CGP: Centroid-guided Graph Poisoning for Link Inference Attacks in Graph Neural Networks

Haozhe Tian, Haibo Hu, Qingqing Ye
Department of Electrical and Electronic Engineering
The Hong Kong Polytechnic University
Hong Kong SAR, China
haozhe_tian@outlook.com, {haibo.hu, qqing.ye}@polyu.edu.hk

***Abstract*—Graph Neural Network (GNN) is the state-of-the-art machine learning model on graph data, which many modern big data applications rely on. However, GNN's potential leakage of sensitive graph node relationships (i.e., links) could cause severe user privacy infringements. An attacker might infer the sensitive graph links from the posteriors of a GNN. Such attacks are named graph link inference attacks. While most existing research considers attack settings without malicious users, this work considers the setting where some malicious nodes are established by the attacker. This setting enables link inference without relying on the estimation of the number of links in the target graph, which significantly enhances the practicality of link inference attacks. This work further proposes centroid-guided graph poisoning (CGP). Without participating in the training process of the target model, CGP operates on links between malicious nodes to make the target model more vulnerable to graph link inference attacks. Experiment results in this work demonstrate that using less than 5% of malicious nodes, i.e. modifying approximately 0.25% of all links, CGP can increase the F-1 of graph link inference attacks by up to 4%.**

***Index Terms*—Centroid-guided graph poisoning, graph link inference attacks, graph neural networks**

I. Introduction

Graph data is drawing increased attention in big data applications because it effectively portrays the connections and dependencies among various entities. Some examples of graph data are the citation networks of published scientific papers [1], [2] and the hyperlinks between weblogs [3]. To learn the representations of relationships within graph data, graph neural networks (GNNs), which reflect graph relationships through node embeddings, were developed [4].

Although GNN achieved ground-breaking results and became the de facto technique in many graph-related applications [5]–[7], its privacy risks were not well investigated.

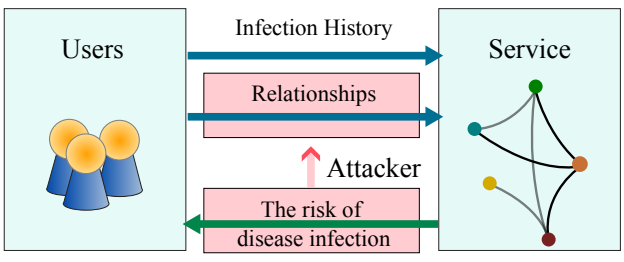

Fig. 1. Example of GNN leaking user relationships in infection prediction.

In particular, attackers may infer sensitive user relationships through graph link inference. Illustrated as an example in Fig. 1, to predict disease infection, users contribute to a server their infection histories, which act as node features, and their social relationships/interactions, which form a social graph. The server builds a GNN based on the social graph and predicts the likelihood of each user getting the disease in the future. A malicious third party (i.e., the attacker) can use the feedback from the server (i.e., the node posteriors) to infer user relationships (i.e., the graph links). Such attacks, namely **graph link inference attacks**, can pose great privacy threats to users since the links in graphs usually contain sensitive information, such as social relationships [8] or mobility patterns of people [9].

Existing works consider link inference attacks without malicious users. It is assumed that the third-party attackers can only access GNN posteriors and are not involved in the training of the target GNN. These works commonly rely on the distance between nodes' posteriors in the feature space. Intuitively, the posteriors of linked nodes are closer in the features space, whereas the posteriors of un-linked nodes are further in the features space [10], [11]. Based on this assumption, [12] used a simple distance-based k-closest principle, which predicts links between the k closest node posterior pairs. However, the k-closest method has three limitations. First, it is sensitive to the selection of distance functions [12]. Second, it is sensitive to the choice of k. Third, the distances between all node posterior pairs need to be considered before making link predictions.

The sensitivity to k of the k-closest method is especially problematic for real-life applications of graph link inference attacks. For the k-closest method, only k positive predictions are made, therefore k reflects a rough estimation of the number of links in the target graph. However, estimating k without knowing the graph structure or graph meta-data (such as the average node degree) is very challenging. Results from previous works showed that small changes in k led to big performance changes, even rendering the predictions ineffective [12], [13].

This work takes a more proactive perspective for the attackers, where a small percentage of malicious nodes ($\leq 5\%$) are established by the attacker. This is a practical real-life attack setting [14]. In the infection prediction example in Fig. 1, the attacker, even as a third party, can create a small percentage

H. Tian, H. Hu and Q. Ye, "CGP: Centroid-guided Graph Poisoning for Link Inference Attacks in Graph Neural Networks," Proceedings of the IEEE International Conference on Big Data (BigData), 2023, pp. 554-561, doi: 10.1109/BigData59044.2023.10386501.

of "fake" users in the system. As shown in this work, once malicious nodes are available, graph link inference attacks can be enhanced from two aspects: **(a) graph poisoning** and **(b) supervised attack**.

For **(a) graph poisoning**, by operating links between the malicious nodes, this work proposes to poison the target GNN model to make it more vulnerable to graph link inference attacks. A method called centroid-guided graph poisoning (CGP) is proposed, which has the following three properties: 1) It relies only on less than 5% of malicious nodes, i.e. approximately 0.25% of all links, 2) it works offline without needing to participate in the training of the target GNN model, and 3) despite only changing the links between malicious nodes, it affects the inherent structure of the whole graph and the clustering of all node posteriors.

For **(b) supervised attack**, the links between malicious nodes form a "shadow dataset" that can be used to train a supervised graph link inference attack model. By transforming the attack into a supervised binary classification problem, we can address the three problems of k-closest link inference attacks mentioned above.

The findings in this work draw attention to the threat to GNN privacy posed by malicious users. The introduction of a small percentage of malicious users greatly enhances the practicality of graph link inference attacks. To summarize, the main contributions of this paper are as follows.

- This work characterizes two potential graph link inference attack settings: without and with malicious users.
- This work explores how the existence of malicious users potentially affects the paradigm of graph link inference attacks.
- This work further exploits the attack setting with malicious users by proposing CGP, which increased the F-1 of link inference by up to 4%.

This paper is organized as follows: Section II introduces the link inference attack setting without malicious users. The corresponding attack method (i.e., the k-closest method) and the challenges it faces are explained. Section III explains the more proactive link inference attack setting, where malicious users exist. The malicious users enable (a) graph poisoning and (b) supervised attack, which solve the challenges in the k-closest method. Section IV compared graph link inference attacks without and with malicious users. Section V concludes the risk of leaking sensitive links posed by malicious users.

II. Background

A. Preliminaries

Graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes a set of N nodes, $\mathcal{E} \subseteq N \times N$ represents the links between nodes in $\mathcal{V}$. The links in $\mathcal{E}$ can be completely characterized by the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$. For undirected graphs, if link exists between node i and node j, then $\mathbf{A}_{i,j} = \mathbf{A}_{j,i} = 1$. Otherwise, $\mathbf{A}_{i,j} = \mathbf{A}_{j,i} = 0$.

Graph Convolutional Network (GCN) was proposed by Thomas Kipf and Max Welling [10]. It is currently one of the most prominent variants of GNN [15]. A GCN layer follows the following propagation rule:

$$\mathbf{H} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{\frac{1}{2}}\mathbf{X}\mathbf{W}) \tag{1}$$

Here, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$. $\mathbf{A}$ is the adjacency matrix of graph $\mathcal{G}$. $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$. $\mathbf{X} \in \mathbb{R}^{N \times F}$ is the input feature matrix for N users each with F node features. $\mathbf{W} \in \mathbb{R}^{F \times E}$ is the trainable parameters of the model. $\mathbf{H} \in \mathbb{R}^{N \times E}$ is the node posteriors. $\sigma(\cdot)$ is a nonlinear activation. A GNN containing GCN layers can have multiple GCN layers. However, in most cases, 1 or 2 GCN layers are utilized and extra layers might cause a performance decrease [16]. In this work, we focus on graph inference attacks on GCN.

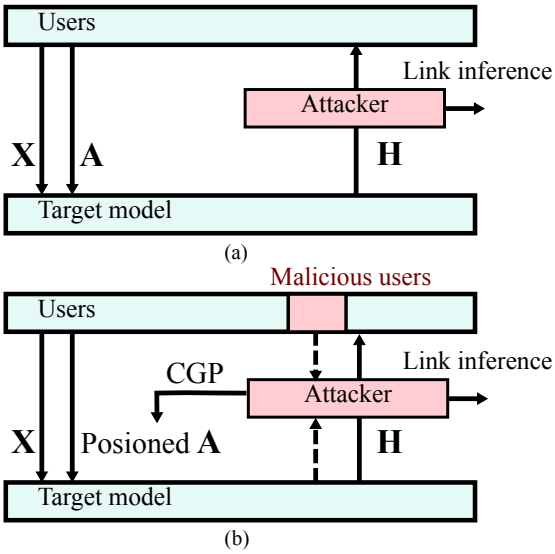

Fig. 2. (a). Graph link inference attack without malicious users. (b). Graph link inference attack with malicious users. The attacker makes use of malicious users to poison the target model and increase link inference accuracy.

Graph Link Inference Attack targets sensitive links in the graph collected by the GNN service provider. On the service provider (i.e. the target model) side, the target graph $\mathcal{G}$ and node features $\mathbf{X}$ of users are collected and used to train a GNN, which outputs node posteriors $\mathbf{H}$. The objective of the graph link inference attack is to infer the $\mathcal{E}$ based on $\mathbf{H}$.

B. Graph Link Inference Attack without Malicious Nodes

As mentioned in Section I, the k-closest method has been thoroughly discussed in the literature [12]. Nevertheless, this section introduces the settings and methodologies of the k-closest method to facilitate the illustration of the graph link inference attack problem. This technique also acts as a baseline in the scope of this work.

The attack setting without malicious nodes (Fig. 2 (a)) is the strictest setting, where the attacker only has access to the node posteriors of the target GNN model. Neither model parameters, structures, nor gradients are available to the attacker. While attempting to infer the edges, the attacker works offline, i.e. without participating in the training of the target model.

In this setting, the commonly used attack method is the k-closest method, which works in an unsupervised manner. The k-closest method is based on the intuition that nodes with similar posteriors/predictions are more likely to be connected [12]. The "similar" here means distance in the feature space. Let the node posteriors, i.e. the attacker's information, be $\mathbf{H} \in \mathbb{R}^{N \times E}$, where N is the number of users, E is the size of node posteriors. The i-th row of $\mathbf{H}$, $\mathbf{h}_i \in \mathbb{R}^E$, is the node posterior of node i. The k-closest method starts by constructing a set of node posterior pairs as illustrated below:

$$\mathcal{P} = \{(\mathbf{h}_0, \mathbf{h}_1), (\mathbf{h}_0, \mathbf{h}_2), \ldots, (\mathbf{h}_0, \mathbf{h}_{N-1}), (\mathbf{h}_1, \mathbf{h}_2), \ldots\}. \tag{2}$$

For N users, there are $N(N-1)/2$ node posterior pairs in $\mathcal{P}$. For each node posterior pair, a distance is calculated as:

$$d_{i,j} = d_{j,i} = f(\mathbf{h}_i, \mathbf{h}_j), \tag{3}$$

where $f(\cdot)$ is the distance function. In this work, 8 distance functions: *Cosine, Euclidean, Sqeuclidean, Correlation, Chebyshev, Braycurtis, Canberra*, and *Cityblock* are considered. Since this work considers undirected graphs, all these distance functions are symmetric, i.e. $f(\mathbf{h}_i, \mathbf{h}_j) = f(\mathbf{h}_j, \mathbf{h}_i)$. After obtaining the distance between node posterior pairs, the pairs in $\mathcal{P}$ are ranked accordingly, and the k closest node posterior pairs are considered connected.

C. The Challenges of the k-closest Method

Although the k-closest method can produce valid inferences on links in the target graph, the application of the method in real-life attack settings faces the following three challenges.

1) It is sensitive to the selection of distance function $f(\cdot)$.
2) It is subjected to reliable estimation of k.
3) It needs to consider all $N(N-1)/2$ node posterior pairs before making inference.

To illustrate challenge 1), we conducted experiments using all 8 distance functions mentioned in Section II-B on three datasets. The details of the datasets are given in Section IV and the results are shown in Fig. 3. The metric used for evaluation is the Area Under the Receiver Operating Characteristic Curve (AUC). We observed that the k-closest method based on different distance functions showed different AUC values. For each dataset, the best-performing distance function was different. Without interacting with the target graph, the attacker cannot determine a distance function that performs well universally.

Regarding challenge 2), the selection of k, i.e. the number of predicted links, directly affects the performance of the k-closest method. If k is smaller than the ideal value, few links are predicted. If it is larger than the ideal value, too many false positives will render the predictions meaningless. However, since the attacker does not have such information on the graph, it is hard to obtain a reliable estimation of k. As shown in the ROC curves in Fig. 6, different choices of k, reflected by the thresholds used to plot the ROC curve, resulted in very different performances. [17] proposed three methods for estimating k, but these methods are based on the attacker knowing partial graphs. [12] proposed a method for unsupervised estimation of k based on clustering, but the method still has relatively low precision (< 0.79).

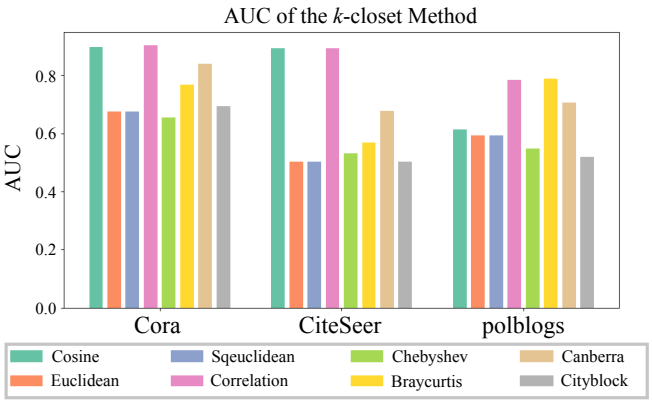

Fig. 3. The effect of distance function on the k-closest method. AUC was used as the metric for evaluation.

For challenge 3), since the k-closest method functions by finding node posterior pairs that are relatively close, distances need to be computed for all $N(N-1)/2$ pairs before making predictions. This complexity might be problematic when dealing with larger graphs.

III. Methodology

Although the k-closest method is capable of making valid deductions of graph links, the three aforementioned problems, namely: sensitivity to distance function $f(\cdot)$, requirement of pre-defined k, and the inability to make real-time predictions, restrict its applications in real-life attacks. With the intuition and methodologies of the k-closest method in mind, this section proposes a more proactive attack with malicious users to increase the practicality of graph link inference attacks.

A. Attack Setting

With everything else similar to Section II-B, we consider the attacker establishes a small percentage of malicious nodes. For example, 5% malicious nodes, involving approximately 0.25% of links. This is equivalent to knowing a small sub-graph of $\mathcal{G}$. Formally, the attacker knows:

- $\mathbf{H}$: the node posterior matrix.
- $v_{m_i,m_j} \in \mathcal{V}$: the links between malicious nodes with indexes $\mathcal{M} = \{m_1, m_2, \ldots, m_n\}, n \leq 0.05N$.

The knowledge of the sub-graph can be used to poison the target model and train a supervised attack model, which addresses the three problems of the k-closest method mentioned in Section II-C and enhances the practicality of link inference attack. With malicious nodes, the attack setting and workflow are illustrated in Fig. 2 (b).

It is worth noticing that this attack setting is realistic. Consider the example in Section I Fig. 1, the attacker might bribe some users to reveal their connectivity with others. The attacker might also forge some malicious users over whom she has full control. It is also feasible to observe some users' connections in a recommender system [15]. It is worth noticing

that this work only considers the links between the malicious nodes because the disclosure of a link between two nodes requires consensus from both parties, which is unattainable from non-malicious users.

In the rest of this section, we exploit these malicious nodes in two ways: *Graph Poisoning* and *Supervised Attack*.

B. Graph Poisoning

The main contribution of this work is to propose a graph poisoning method named CGP. With the objective of making the target model vulnerable to graph link inference attacks, CGP poisons the target graph by altering the links between malicious nodes. Since the attacker is not involved in the training and validation of the target model, CGP works in an offline manner, i.e. the final version of the target model is trained after the graph poisoning. CGP relies only on a small percentage of malicious nodes, making it practical in realistic attack settings. Nevertheless, since the malicious nodes are connected to other links in the target graph, CGP affects the topology of the whole graph.

As information flows through graph links, GNN achieves a clustering effect to some degree [18]. In general, after the transformation by a GNN layer, the posteriors of linked nodes are closer than those nodes that are not linked. Observing the methodologies of the k-closest method, it is speculated that the performance of graph link inference attacks highly depends on how well node posteriors cluster in the feature space. If connected nodes cluster better in the feature space, the performance of distance-based link inference will likely improve. Based on this intuition, to achieve the goal of improving the graph link inference attack, CGP aims to amplify the clustering effect of the target GNN model.

Consider an alteration of a link between two malicious nodes. The objective of CGP can be summarized into the following optimization with constraints:

$$\begin{aligned} \mathbf{A}' = \arg\min_{\mathbf{A}} \; & \mathcal{L}(\mathbf{A}, \mathbf{W}, \mathbf{X}) \\ \text{s.t. } & \mathbf{c}' = \mathbf{c} \\ & \|\mathbf{A}' - \mathbf{A}\|_0 = 2 \\ & W = \arg\min_{\mathbf{W}} \mathcal{L}(\mathbf{A}, \mathbf{W}, \mathbf{X}) \end{aligned}, \tag{4}$$

in which $\mathbf{A}'$ is the adjacency matrix after altering the link between the two malicious nodes. $\mathcal{L}(\cdot)$ is the loss function, $\mathbf{c}$ is the original prediction of the target model, and $\mathbf{c}'$ is the prediction of the target model after altering a link. The constrained optimization in (4) only involves one edge and works on a well-trained target model with parameter $\mathbf{W}$.

However, solving the optimization problem in (4) remains challenging. due to the following two reasons. 1) the graphs are discrete in nature and 2) the feedback from the target model is not accessible to the attacker. Due to these challenges, gradient methods cannot be used. There have been works that adversely affect the performance of GNN by altering the graph structure, which was the "arg max" version of the optimization problem in (4). The work in [19] proposed to derive the graph perturbation by using a linearized surrogate model based on GCN. The work in [20] achieves offline optimization by deriving the relationship between DeepWalk [11] and Singular Value Decomposition (SVD).

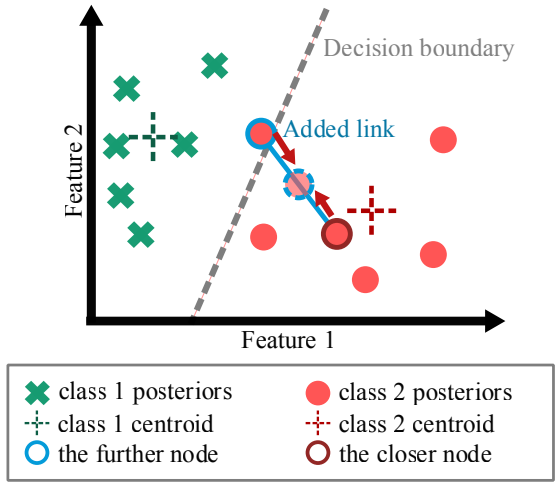

Fig. 4. The schematic of CGP. The plot shows the posteriors of nodes in the feature space. For nodes of the same class, connecting a node that is far from the class centroid with a node that is close to the class centroid changes the decision boundary and makes the cluster more compact.

CGP views the constrained optimization in (4) from a clustering point of view. The schematic in Fig. 4 illustrates the intuition behind CGP. For conciseness, let us consider the binary classification of graph nodes. After the transformation of GNN, connected nodes that belong to the same class form a cluster. On the schematic, one red dot goes beyond the decision boundary, which will likely cause wrong predictions when the attacker performs graph link inference attacks. CGP improves the clustering of node posteriors and the decision boundaries by adding a link between the node far from the class centroid (the further node) and the node close to the class centroid (the closer node). Indeed, the closer node will be moved further from the class centroid. However, the further away node, which is more significant in the determination of the decision boundary, is moved closer to the class centroid.

We now illustrate the intuition of CGP mathematically. In a simplified setting, where there are only two nodes i and j ($i \neq j$). According to the propagation rule of GCN layer in (1), the posterior $\mathbf{h}_i$ of node i can be written as:

$$\begin{aligned} \mathbf{h}_i &= \sum_{\substack{j=0 \\ \mathbf{A}_{i,j} \neq 0}}^{N-1} \frac{\mathbf{A}_{i,j}}{\sqrt{d_i d_j}} X_j \\ &= \begin{cases} \mathbf{x}_i & , \mathbf{A}_{i,j} = \mathbf{A}_{j,i} = 0 \\ (\mathbf{x}_i + \mathbf{x}_j)/2 & , \mathbf{A}_{i,j} = \mathbf{A}_{j,i} = 1 \end{cases} \end{aligned} \tag{5}$$

Since CGP operates on well-optimized $\mathbf{W}$, we assume constant $\mathbf{W}$ before and after the operation on $\mathbf{A}$, therefore $\mathbf{W}$ in (1) is omitted. Here $\mathbf{x}_i$ and $\mathbf{x}_j$ are the node features of node i and j. Take the Euclidean distance as an example. Assume nodes i and j are not connected originally. The

original distance between $\mathbf{h}_i$ and $\mathbf{h}_j$ is calculated as:

$$f(\mathbf{h}_i, \mathbf{h}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2 \tag{6}$$

After linking node i and j, the distance between the new posterior $\mathbf{h}_i$ and the original posterior $\mathbf{h}_j$ becomes:

$$f(\mathbf{h}_i, \mathbf{h}_j) = \|(\mathbf{x}_i + \mathbf{x}_j)/2 - \mathbf{x}_j\|_2 = \|\mathbf{x}_i - \mathbf{x}_j\|_2/2 \tag{7}$$

Equations (6) and (7) show that by connecting node i (a node close to the decision boundary) with node j (a node close to the class centroid but far away from the decision boundary), h_i' moves to the direction of h_j, which is closer to the class centroid. This potentially improves the clustering effect of the target model, thus minimizing $\mathcal{L}(\mathbf{A}, \mathbf{W}, \mathbf{X})$ in (4).

Apart from adding links, another intuitive way to improve the clustering effect of node posteriors is to remove links between malicious nodes from different clusters. In CGP, the links between each malicious node and other malicious nodes belonging to different clusters are removed. This link removal brings another benefit. Since both the addition and the removal of links are used, it is possible to maintain a relatively stable node degree. This allows for future work on countering malicious node detection.

The detailed algorithm is given in **Algorithm 1**. The attacker knows the node posteriors $\mathbf{h}_i, i = 1, 2, \ldots, N$. CGP begins by clustering all h_i into C clusters using the k-means clustering. All node posteriors belonging to the cluster c are averaged to obtain cluster centroid $\mathbf{o}_c$ and assigned class c.

Algorithm 1: CGP

input : Node posteriors $\mathbf{H}$, number of node categories C, indices of malicious nodes $\mathcal{M}$, adjacency matrix $\mathbf{A}$, threshold d

output: Altered adjacency matrix $\mathbf{A}'$

C-Clustering to find node posterior centroids $\{\mathbf{o}_1, \mathbf{o}_2, \ldots \mathbf{o}_C\}$;
$\mathbf{A}' \leftarrow \mathbf{A}$;
for *node* $m \in \mathcal{M}$ **do**
  $c \leftarrow \arg\min_c \|\mathbf{h}_m - \mathbf{o}_c\|_2$;
  $\mathcal{M}^{same} \leftarrow$ nodes most approximate to o_c;
  $\mathcal{M}^{diff} \leftarrow \mathcal{M} \backslash \mathcal{M}^{same}$;
  for *node* $n \in \mathcal{M} \backslash m$ **do**
    if *node* $n \in \mathcal{M}^{same}$ ***and*** $\|\mathbf{h}_n - \mathbf{o}_c\|_2 < \min(\|\mathbf{h}_m - \mathbf{o}_c\|_2, d)$ **then**
      $\mathbf{A}'_{m,n} \leftarrow 1$;
      $\mathbf{A}'_{n,m} \leftarrow 1$;
    if *node* $m \in \mathcal{M}^{diff}$ **then**
      $\mathbf{A}'_{m,n} \leftarrow 0$;
      $\mathbf{A}'_{n,m} \leftarrow 0$;

CGP then iterates over all malicious nodes. In each iteration, the malicious node m belonging to class c is considered. Malicious nodes belonging to the same class have indexes $\in \mathcal{M}^{same}$, whereas others have indexes $\in \mathcal{M}^{diff}$. For malicious node n ($n \neq m$), if $n \in \mathcal{M}^{same}$, and $\mathbf{h}_n$ is closer to the class centroid $\mathbf{o}_c$ than $\mathbf{h}_m$, a link is added between nodes m and n. If $n \in \mathcal{M}^{diff}$, the link is removed.

It is worth noticing that while the discussion in this section focuses on one malicious node posterior pair, CGP can affect the whole graph. Although the links between the majority of nodes remain inaccessible to the attacker, they are likely to be linked to some of the malicious nodes. By moving one malicious node closer to its class centroid, the "hyper-node" formed by itself and other benign nodes move as well.

C. Supervised Attack

The information from the malicious nodes forms a dataset $\mathcal{D} = \{((\mathbf{h}_{m_i}, \mathbf{h}_{m_j}), v_{m_i,m_j})\},\ m_i, m_j \in \mathcal{M}$ that can be used to train supervised attack models. To overcome the k-closest method's sensitivity to distance functions, the model first calculates all 8 distance functions (Section II-B) of the pair $(\mathbf{h}_{m_i}, \mathbf{h}_{m_j})$. These metrics go through a multi-layer perception to make predictions $\mathbf{p}$.

This attack model structure solves the three problems of the k-closest method mentioned in Section II-C as follows: First, the model structure takes into account several distance functions, therefore alleviating sensitivity to distance functions. Second, the supervised nature of the attack model means it does not rely on an estimated k of the target graph. Third, after training, the supervised network makes real-time inferences, without the need to consider $N(N-1)/2$ node pairs.

IV. Experimentation

A. Configuration

The dataset used for evaluation were Cora [21], CiteSeer [2], and polblogs [3]. The target GNN model was set to have a (node features size×8) GCN layer with ReLU activation and an (8×C) linear layer. To train the target model, the cross-entropy loss and the Adam optimizer [22] were used. The learning rate was fixed at 0.02, and all the models were trained for 50 epochs.

To justify the choice of target GNN structure, experiments were conducted on target models with different numbers of GCN layers. Results are shown in Table IV in Section IV-F. We observed no significant difference in attack performance under different model structures. Therefore for the rest of this section, we only show the experimental results on the aforementioned target model structure.

The malicious nodes were randomly chosen 5% of the nodes in the target graph, containing approximately 0.25% of all links. To reduce randomness, all experiments were conducted over 10 runs. Notice that although the percentage of malicious nodes was 5% in the majority of discussions, Section IV-E shows 1% of malicious nodes produced decent link inference performance.

The supervised attack model first calculated the 8 distance metrics mentioned in Section II-B, followed by an (8×32) linear layer with ReLU activation and a dropout layer with a rate of 0.5, a (32×32) linear layer with ReLU activation and a dropout layer with a rate of 0.5, and a (32×2) linear layer.

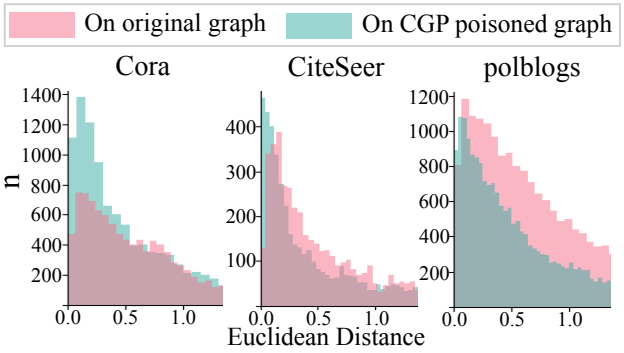

Fig. 5. Linked node pairs' posterior distance distribution, before and after applying CGP. After CGP, we still counted the originally linked nodes, therefore the number of counted nodes in the histograms remained the same.

The cross-entropy loss was adopted, and the optimizer was the Adam [22] with a learning rate of 0.0005. The model was trained for 100 epochs.

B. Effect of CGP on the Target Model

In the feature space, we observed the linked node pairs' distance distributions, before and after applying CGP. The distance distributions are illustrated in Fig. 5. Note that we still count the originally linked nodes for the poisoned histogram, so the total number of linked nodes, i.e. the integration of the distribution, does not change before and after CGP. In general, the mean of the histograms after applying CGP moved towards 0, with reduced variance, indicating more compact node posterior clusters. Since all attack methods mentioned in this work predict links based on the distances between node posteriors, Fig. 5 intuitively shows how CGP might benefit link inference.

C. Effect of CGP on the k-closest Attack

For this section, we tested the effect of CGP on the k-closest method in a realistic link inference attack setting. The target model was first trained. The node posteriors $\mathbf{h}_i$ were recorded. For the k-closest method, we followed the procedures introduced in Section II-B. For the combination of CGP and the k-closest method, 5% of nodes with index were first sampled as malicious nodes. CGP followed the implementation in **Algorithm 1**. Afterward, the target GNN was trained as usual and the k-closest method was applied. AUC was adopted for evaluation. The metric is particularly suitable for evaluating the k-closest method's performance since it removes the bias caused by manually choosing k.

Table I shows the AUC of k-closest without and with CGP, where the standard deviations are shown in the parenthesis. A general performance improvement was observed when applying the k-closest method after CGP, especially on Cora and CiteSeer. On Cora, except for a 0.001 decrease when using the Correlation distance, the average AUC improved from 0.01 to 0.05. On CiteSeer, we observed significant improvement in AUC under all distance functions. On polblogs, the introduction of CGP led to mixed results. This in a way agreed with the distance distribution shift observed in Fig. 5, where the original posterior distribution of the polblogs dataset already had a mean close to zero. This may explain why the introduction of CGP did not result in significant improvements. Nevertheless, the results in Table I show evidence that CGP can alter links between malicious users to make the target model more vulnerable to graph link inference attacks.

The Receiver Operating Characteristic curve (ROC) curves of the k-closest method without and with CGP (marked with orange frame) are shown in Fig. 6. Desirable improvement on the left portion of the ROC curve was observed after applying CGP under certain distance functions. This illustrated another benefit of applying CGP. When a lower k is chosen, the original k-closest method makes relatively more false positive inference, i.e. predicting links that do not exist. For attacks on sensitive information, such results could be misleading. After adding CGP, the false positive rate is lowered in the low k setting.

D. CGP and the Supervised Attack

One of the benefits of introducing malicious nodes into the target graph is the capability of using ground-truth links to train supervised attack models. In this work, the supervised attack was conducted and compared with the k-closest method. The supervised attack was also combined with CGP to further improve link inference accuracy.

Similar to Section IV-C, the target model was first trained to obtain node posteriors $\mathbf{H}$. Afterward, 5% nodes were randomly selected to form a set of malicious nodes. The target model was trained for another round to obtain the poisoned $\mathbf{H}$. The supervised attack was based on the node posteriors $\mathbf{h}_i$, with posterior pairs and link existence each acting as data and label to train the attack model.

For supervised attacks, precision and recall were used as metrics of evaluation. To establish a baseline, they were also calculated for the k-closest method. To select k for the unsupervised attack, we use the method in [12]: a K-means clustering was conducted on $\mathbf{H}$, with K set to 2. The number of data points in the smaller cluster was used as the k for the k-closest method. Except for the unsupervised k-closest method, all precisions and recalls were averaged in 10 runs.

Table II shows the mean precision, recall, and F-1 of all attacks. We observed two phenomena. First, the F-1 of the supervised attack improved up to 0.15 compared to the k-closest method. This is expected since extra information (malicious nodes) was available to the supervised attack. Second, CGP improved both the k-closest method and the supervised attack. On top of the two observations mentioned above, it is worth noticing that the graph link inference attack based on the k-closest method has decent recalls but low precisions. This indicates the k-closest method may produce many false link predictions. On the other hand, supervised attacks have significantly improved precision. Once a link is predicted by the supervised attack, the confidence level is higher, thus posing a greater privacy risk to key sensitive user relationships.

TABLE I
k-CLOSEST AUC ($\mu(\sigma)$) WITHOUT AND WITH CGP

	Cosine	Euclidean	Sqeuclidea	Correlation	Chebyshev	Braycurtis	Canberra	Cityblock
	Cora							
k-closest	0.898	0.675	0.675	**0.904**	0.654	0.769	0.839	0.695
CGP + k-closest	**0.907(.012)**	**0.698(.014)**	**0.698(.014)**	0.903(.022)	**0.682(.013)**	**0.827(.028)**	**0.844(.021)**	**0.717(0.013)**
	CiteSeer							
k-closest	0.893	0.502	0.502	0.894	0.532	0.568	0.678	0.503
CGP + k-closest	**0.916(.025)**	**0.589(.023)**	**0.589(.023)**	**0.918(.020)**	**0.583(.016)**	**0.751(.027)**	**0.794(.018)**	**0.616(.022)**
	polblogs							
k-closest	0.614	**0.594**	**0.594**	0.784	**0.549**	**0.788**	0.706	0.520
CGP + k-closest	**0.679(.020)**	0.556(.025)	0.556(.025)	**0.803(.024)**	0.538(.018)	0.781(.016)	**0.715(.023)**	**0.524(.016)**

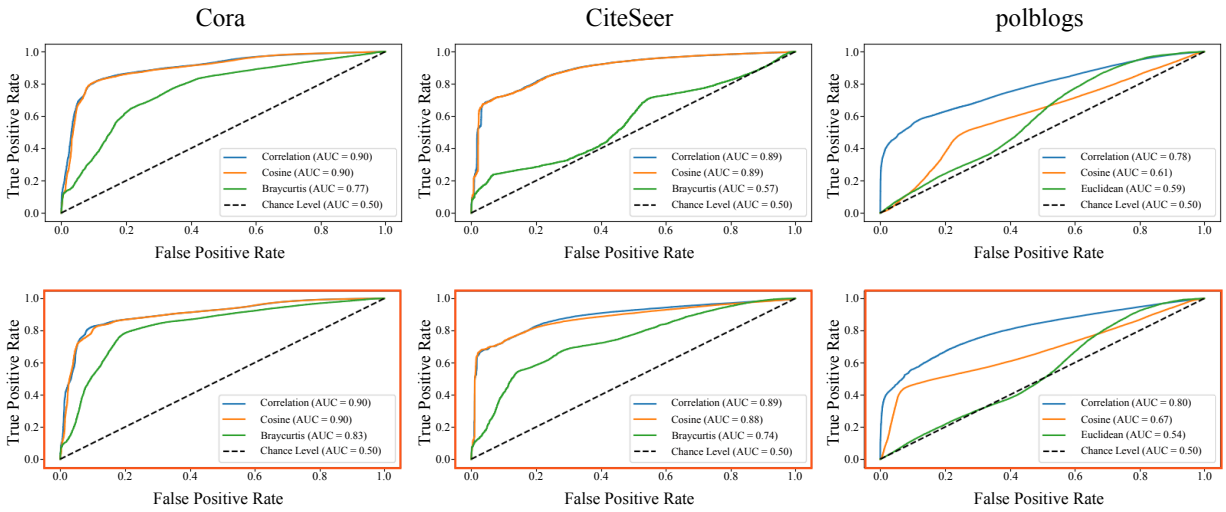

Fig. 6. The ROC curves of the k-closest method, without (upper row) and with CGP (lower row with orange frame).

TABLE II
PRECISION AND RECALL ($\mu(\sigma)$) OF GRAPH LINK INFERENCE ATTACKS

	Cora			CiteSeer			polblogs		
	precision	recall	F-1	precision	recall	F-1	precision	recall	F-1
k-closest	0.732	0.901	0.808	0.788	0.932	0.854	0.647	0.943	0.767
CGP + k-closest	0.743(.013)	**0.942(.001)**	0.831	0.801(.011)	**0.962(.004)**	0.874	0.646(.017)	**0.972(.012)**	0.776
supervised	0.995(.002)	0.882(.030)	0.935	**0.996(.001)**	0.841(.027)	0.912	**0.967**(.015)	0.911(.014)	0.938
CGP + supervised	**0.996(.003)**	0.911(.032)	**0.952**	0.995(.002)	0.923(.021)	**0.958**	0.958(.008)	0.947(.010)	**0.952**

E. Percentage of Malicious Nodes

To validate the effect of malicious node percentage on graph link inference attack performance, we conducted CGP and supervised attacks with malicious node percentage 1%, 2%, 5%, and 10%. Since the k-closest attack does not rely on malicious nodes, we choose the supervised attack for the evaluation of the malicious node percentage's effect. Table III shows the attack performance on polblogs. In general, increasing the malicious node percentage resulted in improved performance. However, more than 5%, e.g. 10%, of malicious nodes did not bring significant improvements. It is also not realistic for attackers to establish a large percentage of malicious nodes, therefore 5% was used in this work. Smaller percentages, e.g. 1% or 2%, still outperform the k-closest method by 0.15 in F-1. As such, we conclude that fewer than 5% malicious nodes are already effective in real-life attacks.

F. Target Model Structure

To validate whether the structure of the target model significantly affects the attack performance, we conducted CGP and

TABLE III
CGP + SUPERVISED ATTACK PERFORMANCE ON POLBLOGS WITH DIFFERENT PERCENTAGES OF MALICIOUS NODES

percentage	precision	recall	F-1
1%	0.955	0.894	0.923
2%	0.967	0.910	0.938
5%	0.960	0.945	0.952
10%	0.966	0.952	0.959

TABLE IV
CGP + SUPERVISED ATTACK PERFORMANCE WHEN THE TARGET MODEL HAS DIFFERENT NUMBER OF GCN LAYERS

	Cora	CiteSeer	polblogs
	one GCN layer		
precision	0.996	0.995	0.958
recall	0.906	0.921	0.951
F-1	0.949	0.957	0.954
	two GCN layers		
precision	0.989	0.994	0.964
recall	0.912	0.920	0.921
F-1	0.949	0.956	0.942
	three GCN layers		
precision	0.992	0.994	0.949
recall	0.922	0.917	0.933
F-1	0.956	0.954	0.941

the supervised attack on the target GNN model with 1, 2, and 3 GCN layers. The precision and recall of the attacks on the polblogs dataset are shown in Table IV. Significant changes in attack performance were not observed, therefore 1 GCN layer was adopted in this work.

V. CONCLUSION

This paper studies the risks of link leakage caused by malicious nodes in GNN. Without the malicious users, the attacker has to rely on distance-based unsupervised attacks. Such attacks have limitations including 1) being highly sensitive to the selection of distance functions, 2) requiring accurately estimated decision threshold, and 3) needing to consider all node pair combinations before making decisions. In this paper, with the introduction of malicious users, the practicality of link inference is greatly improved. The proposed method, CGP, only needs less than 5% malicious nodes, i.e. approximately 0.25% of links to become effective in changing the topology of all target GNN posteriors. Combined with the supervised attack enabled by ground-truth links between malicious nodes, the method proposed in this work achieved F-1> 0.95 on Cora, CiteSeer, and polblogs datasets. This study emphasizes the threat from malicious users in graph link inference attacks, it also underscores the vital importance of detecting and addressing malicious users in GNN applications.

VI. ACKNOWLEDGEMENT

This work was supported by the National Natural Science Foundation of China (Grant No: 92270123, 62072390, 62102334), and the Research Grants Council, Hong Kong SAR, China (Grant No: 15218919, 15203120, 15226221, 15225921, 15209922).

REFERENCES

[1] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, pp. 127–163, 2000.
[2] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.
[3] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 us election: divided they blog," in *Proceedings of the 3rd international workshop on Link discovery*, 2005, pp. 36–43.
[4] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.
[5] Y. Peng, B. Choi, and J. Xu, "Graph learning for combinatorial optimization: a survey of state-of-the-art," *Data Science and Engineering*, vol. 6, no. 2, pp. 119–141, 2021.
[6] K. Do, T. Tran, and S. Venkatesh, "Graph transformation policy network for chemical reaction prediction," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 750–760.
[7] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," *arXiv preprint arXiv:2003.00982*, 2020.
[8] L. Cao, H. Zhang, and L. Feng, "Building and using personal knowledge graph to improve suicidal ideation detection on social media," *IEEE Transactions on Multimedia*, vol. 24, pp. 87–102, 2020.
[9] A. Kapoor, X. Ben, L. Liu, B. Perozzi, M. Barnes, M. Blais, and S. O'Banion, "Examining covid-19 forecasting using spatio-temporal graph neural networks," *arXiv preprint arXiv:2007.03113*, 2020.
[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
[11] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.
[12] X. He, J. Jia, M. Backes, N. Z. Gong, and Y. Zhang, "Stealing links from graph neural networks." in *USENIX Security Symposium*, 2021, pp. 2669–2686.
[13] F. Wu, Y. Long, C. Zhang, and B. Li, "Linkteller: Recovering private edges from graph neural networks via influence analysis," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 2005–2024.
[14] 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*, 2018, pp. 2077–2085.
[15] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: a comprehensive review," *Computational Social Networks*, vol. 6, no. 1, pp. 1–23, 2019.
[16] S. Abu-El-Haija, A. Kapoor, B. Perozzi, and J. Lee, "N-gcn: Multi-scale graph convolution for semi-supervised node classification," in *uncertainty in artificial intelligence*. PMLR, 2020, pp. 841–851.
[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.
[18] E. Müller, "Graph clustering with graph neural networks," *Journal of Machine Learning Research*, vol. 24, pp. 1–21, 2023.
[19] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 2847–2856.
[20] A. Bojchevski and S. Günnemann, "Adversarial attacks on node embeddings via graph poisoning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 695–704.
[21] ——, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," *arXiv preprint arXiv:1707.03815*, 2017.
[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.