From Bi-Level to One-Level: A Framework for Structural Attacks to Graph Anomaly Detection

Yulin Zhu, Yuni Lai, Kaifa Zhao, Xiapu Luo, Mingquan Yuan, Jun Wu, Jian Ren, Kai Zhou

***Abstract*—The success of graph neural networks stimulates the prosperity of graph mining and the corresponding downstream tasks including graph anomaly detection (GAD). However, it has been explored that those graph mining methods are vulnerable to structural manipulations on relational data. That is, the attacker can maliciously perturb the graph structures to assist the target nodes to evade anomaly detection. In this paper, we explore the structural vulnerability of two typical GAD systems: unsupervised FeXtra-based GAD and supervised GCN-based GAD. Specifically, structural poisoning attacks against GAD are formulated as complex bi-level optimization problems. Our first major contribution is then to transform the bi-level problem into one-level leveraging different regression methods. Furthermore, we propose a new way of utilizing gradient information to optimize the one-level optimization problem in the discrete domain. Comprehensive experiments demonstrate the effectiveness of our proposed attack algorithm** BinarizedAttack**.**

***Index Terms*—Graph anomaly detection, Graph neural networks, Structural poisoning attack, Adversarial graph analysis, Discrete optimization.**

I. Introduction

Anomalies or outliers are ubiquitously existed in the real world [1]. They are regarded as those instances which explicitly deviated from the majority. Naturally, the relational data may also contain anomalies, which are called graph anomalies. Recently, a surge of graph learning techniques come out due to the powerful graph representation learning methods – Graph Convolutional Networks (GCNs [2] for short). As a result, plenty of graph-based anomaly detection models equip with the powerful GCN for spotting anomalies more accurately as well as providing strong references for the graph security analyst in several fields such as finance [3], social networks [4] and Botnets [5].

The tremendous success of the representation learning of GCN comes from its unique message-passing mechanism, i.e., each node aggregates the neighbors' features with a low-pass filter (symmetric normalized adjacency matrix) to smooth the features of connected node pairs. As a result, the connected node pairs' features are much closer than disconnected node pairs and thus provide high-quality node embeddings for semi-supervised node classification. GCN-based graph anomaly detection also relies on the message-passing mechanism for powerful graph representation learning to discriminate malicious nodes from benign nodes. Unfortunately, graph-based anomaly detection is prone to be influenced by the adversarial noises (maliciously add or delete links or nodes) injected by the attacker since they highly rely on the structural information of the graph data. As a motivative example, consider the problem of *misinformation diffusion* in social networks, where an attacker aims to pick out a set of nodes as the *seeds* to diffuse misinformation (e.g., fake news or hate speeches) through social media platforms. To maximize the diffusion range, the attacker can employ some *influence maximization* [6] algorithms to identify those most promising seeds, which however are prone to be labeled as anomalous by GAD systems. Meanwhile, the attacker can proactively alter the social ties (e.g., friend/unfriend with other users) of those seeds so as to prevent them from being detected. Hence, the attackers now can help those seed nodes to evade the GAD systems by perturbing the graph structure. In essence, the attacker is *poisoning* the graph data from which the GAD systems are trained, such attacks are termed as *structural poisoning attacks*. In our preliminary work [7], we showed that structural poisoning attacks can effectively help the seed nodes (selected by CELF [8], Greedy [6] and etc.) to evade graph anomaly detection.

Additionally, there also exists structural attacks against the fraudster detection system of the bank's credit card center [9]. The camouflaged fraudster tends to create malicious accounts and mimic the transaction behavior of other normal accounts to evade the detection of the anti-fraud system and then create transactions for malicious purposes such as money laundering and illegal cash out. Moreover, in the recommendation system (bipartite graph data) field [10], the anomaly detection model spots the abnormal users by exploring the user's behavior such as purchase history, etc. Thus, if the user's account is invaded by the hacker, the hacker tends to modify the purchase records to evade anomaly detection in order to engage in criminal activities secretly. The previous example shows that it is of great importance to investigate the *adversarial robustness* of current GAD systems – *how robust could those GAD systems be under carefully designed attacks?* To this end, we continue our preliminary study [7] of structural attacks against GAD systems, answering how much an attacker can help the victim nodes evade detection of the GAD systems via perturbing the graph structures. Analyzing the structural poisoning attacks on relational data plays a key role in the graph security field. Previous studies mainly focus on manipulating the feature space (PDF/Android malware detection [11]). However, the manipulated feature vectors can barely be mapped back to the real entities (PDF/Android software), thus alleviating the effects of the adversarial attacks. On the other hand, changing the graph structure directly represents changing the actual connections among entities, thus leading to a more realistic attack. Moreover, it is also possible for an attacker to have full knowledge of the graph structure and form a global attack. For example, the Command & Control center in Botnets [5]

can coordinate the communications among Bots globally and evade Botnets detection. To address this issue, it is vital to concretely analyze the structural poisoning attacks against the graph anomaly model and motivate the analyst to design a more robust graph learning model.

In our preliminary study [7], we primarily focused on a classical type of GAD systems (termed *FeXtra-based*) that rely on feature engineering. Specifically, FeXtra-based GAD systems (e.g., OddBall [12]) extract hand-crafted *structural features* of each node, then build regression models based on those structural features, and finally get the anomaly score for each node. In this work, we extend the study to GAD systems utilizing the GCNs, which are becoming the *de facto* choice in various domains, and anomaly detection is no exception. In particular, GCNs could automatically learn the anomalous patterns, which significantly simplified the manual feature designing process, and also allows GCN-based GAD systems to achieve state-of-the-art performances. However, also due to the complex structure of GCNs, our proposed techniques in [7] could not be directly applied to attacking GCN-based GAD systems – we thus propose new techniques to address the challenges rooted in GCNs (to be detailed later).

There are several challenges to attacking the GAD systems. Firstly, the GAD systems are mainly unsupervised or semi-supervised models with all nodes dependent on each other. Thus, attacking the GAD systems naturally forms a complex bi-level optimization problem [13], which the attacker should perturb the graph structure during the model training phase and iteratively optimizes the model parameters and graph topology. Especially, the bi-level optimization problem contains a two-level training phase. In the inner training phase, we optimize the GAD model's parameters via vanilla optimization methods such as gradient descent. In the outer training phase, the graph topology is optimized to maximize the training loss and decrease the detection accuracy of the GAD model. Secondly, the entries of relational data fall into the discrete domain, which makes the search space of the structural attacks exponential in the graph size. Moreover, the attacker cannot directly utilize the vanilla gradient descent method [14], [15] on the discrete space, as the attacker needs to learn binary decision variables on inserting or removing the links. To prove this issue, we provide experimental results to demonstrate that vanilla gradient descent will lead to sub-optimal structural attacks.

We thus present a framework for structural poisoning attacks against GADs to address the above challenges. This framework relies on two central innovations: an approach for reformulating bi-level optimization to one-level and a new way of utilizing gradient information for optimization problems over discrete data. Specifically, the hardness of solving the bi-level optimization comes from the fact that the inner optimization involves the training of a GAD system (refer to Section IV-B), which makes gradient-based methods not applicable. Fortunately, FeXtra-based GAD systems such as OddBall have a straightforward solution: exploiting the existence of a close-form solution of linear regression (as we did in [7]) can easily transform the bi-level optimization to one-level. However, this nice feature cannot be exploited by GCN-based GAD systems. To address this issue, we concentrate on the two-layered linearized GCN [16], and innovatively utilize the Ridge weighted least square estimation [17], [18] to approximate its training process. As a result, we can directly represent GCN weights by an explicit mapping of the adjacency matrix and nodal attributes and reformulate the bi-level objective function to a simplified one-level case.

After the one-level reformulation, we propose the BinarizedAttack method, where the central idea sheds light on a more effective way of using gradients. BinarizedAttack mimics the training procedure of the *Binarized Neural Networks* (BNN) [19], which is initially designed for model compression of the very deep neural networks. The BNN transforms the model weights from the continuous space $\mathcal{R}$ to binary values $\{+1, -1\}$ to reduce the model storage space. To optimize the model weights on the discrete domain, BNN relaxes the discrete variable to a continuous version during the backward phase. On the other hand, it adopts a piecewise function to map the continuous weights to $\{+1, -1\}$ during the forward pass. In light of this, BinarizedAttack regards the decision variable for each node pair in the graph as the binary weight to be optimized and then uses a novel projection gradient descent method to tackle this discrete optimization problem. Similarly, BinarizedAttack computes the attack loss function on the discrete domain and update the continuous version of the decision variable based on the fractional gradients. In a sense, BinarizedAttack could utilize the gradient information more precisely compared with vanilla gradient descent or greedy approach. We also conduct comprehensive experiments and show the effectiveness of BinarizedAttack compared with other baselines.

Our main contributions are summarized as follows:

- We propose a framework for designing structural poisoning attacks against the graph-based anomaly detection system, which covers unsupervised FeXtra-based and supervised GCN-based GAD systems.
- We formulate and simplify the complex structural attacks against GAD systems from bi-level optimization to one-level optimization problems by utilizing multiple regression methods.
- We propose a new gradient-based attacking method BinarizedAttack to tackle combinatorial optimization problems for discrete graph data.
- We conduct a comprehensive analysis of the performance of our proposed attacks as well as the possible side effects of attacks over several synthetic and real-world datasets.

To make this extended version self-contained, we included some key results from [7] in Sec. VI (The attack methods comparison in [7] which are consistent with Fig. 4). The rest of the paper is organized as follows: The related works are highlighted in Sec. II. We introduce the target GAD systems (including FeXtra-based and GCN-based) in Sec. III and formulate the structural poisoning attacks to the one-level optimization problem in Sec. IV. The attack methods based on three different gradient-utilization ways are introduced in Sec. V. We conduct comprehensive experiments to evaluate our methods against the GCN-based GAD systems from

several aspects (attack effectiveness against surrogate model; black-box attacks against GCN-based GAD systems; ablation and sensitivity analysis on the regression; side effects on graph structure) in Sec. VI. Finally, we conclude in Section VII.

II. Related Works

A. Graph Anomaly Detection

The goal of graph anomaly detection (GAD) is to spot anomalous instances (link, node, sub-graph or graph) among the relational data. The powerful graph representation learning stimulates the wide application of graph data in the real world such as social networks, the Internet and finance. In this paper, we concentrate on spotting nodal anomalies on the static graphs [20]. Typical GAD systems can be partitioned into four classes: *feature-based* methods [12], *Proximity-based* methods [21], *Community-based* methods [22] and *Relational-learning-based* methods [23]. Especially, feature-based methods extract crafted-designed structural features from the graph data and utilize machine learning methods for anomaly detection. Proximity-based methods determine the anomalies by measuring the distance among nodes and the anomaly nodes are far away from other nodes. Community-based methods utilize the community detection model to classify abnormal nodes from normal nodes. Relational learning methods treat it as a supervised classification problem by utilizing graphical models. Recently, a surge of GAD systems relies on powerful GCN to identify anomalous instances in the graph data [24]–[28]. They outperform other methods of identifying anomalous nodes due to the impactful propagation step.

B. Adversarial Graph Analysis

Recently, there exists a surge of literature that focus on the adversarial robustness of graph learning tasks, like node classification [14], [15], [29], link prediction [30], [31], community detection [32], etc. All these methods formulate the structural poisoning attacks as a complex bi-level discrete optimization problem and can be classified into two categories: task-specific and general. The task-specific methods highly depend on the target model. For example, [30] studies the submodularity of the attack objective and theoretically proposes the approximation algorithm derived from the sub-modular property. In contrast, the general way is to adopt the vanilla gradient descent with the greedy approach to optimize the attack objective. For example, [14] optimizes the structural perturbations by maximizing the classification margin of the victim nodes and make the decision by choosing the instances with top-K largest gradients. [15] alter the graph structure by computing the meta-gradients on the complex bi-level optimization problem. However, the above-mentioned methods can not be directly implemented in GAD since the main difference between GAD and other graph-based downstream tasks is that there exists a severe class-imbalance problem which makes the surrogate model hard to optimize in the inner training loop. Additionally, they still rely on tackling the complex bi-level optimization problem and the attack effectiveness highly relies on the optimization degree of the inner training and easily falls into sub-optimal results. Moreover, current methods do not appropriately deal with the complex discrete optimization problem when optimizing the graph's topology. The greedy approach and optimizing on the continuous space are compromised ways and will lead to sub-optimal results. In this paper, we adapt the greedy approach like [15] to the graph anomaly detection scenario, leading to a strong baseline named GradMaxSearch, and the vanilla gradient descent method on the adjacency matrix as ContinuousA. We later provide the insights and experiment results to demonstrate the advantage of our BinarizedAttack method compared to GradMaxSearch and ContinuousA.

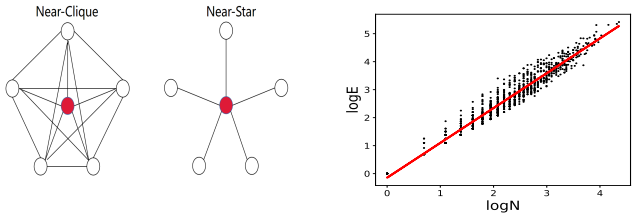
III. Background on GAD Systems

In this section, we provide the necessary background on the two families of GAD systems – one based on feature extractions and the other based on GCNs.

A. FeXtra-Based GAD Systems

We introduce OddBall [12] as one of the representative Feature eXtraction (FeXtra for short) based GAD systems that we aim to attack. At a high level, OddBall firstly distills the hand-crafted nodal structural features, and then assigns the anomaly score for each node based on the structural features. At last, it regards those nodes with large anomaly scores as anomalous.

In this context, we define the graph as $\mathcal{G} = (V, E)$, where V is the node set and E is the edge set. We denote the adjacency matrix of the static unweighted graph $\mathcal{G}$ as $\mathbf{A} \in \{0,1\}^{N\times N}$. OddBall concentrates on the local structure of the nodes for graph anomaly detection. To be detailed, it focuses on the Ego-network ego_i of each node v_i, where ego_i is a sub-graph centered at the node v_i and contains v_i's one-hop neighbors. Referring to [12], A vital point is that the Ego-network of anomalous nodes tend to be either *near-clique* or *near-star* (as shown in Fig. 1a). For anomaly detection, OddBall directly picks out the local structural features E_i and N_i of each node to check their statistics, where E_i and N_i represent the number of links and nodes in the Ego-network. An important finding is that E_i and N_i follow the *Egonet Density Power Law* distribution [12], i.e., $E_i \propto N_i^{\alpha}, 1 \leq \alpha \leq 2$. Those nodes whose structural features are highly deviated from this law are regarded as anomaly.



(a) Anomalous patterns (b) Linear regression

Fig. 1: (a) The *near-clique* and *near-star* anomalous patterns. (b) Linear regression on $\log N$ and $\log E$. (adapted from [7])

OddBall utilizes the distance between the scatters and the regression line to quantify the anomaly score. Based on the power law, $\log E$ and $\log N$ have the linear relationship, i.e.,

$$\ln E_i = \beta_0 + \beta_1 \ln N_i + \epsilon. \tag{1}$$

Then, we can adopt the Ordinary Least Square (OLS) [33] estimation to get the close form of the model parameters:

$$[\beta_0, \beta_1] = ([\mathbf{1}, \ln \mathbf{N}]^T [\mathbf{1}, \ln \mathbf{N}])^{-1} [\mathbf{1}, \ln \mathbf{N}]^T \ln \mathbf{E}, \tag{2}$$

where $\mathbf{1}$ is an n-dimensional vector of all 1's. The nodal anomaly score $S_i(\mathbf{A})$ is calculated as:

$$S_i(\mathbf{A}) = \frac{\max(E_i, e^{\beta_0} N_i^{\beta_1})}{\min(E_i, e^{\beta_0} N_i^{\beta_1})} \ln(|E_i - e^{\beta_0} N_i^{\beta_1}| + 1). \tag{3}$$

It is worth noting that the anomaly score $S_i(\mathbf{A})$ directly relies on the adjacency matrix $\mathbf{A}$ instead of the model parameters or the structural features. This kind of formation can simplify our attack objective to be detailed later.

B. GCN-Based GAD Systems

With the advances in graph neural networks, an increasing number of GAD systems are using GCN [2] as the backbone for automatically learning useful anomalous patterns. In essence, anomaly detection is cast as a supervised classification process where the labels of a portion of nodes are provided for training. Specifically, GCN-based methods will take as input an attributed graph whose nodes are associative with attribute vectors, and generate node embeddings through several convolutional layers as follows:

$$\mathbf{H}^{t+1} = \sigma(\tilde{\mathbf{A}}\mathbf{H}^t\mathbf{W}), \text{where} \tag{4a}$$

$$\tilde{\mathbf{A}} = \text{diag}(\sum_{i=1}^{n} \mathbf{A}_i)^{-\frac{1}{2}} (\mathbf{A} + \mathbf{I}) \text{diag}(\sum_{i=1}^{n} \mathbf{A}_i)^{-\frac{1}{2}} \text{ and} \tag{4b}$$

$$\mathbf{H}^0 = \sigma(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}). \tag{4c}$$

In the above, $\mathbf{A}$ and $\mathbf{X}$ are the adjacency matrix and feature matrix, respectively; $\mathbf{W}$ summarizes the learned model parameters; $\mathbf{H}^i$ are the learned embeddings through the layers. Finally, the embeddings are fed into a fully connected layer for nodal classification (Mostly are binary classification tasks where the label represents the node are benign or abnormal).

The specific methods in this family integrate some minor modifications into this framework to adapt to the various application scenarios that they are designed for. In the following, we introduce six typical GCN-based GAD systems and highlight their differences:

- **GCN-reweight** [2]. This is the GCN model with class-specific loss weight to mitigate the imbalance problem. It assigns higher weight to the minority class to let the model focus more on the minority class.
- **GAT-reweight** [34]. This is the GCN model augmented with the graph attention mechanism to automatically distinguish the contributions from different neighbors in the aggregation phase.
- **FdGars** [24]. It adopts GCN for fraudulent detection in the online APP review system by utilizing important characteristics like similarity, special symbols, timestamps, device and login status.
- **GEM** [25]. It is the first heterogeneous graph neural network model to detect anomalous accounts at Alipay. It also augments the attention mechanism to control the contributions from different nodes and utilizes the EM algorithm to iteratively update the node embeddings and model parameters.
- **Player2vec** [26]. It is specially designed to detect the cybercriminals in the e-commercial system by adopting GCN augmented with the attention mechanism to automatically capture the anomalous behaviors in the environment.
- **GraphSMOTE** [27]. GraphSMOTE adopts synthetic minority oversampling techniques (SMOTE [35]) to generate an augmented graph to mitigate the class imbalance problem in the bot detection field by using the synthetic node generator and edge generator to interpolate new synthetic nodes and edges for the minority class. Then the GCN classifier is implemented on the augmented graphs for imbalanced node classification.
- **PAMFUL** [36]. It is a synergistic approach that incorporates pattern mining and supervises the GCN training via the crafted error-bounded distribution-aware anomaly margin loss. In this way, the model can effectively incorporate global and local anomaly patterns for high-quality node representations.

IV. Formulation of Attacks

A. Threat Model

To mimic the security field in the real world, we consider a scenario with three parties: analyst, attacker and environment, where the interplay among these three parties is shown in Fig. 2. Especially, the analyst's goal is to utilize the GAD system to detect the anomalies in the graph data collected from the environment. However, the graph data is not readily available in the environment, that is, the analyst should construct the graph data via data collection. The data collection is a querying process where the analyst queries node pairs (u, v) in the environment, then the environment will feedback on the results whether (u, v) is equal to 0 or 1 (relation existence between node u and v). Last, the analyst can construct the graph by integrating all the querying results. For example, the query can be taking the question paper on friendships, communication channels, financial transactions, literature references, etc.

Alternatively, the attacker can inject noises during the data collection phase and contaminate the query results in the environment. For instance, the analyst in Fig. 2 makes a query "Do C and D have friendship with each other?", the attacker can contaminate the query results "$\{C, D\} = 1$" (existence) to "$\{C, D\} = 0$" (non-existence). As a result, the attacker deletes an edge in the graph data constructed by the analyst. From this perspective, the attacker can manipulate the graph structure by tampering with the querying results and resulting in structural attacks on the graph data. The attacker's goal, knowledge and ability are:

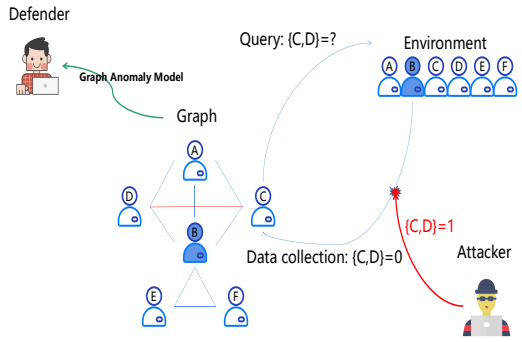


Fig. 2: Interplay among analyst, attacker and the environment. The attacker injects structural poisoning attacks to the environment, and the defender collects the poisoned relational data from the environment.

- **Attacker's Goal.** For attacking an unsupervised learning approach like OddBall, it is assumed that the attacker assigns a target node set in the graph data representing the victim nodes with initially high-level anomaly scores. The attacker aims at helping these victim nodes to evade anomaly detection. Under a supervised setting, the attacker's objective is to degenerate the overall detection performance of the GCN-based GAD systems.
- **Attacker's Knowledge** It is assumed that the attacker has full knowledge of the graph information. Regarding the deployed GAD systems, we consider different cases where the attacker may or may not know the exact model in deployment.
- **Attacker's Capability.** The attacker can fully control the structure of the network and insert or remove the links from the graph data at most B times to save the attacker's resources as much as possible and prevent significant topology changes in the graph data.

B. Problem Formulation

In this section, we first universally formulate the structural poisoning attacks against various GAD systems as bi-level discrete optimization problems. Then, we instantiate the attack problems against OddBall and GCN-based methods, respectively, based on their unique characteristics.

We use $\mathcal{G}_0 = \{\mathbf{V}_0, \mathbf{E}_0, \mathbf{X}_0, \mathbf{Y}_0\}$ to represent the oracle graph in the environment, where $\mathbf{Y}_0$ are the binary (anomalous or not) labels of the nodes. In particular, $\mathbf{X}_0 \in \mathbb{R}^{n\times p}$ denotes the attribute matrix for the nodes, where each node has an attribute vector of dimension p. When the node attributes are not available, we set $\mathbf{X}_0 = \emptyset$. In structural attacks, the attacker will manipulate the graph structure in the data collection phase, resulting in a manipulated graph $\mathcal{G} = \{\mathbf{V}_0, \mathbf{E}, \mathbf{X}_0, \mathbf{Y}_0\}$ *observed* by the defender. We note that only the graph structure $\mathbf{E}_0$ will be modified to $\mathbf{E}$. We use $\mathbf{A}_0$ and $\mathbf{A}$ to represent the adjacency matrices of $\mathcal{G}_0$ and $\mathcal{G}$, respectively. Then the structural poisoning attacks against the GAD systems can be formulated as a bi-level optimization problem:

$$\mathbf{A}^* = \arg\min_{\mathbf{A}} \mathcal{L}_{atk}(\mathbf{A}, \mathbf{X}, \theta^*, \mathbf{Y}), \tag{5a}$$

$$\text{s.t.} \quad \theta^* = \mathsf{Train}(\mathbf{A}, \mathbf{X}, \theta, \mathbf{Y}), \tag{5b}$$

$$\frac{1}{2}||\mathbf{A}_0 - \mathbf{A}||_1 \leq B, \quad \mathbf{A} \in \{0,1\}^{n\times n}. \tag{5c}$$

Here, in (5a) the loss function $\mathcal{L}_{atk}(\cdot)$ quantifies the attacker's malicious goal of evading detection. In (5b), we use $\mathsf{Train}(\cdot)$ to denote the training process of a GAD system, where the output is the optimal model parameter θ^* by training the surrogate model. (5c) specifies the budget constraint on the attacker's power. We emphasize that we solve the complex bi-level optimization problem where whenever one tries to optimize graph structure $\mathbf{A}$, the optimal model parameter θ^* would change according to a complex training process to the one-level case by finding an appropriate point estimation θ^* for different scenarios.

.

V. The Framework

In this section, we illustrate the framework for designing structural poisoning attacks against GAD systems. First, we show how to transform the bi-level optimization to one-level for different target GAD systems. Then, we introduce our new method BinarizedAttack to effectively solve the resulting one-level discrete optimization problem.

A. One-Level Approximation

1) Attacking GCN-based GAD Systems: The primary challenge for attacking GCN-based GAD systems is that the training of GCNs itself is a non-differentiable optimization process. Consequently, the typical gradient-descent method is not applicable. In addition, each GCN-based GAD model has its own characteristics, which makes designing the corresponding attacks a tedious task.

Our idea to address these issues is to leverage a simplified model termed LGCN as a surrogate, which represents the common graph convolutional process shared by all those GCN-based GAD systems. Thus, by attacking this surrogate LGCN model, we can generalize the attacks to all GCN-based GAD systems. Moreover, a nice feature of LGCN is that we can use closed-form functions via Ridge weighted least square estimation [17], [18] to approximately estimate the parameters θ^*. As a result, by substituting θ^* with the closed-form functions, we can reformulate the bi-level optimization (Eqn.(5)) as a one-level problem.

Specifically, we construct a simplified two-layer GCN model LGCN similar to [14] by replacing the non-linear activation function (e.g., ReLU [37]) by the linear activation function:

$$\tilde{\mathbf{A}} = \text{diag}(\sum_{i=1}^{n} \mathbf{A}_i)^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\text{diag}(\sum_{i=1}^{n} \mathbf{A}_i)^{-\frac{1}{2}}, \tag{6a}$$

$$\mathbf{Z} = \text{sigmoid}(\tilde{\mathbf{A}}\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}^1\mathbf{W}^2) = \text{sigmoid}(\tilde{\mathbf{A}}^2\mathbf{X}\mathbf{W}), \tag{6b}$$

where $\mathbf{W}^1$ and $\mathbf{W}^2$ represent the weights for the first and second GCN layer. The goal of removing the nonlinear activation function in the surrogate model is to ease the training

of the inner training loop in Eqn. 5b without sacrificing its performance since it has been theoretically proved that the powerful graph representation learning of GCN mainly comes from the message-passing mechanism instead of the nonlinear mapping [16]. We use the sigmoid function instead of softmax in order to reduce the nonconvex deep learning model to convex logistic regression. We further define a *weighted* binary cross-entropy loss as:

$$\mathcal{L}_{\text{R-BCE}} = -\sum_{i=1}^{n}(\omega y^i \log(\mathbf{Z}_i) + (1-y^i)\log(1-\mathbf{Z}_i)), \quad (7)$$

where ω is the ratio of the positive samples with respect to negative samples. However, the optimal $\mathbf{W}^*$ still need to be found by gradient descent. To address this issue, we replace the cross-entropy loss with the mean square loss with:

$$\begin{aligned}\mathcal{L}_{\text{R-MSE}} &= \sum_{i=1,y^i=1}^{n}(\omega(y^i-(\tilde{\mathbf{A}}^2\mathbf{X}\mathbf{W})_i))^2 + \sum_{i=1,y^i=0}^{n}(y^i-(\tilde{\mathbf{A}}^2\mathbf{X}\mathbf{W})_i)^2 \\ &= (\mathbf{Y}-\tilde{\mathbf{A}}^2\mathbf{X}\mathbf{W})^T\mathbf{D}(\mathbf{Y}-\tilde{\mathbf{A}}^2\mathbf{X}\mathbf{W}),\end{aligned} \quad (8)$$

where

$$\mathbf{D} = \begin{pmatrix} \omega^{y^1} & 0 & \dots & 0 \\ 0 & \omega^{y^2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \omega^{y^n} \end{pmatrix}$$

is a diagonal matrix. $y_i \in \{0,1\}$ is the label for node i. That is, if $y^i = 1$, $\omega^{y^i} = \omega$; if $y^i = 0$, $\omega^{y^i} = 1$. Eqn. (8) is actually the loss function of the weighted least square (WLS [18]) problem. To optimize this objective, we compute the gradient of $\mathcal{L}_{\text{R-MSE}}$ w.r.t $\mathbf{W}$ and set it to zero:

$$\frac{\partial \mathcal{L}_{\text{R-MSE}}}{\partial \mathbf{W}} = -(\tilde{\mathbf{A}}^2\mathbf{X})^T\mathbf{D}\mathbf{Y} + (\tilde{\mathbf{A}}^2\mathbf{X})^T\mathbf{D}\tilde{\mathbf{A}}^2\mathbf{X}\mathbf{W} = 0, \quad (9)$$

then the point estimate of WLS is

$$\mathbf{W}^* = ((\tilde{\mathbf{A}}^2\mathbf{X})^T\mathbf{D}\tilde{\mathbf{A}}^2\mathbf{X})^{-1}(\tilde{\mathbf{A}}^2\mathbf{X})^T\mathbf{D}\mathbf{Y}. \quad (10)$$

That is, instead of training the model to obtain $\mathbf{W}^*$, we can use Eqn. (10) to directly compute $\mathbf{W}^*$.

An additional consideration is that the matrix $(\tilde{\mathbf{A}}^2\mathbf{X})^T\mathbf{D}\tilde{\mathbf{A}}^2\mathbf{X}$ might be singular when $p > n$, which does not have an inverse. Thus, we add Ridge penalty [17] to $\mathcal{L}_{\text{R-MSE}}$ to ameliorate the high dimensional problem, which results in:

$$\mathcal{L}_{\text{RR-MSE}} = (\mathbf{Y}-\tilde{\mathbf{A}}^2\mathbf{X}\mathbf{W})^T\mathbf{D}(\mathbf{Y}-\tilde{\mathbf{A}}^2\mathbf{X}\mathbf{W}) + \xi\mathbf{W}^T\mathbf{W}. \quad (11)$$

Similarly, we can get the corresponding point estimate of $\mathbf{W}^*$ by zeroing out the gradient of $\mathcal{L}_{\text{RR-MSE}}$ similar to Eqn.(9):

$$\mathbf{W}^* = ((\tilde{\mathbf{A}}^2\mathbf{X})^T\mathbf{D}\tilde{\mathbf{A}}^2\mathbf{X} + \xi\mathbf{I}_{p\times p})^{-1}(\tilde{\mathbf{A}}^2\mathbf{X})^T\mathbf{D}\mathbf{Y}. \quad (12)$$

Since the GCN-based GAD system is a supervised structural poisoning attack, the attacker's goal is to decrease the classification accuracy of the surrogate model in the previous context under the limited budgets. Thus, the attack problem is reformulated as:

$$\begin{aligned}\mathbf{A}^* &= \arg\min_{\mathbf{A}} -\mathcal{L}_{\text{R-BCE}}(\mathbf{A},\mathbf{X},\mathbf{W}^*,\mathbf{Y}); \\ \text{s.t.} \quad &\mathbf{W}^* = \arg\min_{\mathbf{W}} \mathcal{L}_{\text{RR-MSE}}(\mathbf{A},\mathbf{X},\mathbf{W},\mathbf{Y}), \\ &\frac{1}{2}||\mathbf{A}_0-\mathbf{A}||_1 \le B, \quad \mathbf{A}\in\{0,1\}^{n\times n}.\end{aligned} \quad (13)$$

Here we emphasize that the attack loss $\mathcal{L}_{\text{R-BCE}}$ can be partitioned into two parts to incorporate the information of the training and testing data:

$$\mathcal{L}_{\text{R-BCE}}(\mathbf{Y}) = h\mathcal{L}_{\text{R-BCE}}(\mathbf{Y}^{train}) + (1-h)\mathcal{L}_{\text{R-BCE}}(\hat{\mathbf{Y}}^{test}), \quad (14)$$

where $\mathbf{Y}^{train}$ is the training node labels, $\hat{\mathbf{Y}}^{test}$ is the predictions for the testing node labels based on the pre-trained LGCN. Referring to [15], we choose $h = 0.5$ during the training phase. With the help of the closed-form of $\mathbf{W}^*$, we can transform the bi-level optimization problem to the one-level:

$$\begin{aligned}\mathbf{A}^* &= -\arg\min_{\mathbf{A}} \mathcal{L}_{\text{R-BCE}}(\mathbf{A},\mathbf{X},\mathbf{W}^*,\mathbf{Y}); \\ \text{s.t.} \quad &\tilde{\mathbf{A}} = \text{diag}(\sum_{i=1}^{n}\mathbf{A}_i)^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\text{diag}(\sum_{i=1}^{n}\mathbf{A}_i)^{-\frac{1}{2}}, \\ &\mathbf{W}^* = ((\tilde{\mathbf{A}}^2\mathbf{X})^T\mathbf{D}\tilde{\mathbf{A}}^2\mathbf{X} + \xi\mathbf{I}_{p\times p})^{-1}(\tilde{\mathbf{A}}^2\mathbf{X})^T\mathbf{D}\mathbf{Y}^{train}, \\ &\frac{1}{2}||\mathbf{A}_0-\mathbf{A}||_1 \le B, \quad \mathbf{A}\in\{0,1\}^{n\times n}.\end{aligned} \quad (15)$$

Solving $\mathbf{A}^*$ leads to an optimal solution to attacking the GCN-based GAD systems.

2) Attacking FeXtra-Based GAD Systems: Under this scenario, it is assumed that the attacker has a target node set $\mathcal{T} \subset V_0$, and the attacker aims at decreasing the anomalous probabilities of the target nodes. As previously mentioned, the attacker can insert or remove at most B edges to the clean graph $\mathcal{G}_0$ to minimize the sum of anomaly scores of target nodes. To formalize, we firstly simplify the original objective S_i by removing the normalization, non-linearity and log transformation and get $\tilde{S}_{\mathcal{T}}(\mathbf{A}) = \sum_{i:v_i\in\mathcal{T}}(E_i - e^{\beta_0^*}N_i^{\beta_1^*})^2$. It is worth noting that the simplified objective is only used for the training process. Secondly, we adopts the OLS estimation to get the close form of the model weights, and replace them with an explicit mapping of the adjacency matrix. Finally, it is natural to represent the local structural features N_i and E_i as $N_i = \sum_{j=1}^{n}\mathbf{A}_{ij}$, $E_i = N_i + \frac{1}{2}\mathbf{A}_{ii}^3$. We then reformulate the attack objective as:

$$\mathbf{A}^* = \arg\min_{\mathbf{A}} \ \tilde{S}_{\mathcal{T}}(\mathbf{A}) \quad (16a)$$

$$= \arg\min_{\mathbf{A}} \sum_{i:i\in\mathcal{T}}(E_i - e^{(1,\ln N_i)^T([\mathbf{1},\ln\mathbf{N}]^T[\mathbf{1},\ln\mathbf{N}])^{-1}[\mathbf{1},\ln\mathbf{N}]^T\ln\mathbf{E}})^2,$$

$$\text{s.t.} \quad N_i = \sum_{j=1}^{n}\mathbf{A}_{ij}, \quad E_i = N_i + \frac{1}{2}\mathbf{A}_{ii}^3; \quad (16b)$$

$$\frac{1}{2}||\mathbf{A}_0-\mathbf{A}||_1 \le B, \quad \mathbf{A}\in\{0,1\}^{n\times n}. \quad (16c)$$

Tackling the above one-level discrete optimization problem results in optimal structural poisoning attacks against the FeXtra GAD system.

B. Attack Methods

From the previous analysis, we successfully reformulate the attacks against GAD systems as a simple one-level optimization question (Eqn. (15) and (16), respectively). However, it is still hard to solve this discrete optimization problem due to the exponential search space. Naturally, we can solve this problem by relaxing the integral constraints to transform the discrete optimization to a continuous one, we then can deploy the vanilla gradient descent approach to tackle this problem. In the sequel, we obtain the optimal solution $\tilde{\mathbf{A}}^*$ and try to map it back to the discrete domain. However, how to map the continuous variable $\tilde{\mathbf{A}}^*$ back to the discrete variable $\mathbf{A}$ is an important challenge that remains to be solved.

To address the above challenge, we first present two conventional methods, GradMaxSearch and ContinuousA, which are adapted from previous works in the literature. Then, we propose a new and more effective method BinarizedAttack.

1) Conventional Methods: Most of the literature related to structural poisoning attacks mainly adopts a vanilla greedy strategy to tackle the discrete optimization problem. To be detailed, the entries in adjacency matrix $\mathbf{A}$ are relaxed to continuous variables, then we can compute the gradient of the attack objective w.r.t each entry. In the sequel, the entry with the largest gradient is chosen to be modified for each iteration because a larger gradient means a higher impact on the attack objective until we reach the budget B. We adapt this greedy way to our scenario and denote it as GradMaxSearch.

It is worth noting that when utilizing the greedy approach, we should especially pay attention to the signs of entry's gradient. For example, if the entry $\mathbf{A}_{ij} = 0$, we should make sure that the corresponding gradient $\frac{\partial AS(v_a)}{\partial \mathbf{A}_{ij}}$ is negative (which is corresponding to add edge) and vice versa. Moreover, it is necessary to add a node pair pool to store the modified node pair to prevent modifying the same entry twice. Additionally, we also include a constraint to avoid the singleton nodes in the graph.

Apparently, an important bottleneck of this greedy approach is that the discrete attack objective is only trained through B steps. Alternatively, we can totally regard the entry in $\mathbf{A}$ as a continuous variable, i.e., $\tilde{\mathbf{A}} \in [0,1]^{N\times N}$ and utilizes the vanilla gradient descent to optimize the attack objective until convergence and obtain the sub-optimal relaxed solution $\tilde{\mathbf{A}}^*$ in the continuous domain. Next, we compute the $L1$ distance between the original adjacency matrix $\mathbf{A}$ and the relaxed solution $\tilde{\mathbf{A}}^*$, and pick out the links with top-B largest distances to manipulate. We denote this vanilla gradient descent method as ContinuousA. Apparently, there exists a large gap between the discrete graph topology $\mathbf{A}$ and its continuous version $\tilde{\mathbf{A}}$. Hence, the optimized $\tilde{\mathbf{A}}^*$ in the continuous space updated by the vanilla gradient descent may be far away from the unreachable best solution $\mathbf{A}^*$ in the discrete space and thus lead to sub-optimal results.

2) Our Proposed BinarizedAttack*:* As previously mentioned, it is natural that GradMaxSearch and ContinuousA have their own weakness. Firstly, the gradient represents a small update on the decision variable. However, we update the entry value with ± 1, which is a number with a large magnitude. Hence this greedy way cannot reach the global optimum. On the other hand, GradMaxSearch can only update the objective up to B steps, which is a strong restriction. For ContinuousA, it totally treats the entry as a continuous variable during training, ignoring the difference between the discrete space and continuous space, thus leading to a sub-optimal solution to the attack objective. Moreover, mapping the continuous solution $\tilde{\mathbf{A}}^*$ back to the discrete variable $\mathbf{A}^*$ may decrease the model performance.

*a) **Idea**:* To bridge this gap, we innovatively propose BinarizedAttack, which is a gradient descent method to optimize the attack objective in iterations. In fact, BinarizedAttack split the forward pass and backward pass in different domains, that is, it computes the attack loss function on the discrete domain to avoid loss distortion, and calculates the gradients in the backward pass and updates the relaxed discrete variable based on the gradients. Then, we specially design a discrete mapping to convert the relaxed variable to a discrete one. At a high level, we associate each node pair $\mathbf{A}_{ij}$ with a discrete decision variable $\mathbf{Z}_{ij} \in \{-1, +1\}$, where $\mathbf{Z}_{ij} = -1$ represents the node pair $\mathbf{A}_{ij}$ should be flipped (from 0 to 1 or vice versa). Let $\mathbf{A}_0$ and $\mathbf{A}$ be the clean and poisoned adjacency matrix, we define:

$$\mathbf{A} = (\mathbf{A}_0 - 0.5 \cdot \mathbf{1}^{N\times N}) \odot \mathbf{Z} + 0.5, \tag{17}$$

where $\odot$ denotes matrices element-wise multiplication. In the sequel, we define a corresponding relaxed decision variable $\dot{\mathbf{Z}} \in [0,1]$ for ease of gradient computation. Then the relationship between $\dot{\mathbf{Z}}$ and $\mathbf{Z}$ is defined as:

$$\mathbf{Z} = -\mathsf{Binarized}(2 \cdot \dot{\mathbf{Z}} - 1), \tag{18}$$

here $\mathsf{Binarized}(\cdot)$ is defined as a step function:

$$\mathsf{Binarized}(x) = \begin{cases} +1, & \text{if } x \geq 0, \\ -1, & \text{if } x < 0, \end{cases} \tag{19}$$

from this perspective, we could rewrite the attack objective $\tilde{S}_{\mathcal{T}}(\mathbf{A})$ based on the two variables $\dot{\mathbf{Z}}$ and $\mathbf{Z}$ as $\tilde{S}_{\mathcal{T}}(\dot{\mathbf{Z}}, \mathbf{Z})$. The above-mentioned discrete mapping process endeavors to mimic the training procedure of the BNN [19] to avoid directly implementing the gradient descent on the discrete decision variables $\mathbf{Z}$. To this end, the accumulation of the gradients of the relaxed continuous variables $\dot{\mathbf{Z}}$ will determine the flip ($+1$ to -1 or vice versa) of the discrete decision variables $\mathbf{Z}$. That is, the gradient information of the continuous $\dot{\mathbf{Z}}$ can serve as an effective search direction to approximate the optimum $\mathbf{Z}^*$ in the discrete space. On the other hand, the updated $\mathbf{Z}$ will be fed into the attack model and update the value of training loss for computing the gradient of $\dot{\mathbf{Z}}$ on the continuous space. Finally, the optimized $\mathbf{Z}^*$ determines which node pairs should be flipped to effectively mitigate the detection performance of the GAD system.

Another important issue is to tackle the discrete budget constraint. To ease optimization, we instead replace this complex budget constraint with a penalty term on the attack objective. As a result, we can optimize the attack objective for more than B steps until convergence. That is, we utilize an L1 penalty [38] based on the relaxed variable $\dot{\mathbf{Z}}$ to replace the

budget constraint. Based on Eqn. (18), it is observed that larger $\dot{\mathbf{Z}}_{ij}$ means $\mathbf{Z}_{ij}$ tend to equal to -1 and the attacker tend to flip the node pair $\mathbf{A}_{ij}$. Hence, the L1 penalty is consistent with restricting the modified times to the clean adjacency matrix. It is worth noting that it will be more appropriate to penalize on the decision variables $\dot{\mathbf{Z}}$ instead of the continuous adjacency matrix $\tilde{\mathbf{A}}$ since penalizing on $\tilde{\mathbf{A}}$ will force the attacker to delete links rather than add links and thus decrease the search space of the model parameters.

*b) **Detailed algorithms**:* Putting all together, it is natural to reformulate the attack problem as an optimization problem based on $\dot{\mathbf{Z}}$ and $\mathbf{Z}$.

Specially, for attacking LGCN, we can reformulate the problem Eqn. (15) as

$$\dot{\mathbf{Z}}^* = \arg\min_{\dot{\mathbf{Z}}} -\mathcal{L}_{\text{R-BCE}}(\mathbf{A}, \mathbf{X}, \mathbf{W}^*, \mathbf{Y}) + \lambda||\dot{\mathbf{Z}}||_1^1 \tag{20a}$$

$$\text{s.t.} \quad \tilde{\mathbf{A}} = \text{diag}(\sum_{i=1}^{n} \mathbf{A}_i)^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\text{diag}(\sum_{i=1}^{n} \mathbf{A}_i)^{-\frac{1}{2}}, \tag{20b}$$

$$\mathbf{W}^* = ((\tilde{\mathbf{A}}^2\mathbf{X})^T\mathbf{D}\tilde{\mathbf{A}}^2\mathbf{X} + \xi\mathbf{I}_{p\times p})^{-1}(\tilde{\mathbf{A}}^2\mathbf{X})^T\mathbf{D}\mathbf{Y}^{train}, \tag{20c}$$

$$\mathbf{A} = (\mathbf{A}_0 - 0.5\cdot\mathbf{1}^{N\times N}) \odot \mathbf{Z} + 0.5, \tag{20d}$$

$$\mathbf{Z} = -\text{Binarized}(2\cdot\dot{\mathbf{Z}} - 1). \tag{20e}$$

It is worth noting that the hyperparameter λ trade-off between the relative impact of the attack objective and the budget constraint and we could solve the Eqn. (20a) via the vanilla projection gradient descent. Similar to the training procedure of the binary weight neural networks, we compute the attack objective based on the discrete variable $\mathbf{Z}$ in the forward pass. Compared to ContinuousA, it will not lead to the attack loss distortion by using the continuous variable for computing the loss. At the same time, BinarizedAttack calculate the gradients w.r.t the relaxed variable $\dot{\mathbf{Z}}$, and then update the discrete variable by Eqn. (20e). In this setting, we can obtain the poisoned graph from Eqn. (17).

The training procedure of the BinarizedAttack against LGCN is presented in Alg. 1. We leave the model comparison between the three attack methods based on the surrogate model in the experiment part.

After finishing the training procedure of the structural poisoning attacks to the LGCN and obtaining the poisoned graph $\mathbf{A}$, we then feed the poisoned graphs with varying attacking powers into the previously mentioned GCN-based GAD systems under the black-box setting: GCN-reweight [2], GAT-reweight [34], FdGars [24], GEM [25], Player2vec [26], GraphSMOTE [27]. The details of the experimental settings and the results are presented in Sec. VI. The intuition here is that we endeavor to design a universal structural poisoning attack against the GAD systems with the GCN as their backbones.

Similarly, for structural poisoning attacks to the unsupervised FeXtra-based GAD systems with the BinarizedAttack

Algorithm 1 BinarizedAttack for LGCN

Input: clean graph $\mathbf{A}^0$, budget B, surrogate model LGCN, training nodal labels $\mathbf{Y}^{train}$, $\Lambda = \{\lambda_k\}_{k=1}^K$, iteration number T, learning rate η.
Parameter: Perturbation $\dot{\mathbf{Z}}$.

```
Pre-train LGCN and get prediction Ŷ^test based on the prediction score: Z*_test = sigmoid(A_0^2 X W*).
Let t = 0 and initialize Ż.
for k ← 1, 2, ..., K do
    while t ≤ T do
        Forward Pass:
            Calculate Z = −binarized(2 · Ż − 1).
            Calculate A = (A_0 − 0.5 · 1^{n×n}) ⊙ Z + 0.5.
            Calculate the symmetric normalized Laplacian Ã = diag(Σ_{i=1}^n A_i)^{−1/2} (A + I) diag(Σ_{i=1}^n A_i)^{−1/2}.
            Obtain RWLS point estimate W* = ((Ã^2 X)^T D Ã^2 X + ξ I_{p×p})^{−1} (Ã^2 X)^T D Y^{train}.
            Obtain goal function L_R-BCE + λ_k ||Ż||_1^1.
        Backward Pass:
            ∀i, j ∈ 1, 2, ..., n, calculate the gradient of the goal function L_R-BCE w.r.t Ż_ij, i.e., ∂L_R-BCE / ∂Ż_ij.
        Projection Gradient Descent:
            Ż → Π_[0,1](Ż − η ∂L_R-BCE / ∂Ż_ij)
    end while
    return Ż
end for
for b ← 1, 2, ..., B do
    Pick out Ż = min L_R-BCE satisfies Σ Z = −b.
    Get poisoned graph A^b = (A_0 − 0.5 · 1^{n×n}) ⊙ Z + 0.5.
    return A^b.
end for
```

method, we can reformulate Eqn. (16) as:

$$\dot{\mathbf{Z}}^* = \arg\min_{\dot{\mathbf{Z}}} \sum_{a=1}^{\tau}(E_a - e^{\rho})^2 + \lambda||\dot{\mathbf{Z}}||_1^1, \tag{21a}$$

$$\text{s.t.} \quad \rho = (1, \ln N_a)^T([1, \ln N]^T[1, \ln N])^{-1}[1, \ln N]^T \ln E; \tag{21b}$$

$$N_i = \sum_{j=1}^{n} \mathbf{A}_{ij}, \quad E_i = N_i + \frac{1}{2}\mathbf{A}_{ii}^3; \tag{21c}$$

$$\mathbf{A} = (\mathbf{A}_0 - 0.5\cdot\mathbf{1}^{N\times N}) \odot \mathbf{Z} + 0.5; \tag{21d}$$

$$\mathbf{Z} = -\text{Binarized}(2\cdot\dot{\mathbf{Z}} - 1). \tag{21e}$$

Then, the above problem can be solved in the same way as in solving Eqn. (20a).

VI. Experiments

A. Datasets

BA (Barabasi-Albert) [39] is a graph generative model which incorporates the preferential attachment method on the probability of the links. we set the number of edges attaching from the target node to the other nodes is 5. Blogcatalog [40] is a social network that describes the connections between the followers and followees in the blog-sharing platform. It

totally has around 88800 nodes with $2.1M$ links. Books [41] is a graph out of the Amazon network, in which we use tags provided by Amazon users. Wikivote [42] has the Wikipedia voting data from the Wikipedia platform until Jan. 2008. The nodes represent the user and links represent "vote for". It has around 7000 nodes with $0.1M$ links. Bitcoin-Alpha [43] is a who-trusts-whom Bictoin-Alpha platform where the traders choose whom to trade the Bitcoin. Bitcoin-Alpha is an unweighted social network with links weights varying from $+10$ to -10, and it has around 3000 nodes with 20000 links. For ease of analysis, we remove the negative links in the Bitcoin-Alpha. Cora [44], Citeseer [45] and Cora-ML [46] are the typical citation networks to curve the citation and co-authors relationships in the academic field. ca-GrQc [47] is a graph that describe the collaboration between the authors' papers and Quantum Cosmology categories from the arXiv e-print platform. For all the real-world networks we choose the largest connected component (LCC) to prevent the singleton nodes and we present the details of the graph data in Tab. I. Citeseer-Wo means to remove the attribute information of the Citeseer dataset.

TABLE I: Statistics of datasets.

Dataset	# Nodes	# Edges	# Attributes	# Anomalies
BA	1000	4975	NA	NA
Wikivote	1012	4860	NA	NA
Bitcoin-Alpha	1025	2311	NA	NA
Cora-ML	2810	7981	NA	NA
Citeseer-Wo	3327	4732	NA	NA
ca-GrQc	5242	14496	NA	NA
Books	1418	3695	21	28
Cora	2708	5803	1433	150
Citeseer	3327	4732	3703	150
Blogcatalog	5196	172652	8189	300

B. Setting

We explore the attack efficacy against the unsupervised FeXtra-based GAD system with the first six datasets. We set the target nodes by random sampling 10 nodes from the top-50 nodes based on the descending order of the anomaly score (AScore). For each trial, we randomly sample the target node set 5 times respectively and provide the mean values of the sum of AScores for the target set under varying attacking powers to quantify the attack effectiveness. We end the attacking procedure until the changes of the mean AScores are saturated.

We explore the structural poisoning attacks against the LGCN with the last three datasets in Tab. I. We synthetically inject structural anomalies into these datasets and flag the perturbed nodes as anomalous. Referring to [48], we randomly sample some nodes in the clean graph and inject cliques to make them fully connected. We randomly split the graph data into training and testing parts, and the ratio of training to testing is $9:1$. We split the datasets 5 times and report the mean AUC scores for model comparison.

C. Attack Performance

1) Effectiveness of attack OddBall*:* We target on exploring the attack effectiveness of the proposed attack methods based on the given target nodes. For convenience, we denote τ_{as} as the decreasing percentage of the mean AScores for evaluation. At the same time, we denote $S_{\mathcal{T}}^0$ and $S_{\mathcal{T}}^B$ be the sum of AScores for the clean and poisoned graphs under the budget B. In the sequel, we define $\tau_{as} = (S_{\mathcal{T}}^0 - S_{\mathcal{T}}^B)/S_{\mathcal{T}}^0$.

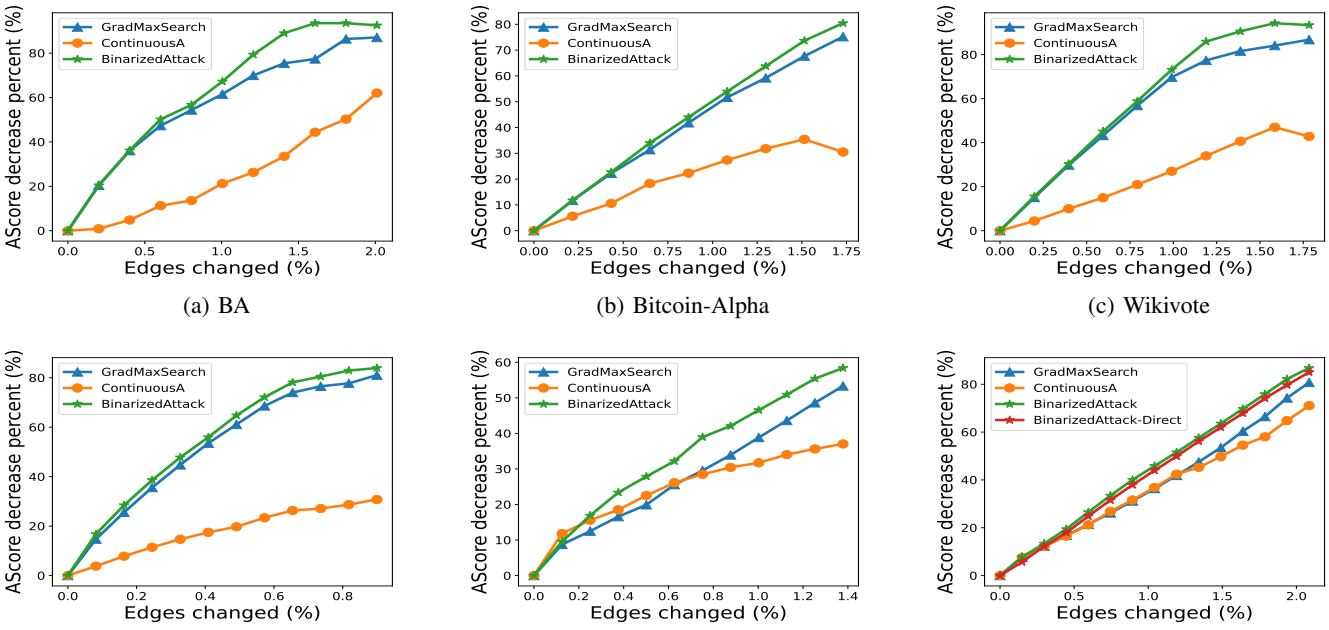
Fig. 3 presents the attack efficacy of BinarizedAttack, GradMaxSearch, and ContinuousA. Specially, we define the attack power as $\frac{B}{|E|}$, where $|E|$ represents the link number to the clean adjacency matrix. It is worth noting that the attacker can only modify a few links to the clean graph, i.e., less than 2% when $|V| = 10$.

It is observed from Fig. 3 that both BinarizedAttack and GradMaxSearch can decrease around up to 90% mean AScores with limited attacking power compared to ContinuousA. This phenomenon demonstrates that training the discrete optimization problem on the continuous domain is not a good choice. In line with expectations, BinarizedAttack outperforms other baselines in all cases. It is noteworthy that the gap between BinarizedAttack and GradMaxSearch is relatively large when the attack power is high. For example, when the attacking power is 0.5%, BinarizedAttack outperforms GradMaxSearch by 41%. Another important thing is that the margin between BinarizedAttack and GradMaxSearch becomes larger as we increase the budget B. It is probable that GradMaxSearch is a greedy approach and cannot find a good solution to a large search space compared to BinarizedAttack.

Unfortunately, the bottleneck of BinarizedAttack is the computation efficiency of the Ego-network features E_i, which includes the computation of $\mathbf{A}^3$. To tackle this problem, we adopt two tricks to increase the efficiency of BinarizedAttack. Firstly, we adopt the sparse matrices operation to replace the dense matrix calculation of the highly sparse matrix $\mathbf{A}$. It is observed that introducing the sparse matrix multiplication will provide around $\times 10$ speed increase on the large dataset ca-GrQc. Secondly, we also observe that BinarizedAttack will almost flip the node pairs within the one-hop Ego-network of the target nodes (the percentage of direct edges is 94.7% for Bitcoin-Alpha). Hence, we can restrict BinarizedAttack to only attacking the direct node pairs to the target nodes and we term it as BinarizedAttack-Direct. This trick significantly decreases the number of parameters during training. However, it will sacrifice the attacking performance to some extent. We deploy BinarizedAttack-Direct on ca-GrQc and find that it can speed up by $\times 2$ times compared with BinarizedAttack with a reasonable sacrifice.

2) Effectiveness of attack LGCN*:* For attacking supervised graph anomaly detection, we focus on analyzing the attack efficacy of three attack methods by measuring the changes in the overall AUC scores after an attack. Fig. 4 presents the mean testing AUC scores of the proposed three attack methods with different attacking powers. Since it is a global attack, we may increase the degree of attack power to up to around 30% to highlight the attacking performance on the supervised graph anomaly detection.

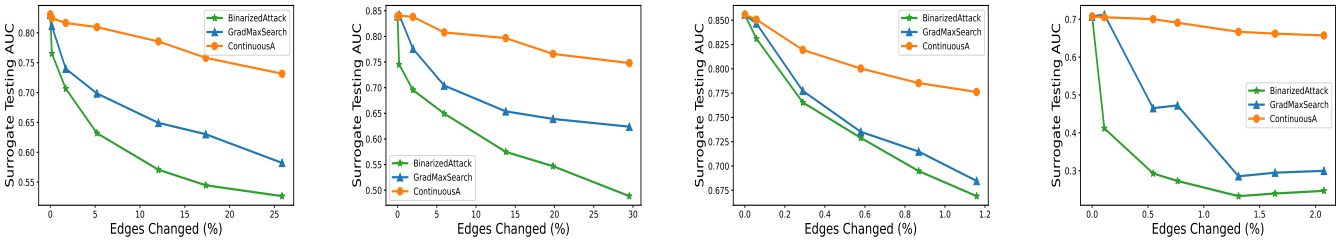
We observe that similar to attacking OddBall, our main method BinarizedAttack consistently outperforms the other two baseline methods in all three real-world datasets. Especially, when the attacking power increase to up to 30%



(a) BA (b) Bitcoin-Alpha (c) Wikivote

(d) Citeseer-Wo (e) Cora-ML (f) ca-GrQc

Fig. 3: Changes in AScores for BA, Bitcoin-Alpha, Wikivote and Citeseer-Wo, Cora-ML and ca-GrQc. (adapted from [7])



(a) Cora (b) Citeseer (c) Blogcatalog (d) Books

Fig. 4: Mean testing AUC scores for Cora, Citeseer, Blogcatalog and Books attribute graphs.

for Citeseer, BinarizedAttack can effectively misguide the LGCN for anomaly node detection (AUC $= 0.5$ means the classification is totally a random guess). At the same time, the AUC margin between BinarizedAttack and GradMaxSearch increase to up to around 0.15 for Citeseer and around 0.08 for Cora.

3) Attacks against GCN-Based GAD Systems: In this section, we explore the black box attacks of the BinarizedAttack against LGCN on other GCN-based GAD models: GCN-reweight, GAT-reweight, FdGars, GEM, Player2vec and GraphSMOTE on Cora and Citeseer dataset as exemplars. We measure the attack efficacy of black-box attacks with the mean AUC scores on the testing data. The results are shown in Tab. II. BR is the modified edges percentage range. We can observe that BinarizedAttack against LGCN has a significant effect on the global accuracy of those six GCN-based GAD systems, especially for FdGars. That is, when the attacking power increases to more than 20%, the mean AUC scores for Cora and Citeseer decrease to 0.48 and 0.52, resulting in an approximately useless anomaly detector. On the other hand, the biggest gap for mean AUC scores before and after an attack is the black box attack against GEM on Cora dataset, which is up to around 32% (decreased from 0.72 to 0.49). In general, the results in Tab. II show that BinarizedAttack against LGCN as a surrogate model can effectively influence the GCN backbones of the other GAD systems, resulting in respectable attack efficacy in black-box attack manner.

D. Ablation Study on WLS

In this section, we start to discuss the contribution of the weighted least square estimation (WLS) in the attack model. If we set the diagonal matrix $\mathbf{D} = \text{Diag}([1, ..., 1]_{n\times 1})$, the point estimate of RWLS in Eqn. (12) changes to Ridge regression [17]:

$$\tilde{\mathbf{A}} = \text{diag}(\sum_{i=1}^{n} \mathbf{A}_i)^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\text{diag}(\sum_{i=1}^{n} \mathbf{A}_i)^{-\frac{1}{2}}, \tag{22a}$$

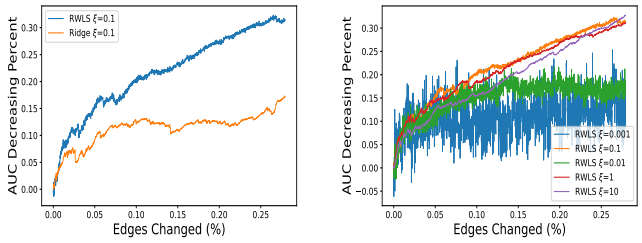
$$\mathbf{W}^* = ((\tilde{\mathbf{A}}^2\mathbf{X})^T\tilde{\mathbf{A}}^2\mathbf{X} + \xi\mathbf{I}_{p\times p})^{-1}(\tilde{\mathbf{A}}^2\mathbf{X})^T\mathbf{Y}. \tag{22b}$$

We compare the decreasing percentage of mean AUC scores for RWLS and Ridge regression on Cora dataset as an ex-

TABLE II: Black-box attacks to GCN-based GAD systems.

BR (%)	GCN-reweight		GAT-reweight		FdGars		GEM		Player2vec		GraphSMOTE		PAMFUL [36]	
	Cora	Citeseer	Cora	Citeseer	Cora	Citeseer	Cora	Citeseer	Cora	Citeseer	Cora	Citeseer	Cora	Citeseer
0	0.72	0.79	0.71	0.79	0.65	0.64	0.72	0.76	0.67	0.64	0.73	0.78	0.72	0.75
$(0, 1]$	0.71	0.76	0.70	0.74	0.60	0.58	0.68	0.75	0.60	0.60	0.73	0.77	0.68	0.72
$(1, 2]$	0.70	0.77	0.72	0.74	0.60	0.60	0.70	0.74	0.61	0.59	0.73	0.78	0.69	0.72
$(2, 5]$	0.68	0.76	0.70	0.72	0.60	0.56	0.70	0.72	0.59	0.55	0.73	0.76	0.67	0.69
$(5, 7]$	0.65	0.75	0.66	0.74	0.59	0.60	0.70	0.72	0.60	0.58	0.72	0.78	0.65	0.68
$(7, 10]$	0.68	0.75	0.68	0.71	0.56	0.56	0.70	0.70	0.58	0.57	0.71	0.77	0.63	0.66
$(10, 15]$	0.60	0.71	0.60	0.69	0.54	0.53	0.57	0.68	0.53	0.52	0.67	0.73	0.62	0.68
$(15, 20]$	0.60	0.64	0.60	0.66	0.52	0.53	0.56	0.66	0.51	0.52	0.66	0.70	0.54	0.68
$(20, 25]$	0.54	0.59	0.56	0.63	0.48	0.52	0.49	0.67	0.51	0.54	0.63	0.66	0.51	0.68

emplar. Fig. 5a illustrates that by using RWLS as the point estimate for LGCN's weights, the attacking performance is significantly higher than Ridge regression (the decreasing percentage of mean AUC scores for RWLS is much larger than Ridge). This phenomenon is attributed to the fact that WLS incorporates the imbalanced sample numbers and pays more attention to the minority class during the training phase, leading to more suitable close-form solutions of W^*. Hence the introduction of RWLS makes a great contribution to the attack model.



(a) Ridge-vs-RWLS (b) Tuning ξ

Fig. 5: (a) RWLS vs Ridge; (b) Sensitivity analysis on ξ.

E. Sensitivity Analysis on ξ

Another important issue is the choice of the penalty parameter ξ for Ridge regression. According to Tab. I, the dimensions of the attributes for real-world graphs are usually more than one thousand, thus leading to a high-dimensional problem in the regression. The introduction of $L2$ penalty here can not only prevent the inexistence of the inverse of the singularity matrix in the point estimate but also highlight the important part of the nodal attributes to prevent over-fitting.

Fig. 5b depicts the effect of choosing ξ with varying order of magnitudes for attack efficacy. The results show that too high and too low value of ξ will damage the attacking performance of the attack model. Especially, if ξ is ranged between $[0.001, 0.01]$, the attacking performance will be unstable. In this context, we regard that $\xi = 0.1$ as a suitable choice for attacking LGCN. Without loss of generality, we choose $\xi = 0.1$ for attacking LGCN in Fig. 4 for the sake of fair comparison. Based on the phenomenon presented in the sensitivity analysis, it is obvious that the choice of ξ can significantly impact the attacking performances of the proposed method. This is due to the fact that the value of ξ will influence the quality of the surrogate model's (LGCN) parameters $\mathbf{W}^*$. An appropriate value of ξ will prevent overfitting and underfitting and hence the surrogate model can perform well on the graph data. As a result, the well-trained surrogate model will provide high-quality gradients to supervise the decision variables $\mathbf{Z}$.

F. Surrogate Model Performances

In this part, we compare the surrogate model's node classification performance with different training methods: point estimate (RWLS) vs gradient descent (GD). We set the learning rate equal to 0.1 and epochs equal to 1000 for gradient descent. Tab. III demonstrates that the point estimate achieves comparable performances with the vanilla gradient descent.

TABLE III: Testing AUC for the surrogate model trained by RWLS and gradient descent.

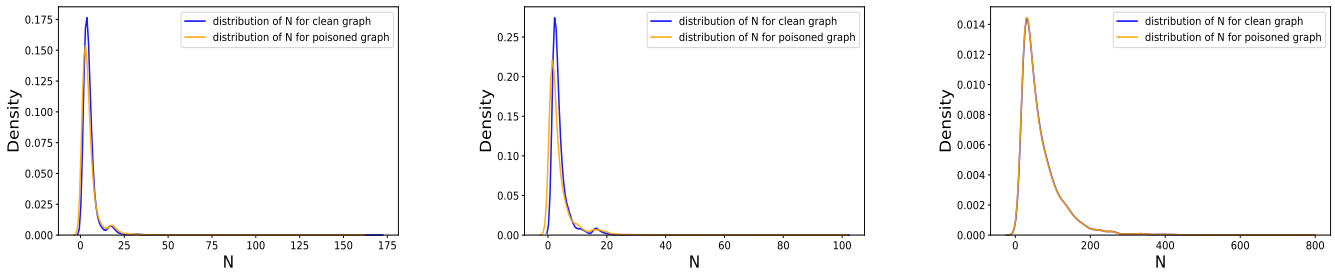
Trial	Cora		Citeseer		BlogCatalog	
	RWLS	GD	RWLS	GD	RWLS	GD
1	0.848	0.918	0.924	0.894	0.762	0.764
2	0.801	0.781	0.878	0.837	0.884	0.854
3	0.903	0.829	0.890	0.860	0.891	0.878
4	0.841	0.838	0.844	0.894	0.830	0.815
5	0.760	0.782	0.698	0.837	0.912	0.897
mean	0.831	0.830	0.847	0.864	0.856	0.842

TABLE IV: p-values for node degree distribution.

p dataset / trial	Cora	Citeseer	BlogCatalog
1	0.181	0.	0.602
2	0.184	0.	0.613
3	0.551	0.	0.601
4	0.206	0.003	0.608
5	0.092	0.002	0.605

G. Side Effect on Node Degree Distribution

We start to analyze whether BinarizedAttack against the GCN-based GAD systems will cause the shift of the node degree distribution as a prominent byproduct. Firstly, we observe the histogram of the node degree distribution of the clean graphs and poisoned graphs in Fig. 6. It shows there exists a slight shift in the node degree distribution qualitatively. We then refer to the permutation test [49] to quantitatively analyze to what extent BinarizedAttack against the GCN-based GAD systems will modify the degree distribution. The permutation



(a) Cora (b) Citeseer (c) BlogCatalog

Fig. 6: Node degree distribution for clean and poisoned graphs.

test is a standard non-parametric test aims at checking whether two groups' data follow the same distribution. However, it is impossible to try for all kinds of permutations due to the exponential search space. To tackle this issue, we can instead use the Monte Carlo approximation to approximate the true p-values with sufficient trials, i.e.,

$$p(t \geq t_0) = \frac{1}{M} \sum_{j=1}^{M} I(t_j \geq t_0), \tag{23}$$

where t_0 is the observed value and $t = |\bar{x'} - \bar{y'}|$ is the resamples statistics and M is the number of trials (we set $M = 10000$ as our setting). x and y can be either $\mathbf{N}^{clean}$ and $\mathbf{N}^{poisoned}$ and y' are re-samples of $\mathsf{Concat}[\mathbf{N}^{clean}||\mathbf{N}^{poisoned}]$. Here $\mathbf{N}$ represents the node degree vector. Tab. IV demonstrates that under 99% confidence level that BinarizedAttack cannot distort the node degree distribution of Cora and BlogCatalog apart from Citeseer. The reason is that Citeseer is much sparser than Cora and BlogCatalog, hence 25% attacking power will significantly change the topology information of Citeseer.

H. Time Cost

Tab. V presents the time required for each epoch of the training phase of various attack methods on the four graph datasets. It is evident that our proposed method achieves the minimum time cost compared to other baselines.

TABLE V: Time cost (s) for various methods on four datasets.

time \ dataset / method	Cora	Citeseer	BlogCatalog	Books
GradMaxSearch	0.68	1.29	5.12	0.20
ContinuousA	0.11	0.20	0.68	0.03
BinarizedAttack	0.06	0.15	0.58	0.02

VII. Conclusion

Due to the exploding development of graph learning, graph-based anomaly detection is becoming an important topic in the machine learning field. However, similar to other machine learning models, it is vulnerable to adversarial attacks. In this paper, we focus on OddBall as a feature-extraction-based anomaly detector and LGCN as the surrogate model of the GCN-based anomaly detector, we further mathematically formulate it as a discrete one-level optimization problem by utilizing the specially designed regression methods (OLS estimation for OddBall and RWLS estimation for LGCN). For ease of optimization, we propose BinarizedAttack to mimic the training of binary weight neural network which can effectively degenerate the GAD systems. The comprehensive experiments demonstrate that our method consistently outperforms other baselines. Moreover, we explore the black-box attacks of the typical six GCN-based GAD systems (GCN-reweight, GAT-reweight, FdGars, GEM, Player2vec and GraphSMOTE) and prove that by universally attacking LGCN can effectively attack other GCN-based GAD systems under a black-box scenario. In the future, we endeavor to study the defense method against structural attacks to enhance the robustness of GAD systems.

References

[1] F. E. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.

[2] T. N. Kipf and M. Welling, "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, 2017.

[3] A. Diadiushkin, K. Sandkuhl, and A. Maiatin, "Fraud detection in payments transactions: Overview of existing approaches and usage for instant payments," *Complex Systems Informatics and Modeling Quarterly*, pp. 72–88, 10 2019.

[4] S. Khaled, N. El-Tazi, and H. M. O. Mokhtar, "Detecting fake accounts on social media," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 3672–3681, 2018.

[5] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, "Aiding the detection of fake accounts in large scale social online services," in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, (San Jose, CA), pp. 197–210, USENIX Association, Apr. 2012.

[6] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146, 2003.

[7] Y. Zhu, Y. Lai, K. Zhao, X. Luo, M. Yuan, J. Ren, and K. Zhou, "Binarizedattack: Structural poisoning attacks to graph-based anomaly detection," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pp. 14–26, 2022.

[8] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 420–429, 2007.

[9] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection: a realistic modeling and a novel learning strategy," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 8, pp. 3784–3797, 2017.

[10] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The World Wide Web Conference*, WWW '19, (New York, NY, USA), p. 417–426, Association for Computing Machinery, 2019.

[11] M. Iwamoto, S. Oshima, and T. Nakashima, "A study of malicious pdf detection technique," in *2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, pp. 197–203, 2016.

[12] L. Akoglu, M. McGlohon, and C. Faloutsos, "oddball: Spotting anomalies in weighted graphs," in *Advances in Knowledge Discovery and Data Mining* (M. J. Zaki, J. X. Yu, B. Ravindran, and V. Pudi, eds.), (Berlin, Heidelberg), pp. 410–421, Springer Berlin Heidelberg, 2010.

[13] R. Liu, J. Gao, J. Zhang, D. Meng, and Z. Lin, "Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 10045–10067, 2022.

[14] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," *SIGKDD*, Jul 2018.

[15] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *International Conference on Learning Representations*, 2019.

[16] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning*, pp. 6861–6871, PMLR, 2019.

[17] D. E. Hilt, D. W. Seegrist, U. S. F. Service., and P. Northeastern Forest Experiment Station (Radnor, *Ridge, a computer program for calculating ridge regression estimates*, vol. no.236. Upper Darby, Pa, Dept. of Agriculture, Forest Service, Northeastern Forest Experiment Station, 1977. https://www.biodiversitylibrary.org/bibliography/68934.

[18] *Weighted Least-Squares Method*, pp. 566–569. New York, NY: Springer New York, 2008.

[19] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 4114–4122, 2016.

[20] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data mining and knowledge discovery*, vol. 29, no. 3, pp. 626–688, 2015.

[21] H.-H. Chen and C. L. Giles, "Ascos: an asymmetric network structure context similarity measure," in *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, pp. 442–449, IEEE, 2013.

[22] D. Kuang, C. Ding, and H. Park, "Symmetric nonnegative matrix factorization for graph clustering," in *Proceedings of the 2012 SIAM international conference on data mining*, pp. 106–117, SIAM, 2012.

[23] U. Kang, D. H. Chau, and C. Faloutsos, "Mining large graphs: Algorithms, inference, and discoveries," in *2011 IEEE 27th International Conference on Data Engineering*, pp. 243–254, IEEE, 2011.

[24] J. Wang, R. Wen, C. Wu, Y. Huang, and J. Xion, "Fdgars: Fraudster detection via graph convolutional networks in online app review system," in *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, (New York, NY, USA), p. 310–316, Association for Computing Machinery, 2019.

[25] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song, "Heterogeneous graph neural networks for malicious account detection," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 2077–2085, 2018.

[26] Y. Zhang, Y. Fan, Y. Ye, L. Zhao, and C. Shi, "Key player identification in underground forums over attributed heterogeneous information network embedding framework," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, (New York, NY, USA), p. 549–558, Association for Computing Machinery, 2019.

[27] T. Zhao, X. Zhang, and S. Wang, "Graphsmote: Imbalanced node classification on graphs with graph neural networks," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, (New York, NY, USA), p. 833–841, Association for Computing Machinery, 2021.

[28] J. Tang, J. Li, Z. Gao, and J. Li, "Rethinking graph neural networks for anomaly detection," in *International Conference on Machine Learning*, 2022.

[29] K. Zhou and Y. Vorobeychik, "Robust collective classification against structural attacks," in *Conference on Uncertainty in Artificial Intelligence*, pp. 250–259, PMLR, 2020.

[30] K. Zhou, T. P. Michalak, M. Waniek, T. Rahwan, and Y. Vorobeychik, "Attacking similarity-based link prediction in social networks," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 305–313, 2019.

[31] M. Waniek, K. Zhou, Y. Vorobeychik, E. Moro, T. P. Michalak, and T. Rahwan, "How to hide one's relationships from link prediction algorithms," *Scientific reports*, vol. 9, no. 1, pp. 1–10, 2019.

[32] M. Waniek, T. P. Michalak, M. J. Wooldridge, and T. Rahwan, "Hiding individuals and communities in a social network," *Nature Human Behaviour*, vol. 2, no. 2, pp. 139–147, 2018.

[33] B. Zdaniuk, *Ordinary Least-Squares (OLS) Model*, pp. 4515–4517. Dordrecht: Springer Netherlands, 2014.

[34] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[35] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[36] T. Zhao, T. Jiang, N. Shah, and M. Jiang, "A synergistic approach for graph anomaly detection with pattern mining and feature learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 6, pp. 2393–2405, 2022.

[37] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[38] R. Tibshirani, "Regression shrinkage and selection via the lasso," *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, vol. 58, pp. 267–288, 1994.

[39] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Rev. Mod. Phys.*, vol. 74, pp. 47–97, Jan 2002.

[40] R. Zafarani and H. Liu, "Users joining multiple sites: Distributions and patterns," 2014.

[41] K. Liu, Y. Dou, Y. Zhao, X. Ding, X. Hu, R. Zhang, K. Ding, C. Chen, H. Peng, K. Shu, G. H. Chen, Z. Jia, and P. S. Yu, "Pygod: A python library for graph outlier detection," 2022.

[42] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *Proceedings of the 19th international conference on World wide web*, pp. 641–650, 2010.

[43] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. Subrahmanian, "Rev2: Fraudulent user prediction in rating platforms," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 333–341, ACM, 2018.

[44] Y. Liu, Z. Li, S. Pan, C. Gong, C. Zhou, and G. Karypis, "Anomaly detection on attributed networks via contrastive self-supervised learning," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.

[45] C. Giles, K. Bollacker, and S. Lawrence, "Citeseer: an automatic citation indexing system," in *Proceedings of the ACM International Conference on Digital Libraries*, pp. 89–98, ACM, 1998. Proceedings of the 1998 3rd ACM Conference on Digital Libraries ; Conference date: 23-06-1998 Through 26-06-1998.

[46] A. Bojchevski and S. Günnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," in *International Conference on Learning Representations*, 2018.

[47] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 2–es, 2007.

[48] D. B. Skillicorn, "Detecting anomalies in graphs," in *2007 IEEE Intelligence and Security Informatics*, pp. 209–216, IEEE, 2007.

[49] L. Stanberry, *Permutation Test*, pp. 1678–1678. New York, NY: Springer New York, 2013.