z}}_i = \text{MLP}\left(\sigma([(\overline{\mathbf{W}}\mathbf{x}_{t_i} + \overline{\mathbf{b}}_1) \,||\, (\overline{\mathbf{w}} d_{t_i} + \overline{\mathbf{b}}_2) \,||\, \mathbf{h}_{t_i}])\right) \quad (3)$$

where MLP transforms the input into $\mathbb{R}^{D_H}$, $\overline{\mathbf{W}} \in \mathbb{R}^{D_H \times D}$, $\overline{\mathbf{w}} \in \mathbb{R}^{D_H}$, $\overline{\mathbf{b}}_1 \in \mathbb{R}^{D_H}$, $\overline{\mathbf{b}}_2 \in \mathbb{R}^{D_H}$. Similarly, we compute a candidate encoding $\widetilde{\mathbf{z}}_i = \widetilde{\mathcal{P}}(\mathbf{x}_{c_i}, d_{c_i}, \mathbf{h}_{c_i})$ for a candidate node $c_i \in \mathcal{C}$ where $\widetilde{\mathcal{P}}$ denotes a learnable candidate encoding function. Note that the parameters of $\overline{\mathcal{P}}$ and $\widetilde{\mathcal{P}}$ are distinct.

Adversarial Structure Encoding Transformer We propose to adopt the transformer (Vaswani et al. 2017) encoder to learn the intermediate node representations of the target, candidate, and attack nodes for adversarial attribute and edge generation. Each attack node $u_i \in \mathcal{V}_{\text{in}}$ is initialized with the representation $\widehat{\mathbf{z}}_i \in \mathbb{R}^{D_H}$ sampled from a standard Gaussian distribution. The input sequence for the adversarial structure encoding transformer is defined as $\mathbf{Z}^{(0)} = [\overline{\mathbf{Z}}^{(0)} || \widetilde{\mathbf{Z}}^{(0)} || \widehat{\mathbf{Z}}^{(0)}] = [\overline{\mathbf{z}}_1 \cdots \overline{\mathbf{z}}_m || \widetilde{\mathbf{z}}_1 \cdots \widetilde{\mathbf{z}}_\alpha || \widehat{\mathbf{z}}_1 \cdots \widehat{\mathbf{z}}_\Delta]$ where $\overline{\mathbf{Z}}^{(0)}$, $\widetilde{\mathbf{Z}}^{(0)}$, and $\widehat{\mathbf{Z}}^{(0)}$ denote the input sequence corresponding to the target, candidate, and attack nodes, respectively. We then add learnable positional encodings $\overline{\mathbf{p}}, \widetilde{\mathbf{p}}, \widehat{\mathbf{p}} \in \mathbb{R}^{D_H}$ to the corresponding node representations. Subsequently, $\mathbf{Z}^{(0)}$ is processed through an L-layer transformer encoder with n_h heads to obtain the final representations $\mathbf{Z}^{(L)} = [\overline{\mathbf{Z}}^{(L)} || \widetilde{\mathbf{Z}}^{(L)} || \widehat{\mathbf{Z}}^{(L)}]$. To mitigate noise from attack nodes whose edges are not formed yet, we optionally mask the attack nodes when calculating target and candidate node representations, treating this masking as a hyperparameter.

Adversarial Attribute Generation

To generate the attributes of attack nodes, we project $\widehat{\mathbf{Z}}^{(L)}$ into D-dimensional space: $\mathbf{F} = \text{sigmoid}(\mathbf{W}_a \widehat{\mathbf{Z}}^{(L)} + \mathbf{b}_a) \in \mathbb{R}^{D_H \times \Delta}$ where $\mathbf{W}_a \in \mathbb{R}^{D \times D_H}$ and $\mathbf{b}_a \in \mathbb{R}^D$. Depending on whether the raw attributes are continuous or discrete, $\mathbf{F}$ is transformed into the malicious attributes in different ways.

For continuous attributes, we rescale $\mathbf{F}$ by using the min-max vectors of raw attributes $\mathbf{x}_{\text{min}}, \mathbf{x}_{\text{max}} \in \mathbb{R}^D$ derived from the original graph. The adversarial attribute vector $\widehat{\mathbf{x}}_i$ for the attack node $u_i \in \mathcal{V}_{\text{in}}$ is computed by $\widehat{\mathbf{x}}_i = \mathbf{F}_i \odot (\mathbf{x}_{\text{max}} - \mathbf{x}_{\text{min}}) + \mathbf{x}_{\text{min}}$ where $\mathbf{F}_i \in \mathbb{R}^D$ denotes the i-th column vector of $\mathbf{F}$, and $\odot$ denotes the element-wise product operator.

We adopt the Gumbel-Top-k technique to optimize discrete choices, such as candidate selection, discrete attribute generation, and edge generation. Following (Tao et al. 2021; Nguyen Thanh et al. 2023), we additionally utilize the exploration parameter ϵ to control the randomness. The Gumbel-Softmax for the discrete attribute generation is defined as:

$$\text{Gumbel-Softmax}(\mathbf{F}_i, \epsilon)_j = \frac{\exp\left((F_{ij} + \epsilon N_j)/\tau\right)}{\sum_{j'=1}^{D} \exp\left((F_{ij'} + \epsilon N_{j'}/\tau)\right)} \quad (4)$$

where F_{ij} is the j-th element of $\mathbf{F}_i$, $N_j = -\log(-\log(U_j))$, $U_j \sim \text{Uniform}(0, 1)$ is a Uniform random variable, and τ is the temperature parameter. The Gumbel-Top-k function $\mathcal{G}^a$ for discrete attribute generation is defined as:

$$\mathcal{G}^a(\mathbf{F}_i) = \arg\underset{j}{\text{top-}k}\ \text{Gumbel-Softmax}(\mathbf{F}_i, \epsilon)_j \quad (5)$$

where $\arg$ top-k function returns the indices corresponding to the k highest values of the input. Here, we set k as λ, the average count of non-zero values in the node attributes of the entire graph. Finally, the j-th element of $\widehat{\mathbf{x}}_i$ becomes $\widehat{x}_{ij} = \mathbb{I}(j \in \mathcal{G}^a(\mathbf{F}_i))$. We employ the straight-through estimator to optimize the discrete selection processes, allowing for gradient-based optimization with discrete elements.

Adversarial Edge Generation

To generate all adversarial edges at once within the edge budget η, we construct an edge score matrix and apply the Gumbel-Top-k. First, we project $\mathbf{Z}^{(L)}$ into edge score space:

$$\begin{aligned} \mathbf{R} &= \mathbf{W}_e \mathbf{Z}^{(L)} + \mathbf{b}_e = [\,\overline{\mathbf{R}} \,||\, \widetilde{\mathbf{R}} \,||\, \widehat{\mathbf{R}}\,] \\ &= [\,\overline{\mathbf{r}}_1 \cdots \overline{\mathbf{r}}_m \,||\, \widetilde{\mathbf{r}}_1 \cdots \widetilde{\mathbf{r}}_\alpha \,||\, \widehat{\mathbf{r}}_1 \cdots \widehat{\mathbf{r}}_\Delta] \\ &= [\,\mathbf{r}_1 \cdots \mathbf{r}_M] \end{aligned} \quad (6)$$

where $\mathbf{W}_e \in \mathbb{R}^{D_H \times D_H}$, $\mathbf{b}_e \in \mathbb{R}^{D_H}$, $\overline{\mathbf{R}} \in \mathbb{R}^{D_H \times m}$, $\widetilde{\mathbf{R}} \in \mathbb{R}^{D_H \times \alpha}$, $\widehat{\mathbf{R}} \in \mathbb{R}^{D_H \times \Delta}$, m is the number of target nodes, α is the number of candidate nodes, Δ is a node budget, and $M = m + \alpha + \Delta$. We define the edge score e_{ij} between an attack node $u_i \in \mathcal{V}_{\text{in}}$ and a node $v_j \in \mathcal{T} \cup \mathcal{C} \cup \mathcal{V}_{\text{in}}$ as the cosine similarity between their representations. Then, the edge score matrix $\widehat{\mathbf{E}} \in \mathbb{R}^{\Delta \times M}$ can be written as $\widehat{\mathbf{E}} = \widehat{\mathbf{D}}^{-1} \widehat{\mathbf{R}}^\top \mathbf{R} \mathbf{D}^{-1}$ where $\mathbf{D} = \text{diag}(||\mathbf{r}_1||, \cdots, ||\mathbf{r}_M||)$, and $\widehat{\mathbf{D}} = \text{diag}(||\,\widehat{\mathbf{r}}_1||, \cdots, ||\,\widehat{\mathbf{r}}_\Delta||)$. To guarantee that every attack node is directly connected to the original graph, we generate at least a single edge for each attack node to one of the target nodes by applying Gumbel-Top-k with $k = 1$. For all remaining possible edges, we apply Gumbel-Top-k across the entire $\widehat{\mathbf{E}}$ with $k = \eta - \Delta$. Note that edge scores corresponding to self-loops and duplicated edges are masked.

Training of MonTi

Following the previous works in graph injection attacks (Tao et al. 2021; Wang et al. 2022), we define the loss function for MonTi based on C&W loss (Carlini and Wagner 2017):

$$\min_{G'} \mathcal{L}(f_{\theta^*}, G', \mathcal{T}) = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} \max\left(s'_{t,1} - s'_{t,0}\,,\, 0\right) \quad (7)$$

where $\mathbf{s}'_t = f_{\theta^*}(G', t) \in \mathbb{R}^2$, and $s'_{t,i}$ denotes the i-th element of $\mathbf{s}'_t$. We focus on increasing normal scores and decreasing fraud scores of target nodes to align with our attack scenarios. In line with our emphasis on black-box attack settings, the loss is calculated using a surrogate model.

Experiments

We compare MonTi with the state-of-the-art graph injection attack baselines on five real-world graphs.

Datasets Our experiments on multi-target graph injection attacks cover three real-world datasets: *GossipCop-S*, *YelpChi*, and *LifeIns*. *GossipCop-S* and *YelpChi* have continuous node attributes, whereas *LifeIns* contains discrete ones. Using *GossipCop* (Shu et al. 2020), which includes news articles and their Twitter engagements (Shu et al. 2020), we create *GossipCop-S* by following (Wu and Hooi 2023), linking articles tweeted by the same multiple users. *YelpChi* (Rayana and Akoglu 2015; Mukherjee et al. 2013) is a review graph where nodes represent reviews. *LifeIns* is a medical insurance graph based on real-world data provided by an anonymous insurance company. In *LifeIns*, nodes correspond to claims, and edges represent relationships predefined by domain experts. In experiments, we use $p\%$ of nodes as training sets, setting $p = 40$ for *GossipCop-S* and *YelpChi*, and $p = 10$ for *LifeIns*. The remaining nodes are split into validation and test sets with a ratio of 1:2, following the conventional setting in the GNN-based fraud detection (Tang et al. 2022). We create the training, validation, and test target sets with fraud nodes belonging to each split. Each target set represents a fraud gang organized based on metadata or relations in each dataset. The statistic of datasets for multi-target attacks is summarized in Table 2. More detailed descriptions of the datasets are in Appendix B.

Although MonTi is designed for multi-target attacks, we present benchmark results for single-target attacks on *OGB-Prod* (Hu et al. 2020) and *PubMed* (Sen et al. 2008) in Appendix E.1. Despite not being specifically designed for single-target attacks, MonTi achieves the highest misclassification rates on both datasets. This demonstrates the effectiveness of MonTi's transformer-based architecture, which captures intricate interactions around a target node via adversarial structure encoding.

Budgets Due to the diverse sizes and substructures of target sets, node and edge budgets should be allocated according to the characteristics of each target set. We also impose limits on the budgets since excessively large budgets can lead to highly noticeable and easy attacks. The node budget Δ for each target set is defined as $\Delta = \max(\lfloor \rho \cdot \min(B, \overline{B}) + 0.5 \rfloor, 1)$ where ρ is a parameter to control node budgets, $B = |\mathcal{N}^{(1)} \cup \mathcal{T}|$, and $\overline{B}$ is the average value of B across all target sets within the dataset. The edge budget η for each target set is calculated as $\eta = \Delta \cdot \max(\lfloor \min(d_{\mathcal{T}},\ \xi \cdot \overline{d}) + 0.5 \rfloor, 1)$ where $d_{\mathcal{T}}$ is the average node degree of the target set, ξ is a parameter to control edge budgets, and $\overline{d}$ is the average node degree of all nodes in the graph. Unless specifically stated otherwise, we set $\rho = 0.05$, $\xi = 0.1$ for *GossipCop-S*, $\rho = 0.05$, $\xi = 0.5$ for *YelpChi*, and $\rho = 0.2$, $\xi = 0.5$ for *LifeIns*. A detailed explanation of the rationale behind defining the budgets and analysis of the effect of ρ and ξ is provided in Appendix E.2.

	$\lvert\mathcal{V}\rvert$	#Frauds	#Target Sets	$\lvert\mathcal{E}\rvert$	D
GossipCop-S	16,488	3,898	2,438	3,865,058	768
YelpChi	45,900	6,656	1,435	3,846,910	32
LifeIns	122,792	1,264	380	912,833	1,611

Table 2: Statistic of datasets for multi-target attacks.

Fraud Detectors, Baselines, and Implementation Details We consider following models as surrogate and victim models: GCN (Kipf and Welling 2017), GraphSAGE (Hamilton, Ying, and Leskovec 2017), GAT (Veličković et al. 2018), CARE-GNN (Dou et al. 2020), PC-GNN (Liu et al. 2021), and GAGA (Wang et al. 2023c). Since we focus on graph injection evasion attacks where attack nodes are injected into the original graph during the inference phase, we exclude fraud detectors that cannot handle the nodes added after training, such as BWGNN (Tang et al. 2022), SplitGNN (Wu et al. 2023), and DGA-GNN (Duan et al. 2024). We use G-NIA (Tao et al. 2021), TDGIA (Zou et al. 2021), Cluster Attack (Wang et al. 2022), and G^2A2C (Ju et al. 2023) as attack baselines. Further details on the experimental environment and implementation details of the fraud detectors and attack baselines are available in Appendix C, and implementation details of MonTi are described in Appendix D.

Performance of Multi-target Attacks

Table 3 presents the results of multi-target attacks when the types of surrogate and victim models are the same. We repeat all experiments five times and report the average and standard deviation of the misclassification rates of all target sets weighted by their sizes, as the size varies across different target sets. `Clean` represents the misclassification rates on the original clean graphs. We see that GNN-based fraud detectors are more robust than vanilla GNNs due to their ability to handle heterophily. MonTi shows the best attack performance across all datasets, as it can search adversarial structures more extensively than other methods.

Table 4 shows the results in a more realistic and challenging scenario where GCN is the surrogate model and CARE-GNN, PC-GNN, and GAGA are victim models. MonTi outperforms all baselines, even when merely using GCN as the surrogate model. This demonstrates that MonTi effectively generalizes its attacks by comprehensively capturing and leveraging the interdependencies between node attributes and edges, as well as among target, candidate, and attack nodes. More detailed analyses are provided in Appendix E.3.

Dataset	Attack Method	Surrogate / Victim Model: GCN	GraphSAGE	GAT	CARE-GNN	PC-GNN	GAGA
GossipCop-S	`Clean`	46.70	26.04	11.29	48.02	55.62	21.68
	G-NIA	75.12±0.11	67.70±2.05	63.21±3.33	59.96±2.89	62.60±1.53	25.69±1.44
	TDGIA	42.93±0.26	41.07±0.47	24.70±0.62	57.49±0.11	62.24±0.12	22.09±0.05
	Cluster Attack	43.67±0.21	39.89±0.49	24.88±0.33	57.12±0.17	61.83±0.15	22.09±0.05
	G^2A2C	OOM	OOM	OOM	OOM	OOM	OOM
	MonTi	**92.60±0.34**	**97.05±0.15**	**94.30±2.55**	**90.15±0.44**	**90.12±0.17**	**46.94±1.48**
YelpChi	`Clean`	87.14	43.81	35.12	29.79	59.13	28.00
	G-NIA	90.93±0.90	64.56±1.34	55.51±11.72	**32.45±1.13**	63.18±1.71	31.08±0.43
	TDGIA	86.87±0.10	43.28±0.23	46.95±1.14	30.66±0.24	58.01±0.21	28.03±0.02
	Cluster Attack	86.97±0.04	43.72±0.05	45.16±0.80	30.13±0.05	58.34±0.15	28.02±0.05
	G^2A2C	OOM	OOM	OOM	OOM	OOM	OOM
	MonTi	**92.23±0.95**	**65.31±1.19**	**93.27±7.84**	31.92±0.53	**69.93±1.31**	**37.66±0.92**
LifeIns	`Clean`	27.72	13.70	16.75	16.42	16.17	15.68
	G-NIA	33.50±4.84	13.20±1.02	16.50±0.79	16.09±0.43	16.09±0.47	17.44±0.23
	TDGIA	83.28±0.17	37.80±0.12	96.60±0.59	18.05±0.19	17.90±0.10	16.87±0.13
	Cluster Attack	N/A	N/A	N/A	N/A	N/A	N/A
	G^2A2C	45.05±1.82	13.00±0.05	35.85±2.02	17.24±0.00	20.08±0.04	OOM
	MonTi	**99.47±0.31**	**60.97±0.97**	**100.00±0.00**	**26.80±4.29**	**20.64±0.30**	**35.03±1.54**

Table 3: Multi-target attack performance on *GossipCop-S*, *YelpChi*, and *LifeIns* where the types of surrogate and victim models are the same. We report misclassification rates (%). OOM: Out of Memory. N/A: Not completed in 5 days.

	GossipCop-S			*YelpChi*			*LifeIns*		
	Victim Model			Victim Model			Victim Model		
Attack Method	CARE-GNN	PC-GNN	GAGA	CARE-GNN	PC-GNN	GAGA	CARE-GNN	PC-GNN	GAGA
`Clean`	48.02	55.62	21.68	29.79	59.13	28.00	16.42	16.17	15.68
G-NIA	60.67±0.21	66.25±0.24	25.76±0.32	34.81±0.30	63.57±0.17	28.83±0.44	15.89±0.67	16.27±0.76	17.13±0.40
TDGIA	55.17±0.03	60.72±0.02	24.59±0.04	30.08±0.09	58.33±0.10	28.23±0.04	18.34±0.03	18.25±0.04	23.38±0.08
Cluster Attack	57.10±0.10	61.82±0.07	22.13±0.06	30.02±0.08	58.41±0.15	27.99±0.04	N/A	N/A	N/A
G^2A2C	OOM	OOM	OOM	OOM	OOM	OOM	17.24±0.00	**20.08±0.04**	OOM
MonTi	**88.40±0.42**	**89.36±0.69**	**41.34±1.90**	**55.59±2.94**	**94.21±1.79**	**29.63±0.54**	**18.63±0.63**	19.78±0.26	**27.25±1.59**

Table 4: Multi-target attack performance on *GossipCop-S*, *YelpChi*, and *LifeIns* where GCN is the surrogate model. We report misclassification rates (%). OOM: Out of Memory. N/A: Not completed in 5 days.

Ablation Studies and Efficiency Analysis of MonTi

Table 5 shows the results of ablation studies on *GossipCop-S* for MonTi with the same settings as in Table 4. We replace the adversarial attribute and edge generation methods with random generation. For attributes, nodes are randomly selected from the original graph, and their attributes are assigned to attack nodes (random attributes). For edges, connections are randomly created among target, candidate, and attack nodes (random edges). We remove the learnable positional encodings $\overline{\mathbf{p}}$, $\widetilde{\mathbf{p}}$, and $\widehat{\mathbf{p}}$ (w/o pos. encoding) and modify MonTi to select the candidates randomly (random candidates). Lastly, we fix the degree budget for each attack node (fixed budget). In conclusion, the original MonTi shows the best performance, demonstrating the importance of each component of MonTi for effective graph injection attacks. More ablation studies are presented in Appendix E.4.

We also compare MonTi with attack baselines in terms of runtime and memory usage on *GossipCop-S*, *YelpChi*, and *LifeIns*, using GCN as the surrogate model and GAGA as the victim model in Appendix E.5. Overall, MonTi is the most efficient in terms of runtime while maintaining moderate memory usage. This is because MonTi effectively narrows the search space through candidate selection and adopts efficient matrix operations to inject all attack nodes at once. In addition, the complexity analysis of MonTi is provided in Appendix E.6.

Case Study: Effects of the Size of Fraud Gangs

To analyze vulnerabilities of GNN-based fraud detectors regarding the size of fraud gangs, we categorize target sets in *GossipCop-S* into three groups based on $B = |\mathcal{N}^{(1)} \cup \mathcal{T}|$, which reflects the size of the fraud gang. Table 6 presents the multi-target attack performance of G-NIA and MonTi for each category, using GCN as the surrogate model. The results show that the disparity in misclassification rates before and after the attack widens with increasing the size of gangs (B). This highlights the threats that large-scale fraud gangs pose to GNNs. Specifically, CARE-GNN and PC-GNN, which only filter node-level camouflages, are more vulnerable to relatively large fraud gangs ($B > 10$) compared to GAGA. Notably, the performance gap between G-NIA and MonTi becomes larger as the size of gangs increases, suggesting that MonTi can explore the complex structures that could be formed by large fraud gangs more

	CARE-GNN	PC-GNN	GAGA
`Clean`	48.02	55.62	21.68
random attributes	78.18	77.81	25.70
random edges	79.64	81.40	39.33
w/o pos. encoding	86.12	86.98	40.21
random candidates	88.25	88.38	39.13
fixed budget	87.28	88.66	42.17
MonTi	**88.78**	**89.90**	**43.70**

Table 5: Ablation studies of MonTi on *GossipCop-S* with surrogate GCN. We report misclassification rates (%).

	CARE-GNN	PC-GNN	GAGA
$B \leq 10$ (#Sets = 277)			
`Clean`	48.04%	54.46%	16.39%
G-NIA	49.17%	54.53%	17.75%
MonTi	**58.69%**	**61.63%**	**27.95%**
$10 < B \leq 1000$ (#Sets = 316)			
`Clean`	51.30%	58.35%	11.59%
G-NIA	56.47%	63.73%	15.18%
MonTi	**83.39%**	**86.44%**	**27.97%**
$B > 1000$ (#Sets = 316)			
`Clean`	55.00%	61.41%	28.66%
G-NIA	65.52%	70.45%	33.21%
MonTi	**98.70%**	**98.22%**	**56.14%**

Table 6: Multi-target attack performance on *GossipCop-S* using GCN as the surrogate model, with target sets categorized into three groups based on B. #Sets denotes the number of target sets within each category.

comprehensively than G-NIA.

To analyze cases where G-NIA fails but MonTi succeeds, we show a case study on *GossipCop-S* using target sets with $B > 1000$ and GCN as the surrogate model. We visualize the latent node representations computed by GAGA before and after the attack using t-SNE (Van der Maaten and Hinton 2008) in Figure 3 and Appendix E.7. Blue circles indicate the representations of target nodes before the attack, and orange diamonds indicate those after the attack. A blue circle and an orange diamond that correspond to the same target node are connected. Below each t-SNE visualization, we provide misclassification rates before and after the attack, the size of the target set, and B. G-NIA results in minor changes in target node representations, leading to a small increase in misclassification rates. In contrast, MonTi significantly shifts the representations from the fraud to the benign area, effectively making most target nodes misclassified.

Discussion on Defenses against MonTi

Our research aims to investigate the vulnerabilities of GNN-based fraud detectors and highlight their risks by simulating attacks of fraud gangs in practical settings. In particular, our findings suggest that GNN-based fraud detectors are especially susceptible to large-scale group attacks, which could undermine their reliability in real-world applications. We discuss potential approaches that could be beneficial to safeguarding against such threats. Our promising approach is to

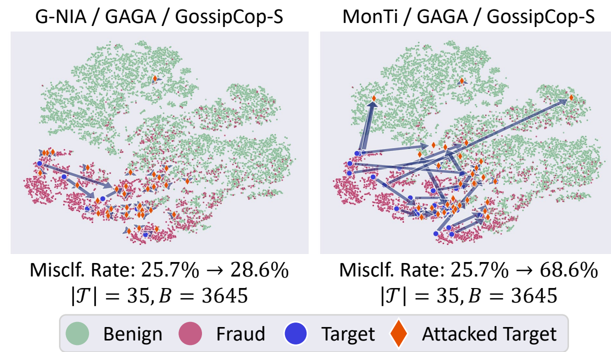

Figure 3: The t-SNE visualization of the changes in the latent representations of target nodes computed by GAGA on *GossipCop-S*, incurred by G-NIA (Left) and MonTi (Right).

extend existing graph adversarial defense methods (Zhang and Zitnik 2020; Zhang et al. 2021), such that those methods can be applied to GNN-based fraud detectors by identifying collusive patterns of fraudulent nodes. For instance, rather than solely focusing on individual nodes to detect malicious behavior, considering how the neighborhood structure around target nodes is partitioned into distinct groups could provide more effective defenses against attacks of fraud gangs. Additionally, incorporating adversarial training (Jin et al. 2020; Sun et al. 2023) with generative models could also be helpful in developing community-aware GNN-based fraud detectors. We believe that these approaches can mitigate the impact of gang-level attacks and improve the robustness of GNN-based fraud detection methods.

Conclusion and Future Work

We investigate adversarial attacks against GNN-based fraud detectors with various practical scenarios and real-world datasets. To the best of our knowledge, our work is the first study to explore such attacks against GNN-based fraud detectors and graph injection attacks for multiple target nodes formed by fraud gangs. We define this task as a multi-target graph injection attack and propose MonTi, a new adversarial attack model that injects all attack nodes at once. MonTi employs adaptive degree budget allocation for each node to explore diverse injection structures. Furthermore, MonTi simultaneously generates attributes and edges of attack nodes, considering the interdependency between them. Extensive experiments on five real-world graphs show that MonTi outperforms state-of-the-art graph injection attack methods in both multi- and single-target settings. For future work, we will extend MonTi to multi-relational graphs (Kim, Choi, and Whang 2023), as considering multiple relation types can be beneficial to effective fraud detection. We also plan to examine our work in inductive settings (Lee, Chung, and Whang 2023) where both victim and attack models do not observe test nodes and their connections during training. Moreover, we will theoretically investigate MonTi in terms of generalization bounds (Lee, Hwang, and Whang 2024).

Acknowledgments

This research was supported by an NRF grant funded by MSIT 2022R1A2C4001594 (Extendable Graph Representation Learning) and an IITP grant funded by MSIT 2022-0-00369, RS-2022-II220369 (Development of AI Technology to support Expert Decision-making that can Explain the Reasons/Grounds for Judgment Results based on Expert Knowledge).

References

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. arXiv:1308.3432.

Bojchevski, A.; and Günnemann, S. 2019. Adversarial Attacks on Node Embeddings via Graph Poisoning. In *Proceedings of the 36th International Conference on Machine Learning*, 695–704.

Carlini, N.; and Wagner, D. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, 39–57.

Chai, Z.; You, S.; Yang, Y.; Pu, S.; Xu, J.; Cai, H.; and Jiang, W. 2022. Can Abnormality be Detected by Graph Neural Networks? In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, 1945–1951.

Chen, J.; Fan, W.; Zhu, G.; Zhao, X.; Yuan, C.; Li, Q.; and Huang, Y. 2022a. Knowledge-enhanced Black-box Attacks for Recommendations. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 108–117.

Chen, Y.; Yang, H.; Zhang, Y.; Ma, K.; Liu, T.; Han, B.; and Cheng, J. 2022b. Understanding and Improving Graph Injection Attack by Promoting Unnoticeability. In *Proceedings of the 10th International Conference on Learning Representations*.

Dou, Y.; Liu, Z.; Sun, L.; Deng, Y.; Peng, H.; and Yu, P. S. 2020. Enhancing Graph Neural Network-based Fraud Detectors against Camouflaged Fraudsters. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, 315–324.

Dou, Y.; Shu, K.; Xia, C.; Yu, P. S.; and Sun, L. 2021. User Preference-aware Fake News Detection. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2051–2055.

Duan, M.; Zheng, T.; Gao, Y.; Wang, G.; Feng, Z.; and Wang, X. 2024. DGA-GNN: Dynamic Grouping Aggregation GNN for Fraud Detection. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, 11820–11828.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, 1025–1034.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *Proceedings of the 34th Conference on Neural Information Processing Systems*, volume 33, 22118–22133.

Huang, C.; and Li, H. 2023. Single-User Injection for Invisible Shilling Attack against Recommender Systems. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 864–873.

Huang, M.; Liu, Y.; Ao, X.; Li, K.; Chi, J.; Feng, J.; Yang, H.; and He, Q. 2022. AUC-Oriented Graph Neural Network for Fraud Detection. In *Proceedings of the ACM Web Conference 2022*, 1311–1321.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical Reparameterization with Gumbel-Softmax. In *Proceedings of the 5th International Conference on Learning Representations*.

Jin, W.; Li, Y.; Xu, H.; Wang, Y.; Ji, S.; Aggarwal, C.; and Tang, J. 2020. Adversarial Attacks and Defenses on Graphs: A Review, A Tool and Empirical Studies. *ACM SIGKDD Explorations Newsletter*, 22(2): 19–34.

Ju, M.; Fan, Y.; Zhang, C.; and Ye, Y. 2023. Let Graph be the Go Board: Gradient-free Node Injection Attack for Graph Neural Networks via Reinforcement Learning. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, 4383–4390.

Kim, H.; Choi, J.; and Whang, J. J. 2023. Dynamic Relation-Attentive Graph Neural Networks for Fraud Detection. In *2023 IEEE International Conference on Data Mining Workshops (ICDMW)*, 1092–1096.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*.

Kool, W.; Hoof, H. v.; and Welling, M. 2019. Stochastic Beams and Where to Find Them: The Gumbel-Top-k Trick for Sampling Sequences Without Replacement. In *Proceedings of the 36th International Conference on Machine Learning*, 3499–3508.

Lee, J.; Chung, C.; and Whang, J. J. 2023. InGram: Inductive Knowledge Graph Embedding via Relation Graphs. In *Proceedings of the 40th International Conference on Machine Learning*, 18796–18809.

Lee, J.; Hwang, M.; and Whang, J. J. 2024. PAC-Bayesian Generalization Bounds for Knowledge Graph Representation Learning. In *Proceedings of the 41st International Conference on Machine Learning*, 26589–26620.

Li, Q.; He, Y.; Xu, C.; Wu, F.; Gao, J.; and Li, Z. 2022a. Dual-Augment Graph Neural Network for Fraud Detection. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, 4188–4192.

Li, Z.; Chen, D.; Liu, Q.; and Wu, S. 2022b. The Devil is in the Conflict: Disentangled Information Graph Neural Networks for Fraud Detection. In *Proceedings of 2022 IEEE International Conference on Data Mining*, 1059–1064.

Lin, L.; Blaser, E.; and Wang, H. 2022. Graph Structural Attack by Perturbing Spectral Distance. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 989–998.

Liu, K.; Dou, Y.; Zhao, Y.; Ding, X.; Hu, X.; Zhang, R.; Ding, K.; Chen, C.; Peng, H.; Shu, K.; Sun, L.; Li, J.; Chen, G. H.; Jia, Z.; and Yu, P. S. 2022. BOND: Benchmarking Unsupervised Outlier Node Detection on Static Attributed Graphs. In *Proceedings of the 36th Conference on Neural Information Processing Systems*, 27021–27035.

Liu, Y.; Ao, X.; Qin, Z.; Chi, J.; Feng, J.; Yang, H.; and He, Q. 2021. Pick and Choose: A GNN-Based Imbalanced Learning Approach for Fraud Detection. In *Proceedings of the ACM Web Conference 2021*, 3168–3177.

Ma, J.; Deng, J.; and Mei, Q. 2022. Adversarial Attack on Graph Neural Networks as An Influence Maximization Problem. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 675–685.

Ma, J.; Li, F.; Zhang, R.; Xu, Z.; Cheng, D.; Ouyang, Y.; Zhao, R.; Zheng, J.; Zheng, Y.; and Jiang, C. 2023. Fighting against Organized Fraudsters Using Risk Diffusion-based Parallel Graph Neural Network. In *Proceedings of the 32th International Joint Conference on Artificial Intelligence*, 6138–6146.

Ma, Y.; Wang, S.; Derr, T.; Wu, L.; and Tang, J. 2021. Graph Adversarial Attack via Rewiring. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1161–1169.

Mukherjee, A.; Venkataraman, V.; Liu, B.; and Glance, N. 2013. What Yelp Fake Review Filter Might Be Doing? In *Proceedings of the 7th International AAAI Conference on Weblogs and Social Media*, 409–418.

Nguyen Thanh, T.; Quach, N. D. K.; Nguyen, T. T.; Huynh, T. T.; Vu, V. H.; Nguyen, P. L.; Jo, J.; and Nguyen, Q. V. H. 2023. Poisoning GNN-based Recommender Systems with Generative Surrogate-based Attacks. *ACM Transactions on Information Systems*, 41(3): 1–24.

Rayana, S.; and Akoglu, L. 2015. Collective Opinion Spam Detection: Bridging Review Networks and Metadata. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 985–994.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective Classification in Network Data. *AI Magazine*, 29(3): 93.

Shao, J.; Wang, Y.; Guo, F.; Shi, B.; Shen, H.; and Cheng, X. 2023. TOAK: A Topology-oriented Attack Strategy for Degrading User Identity Linkage in Cross-network Learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2208–2218.

Shi, F.; Cao, Y.; Shang, Y.; Zhou, Y.; Zhou, C.; and Wu, J. 2022. H2-FDetector: A GNN-Based Fraud Detector with Homophilic and Heterophilic Connections. In *Proceedings of the ACM Web Conference 2022*, 1486–1494.

Shu, K.; Mahudeswaran, D.; Wang, S.; Lee, D.; and Liu, H. 2020. FakeNewsNet: A Data Repository with News Content, Social Context and Spatiotemporal Information for Studying Fake News on Social Media. *Big Data*, 8(3): 171–188.

Sun, L.; Dou, Y.; Yang, C.; Zhang, K.; Wang, J.; Yu, P. S.; He, L.; and Li, B. 2023. Adversarial Attack and Defense on Graph Data: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(8): 7693–7711.

Sun, Y.; Wang, S.; Tang, X.; Hsieh, T.-Y.; and Honavar, V. 2020. Adversarial Attacks on Graph Neural Networks via Node Injections: A Hierarchical Reinforcement Learning Approach. In *Proceedings of The Web Conference 2020*, 673–683.

Tang, J.; Li, J.; Gao, Z.; and Li, J. 2022. Rethinking Graph Neural Networks for Anomaly Detection. In *Proceedings of the 39th International Conference on Machine Learning*, 21076–21089.

Tao, S.; Cao, Q.; Shen, H.; Huang, J.; Wu, Y.; and Cheng, X. 2021. Single Node Injection Attack against Graph Neural Networks. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, 1794–1803.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11): 2579–2605.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is All You Need. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, 5998–6008.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *Proceedings of the 6th International Conference on Learning Representations*.

Wang, H.; Dou, Y.; Chen, C.; Sun, L.; Yu, P. S.; and Shu, K. 2023a. Attacking Fake News Detectors via Manipulating News Social Engagement. In *Proceedings of the ACM Web Conference 2023*, 3978–3986.

Wang, J.; Luo, M.; Suya, F.; Li, J.; Yang, Z.; and Zheng, Q. 2020. Scalable attack on graph data by injecting vicious nodes. *Data Mining and Knowledge Discovery*, 34(5): 1363–1389.

Wang, L.; Zhao, H.; Feng, C.; Liu, W.; Huang, C.; Santoni, M.; Cristofaro, M.; Jafrancesco, P.; and Bian, J. 2023b. Removing Camouflage and Revealing Collusion: Leveraging Gang-crime Pattern in Fraudster Detection. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 5104–5115.

Wang, Y.; Zhang, J.; Huang, Z.; Li, W.; Feng, S.; Ma, Z.; Sun, Y.; Yu, D.; Dong, F.; Jin, J.; Wang, B.; and Luo, J. 2023c. Label Information Enhanced Fraud Detection against Low Homophily in Graphs. In *Proceedings of the ACM Web Conference 2023*, 406–416.

Wang, Z.; Hao, Z.; Wang, Z.; Su, H.; and Zhu, J. 2022. Cluster Attack: Query-based Adversarial Attacks on Graph with Graph-Dependent Priors. In *Proceedings of the 31th International Joint Conference on Artificial Intelligence*, 768–775.

Wu, B.; Yao, X.; Zhang, B.; Chao, K.-M.; and Li, Y. 2023. Split-GNN: Spectral Graph Neural Network for Fraud Detection against Heterophily. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2737–2746.

Wu, J.; and Hooi, B. 2023. DECOR: Degree-Corrected Social Graph Refinement for Fake News Detection. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2582–2593.

Xu, W.; Wu, J.; Liu, Q.; Wu, S.; and Wang, L. 2022. Evidence-aware Fake News Detection with Graph Neural Networks. In *Proceedings of the ACM Web Conference 2022*, 2501–2510.

You, X.; Li, C.; Ding, D.; Zhang, M.; Feng, F.; Pan, X.; and Yang, M. 2023. Anti-FakeU: Defending Shilling Attacks on Graph Neural Network based Recommender Model. In *Proceedings of the ACM Web Conference 2023*, 938–948.

Yu, J.; Wang, H.; Wang, X.; Li, Z.; Qin, L.; Zhang, W.; Liao, J.; and Zhang, Y. 2023. Group-based Fraud Detection Network on e-Commerce Platforms. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 5463–5475.

Zhang, X.; and Zitnik, M. 2020. GNNGuard: Defending Graph Neural Networks against Adversarial Attacks. In *Proceedings of the 34th Conference on Neural Information Processing Systems*, 9263–9275.

Zhang, Y.; Regol, F.; Pal, S.; Khan, S.; Ma, L.; and Coates, M. 2021. Detection and Defense of Topological Adversarial Attacks on Graphs. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, 2989–2997.

Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; and Koutra, D. 2020. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In *Proceedings of the 34th Conference on Neural Information Processing Systems*, 7793–7804.

Zou, X.; Zheng, Q.; Dong, Y.; Guan, X.; Kharlamov, E.; Lu, J.; and Tang, J. 2021. TDGIA: Effective Injection Attacks on Graph Neural Networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2461–2471.

Zügner, D.; Akbarnejad, A.; and Günnemann, S. 2018. Adversarial Attacks on Neural Networks for Graph Data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2847–2856.

Zügner, D.; and Günnemann, S. 2019. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *Proceedings of the 7th International Conference on Learning Representations*.