Machine Learning (2024) 113:2921–2953
https://doi.org/10.1007/s10994-022-06234-4

Dealing with the unevenness: deeper insights in graph-based attack and defense

Haoxi Zhan[1] · **Xiaobing Pei**[1]

Received: 22 February 2022 / Revised: 12 August 2022 / Accepted: 31 August 2022 /
Published online: 7 October 2022

Abstract
Graph Neural Networks (GNNs) have achieved state-of-the-art performance on various graph-related learning tasks. Due to the importance of safety in real-life applications, adversarial attacks and defenses on GNNs have attracted significant research attention. While the adversarial attacks successfully degrade GNNs' performance significantly, the internal mechanisms and theoretical properties of graph-based attacks remain largely unexplored. In this paper, we develop deeper insights into graph structure attacks. Firstly, investigating the perturbations of representative attacking methods such as Metattack, we reveal that the perturbations are unevenly distributed on the graph. By analyzing empirically, we show that such perturbations shift the distribution of the training set to break the i.i.d. assumption. Although degrading GNNs' performance successfully, such attacks lack robustness. Simply training the network on the validation set could severely degrade the attacking performance. To overcome the drawbacks, we propose a novel k-fold training strategy, leading to the Black-Box Gradient Attack algorithm. Extensive experiments are conducted to demonstrate that our proposed algorithm is able to achieve stable attacking performance without accessing the training sets. Finally, we introduce the first study to analyze the theoretical properties of graph structure attacks by verifying the existence of trade-offs when conducting graph structure attacks.

Keywords Graph neural networks · Graph convolutional networks · GNN · Adversarial attacks

Editors: Dana Drachsler Cohen, Javier Garcia, Mohammad Ghavamzadeh, Marek Petrik, Philip S. Thomas.

✉ Xiaobing Pei
xiaobingp@hust.edu.cn

Haoxi Zhan
zhanhaoxi@foxmail.com

[1] School of Software Engineering, Huazhong University of Science and Technology, Luoyu Road, Wuhan 430074, Hubei, China

1 Introduction

Graph structured data is widely used in a variety of domains, such as social networks (Rossetti et al., 2016), academic publishing (Fricke, 2018), recommender systems (Liu et al., 2016), financial transactions (Ron and Shamir, 2013), and public health (Chen et al., 2020). How to learn effective graph representations has long been an important research direction. Recently, Graph Neural Networks (GNNs) have become the mainstream method for graph representation learning (Zhou et al., 2020). Graph Convolutional Network (GCN) (Kipf and Welling, 2017) achieved state-of-the-art performance in the node-classification task and is considered as the most representative GNN model. Then, numerous GNN models, such as GAT (Velickovic et al., 2018), GraphSAGE (Hamilton et al., 2017), and JK-Net (Xu et al., 2018), have been proposed.

It has been revealed that deep learning models often lack robustness (Goodfellow et al., 2015). It is possible to fool the model by generating perturbations deliberately. GNNs are no exceptions. Adversarial attacks against GNNs could be classified by various standards (Jin et al., 2020). According to the goal of the attackers, we can divide graph attacks into two groups: *targeted attacks*, which aim to misclassify a small set of target nodes, and *non-targeted attacks*, which aim to degrade the overall performance of GCN models without specifying any victim nodes. According to the amount of knowledge available to the attackers, three different types of settings are proposed: *white-box attacks* allow the attackers to access all possible information, *grey-box attacks* prohibit the attackers to access model parameters and testing labels while in *black-box attacks*, even the information about training set is not available. The attacking method also varies such that some methods modify the node features while other methods modify the edges.

Various defense methods have also been proposed. For instance, GCN-Jaccard (Wu et al., 2019) increases the robustness of GCNs by eliminating edges with low similarity before the training process. R-GCN (Zhu et al., 2019) utilizes the attention mechanism which regards node features as Gaussian distributions and assigns attention scores according to the variances. By introducing structure learning, Pro-GNN (Jin et al., 2020) achieves promising results when defending against structure perturbations.

Although defense algorithms have emerged to enhance the security of GNNs, studies on deep learning models show that it is possible to improve the adversarial attacks to degrade the defense performance (Athalye et al., 2018; Carlini and Wagner, 2017). Meanwhile, in-depth studies on adversarial attacks are efficient tools to develop insights into deep learning models (Geirhos et al., 2019). In comparison to Deep Neural Networks (DNNs), adversarial attacks on GNNs remain poorly understood in general, despite the fact that many patterns have been found by various studies (Zügner et al., 2020; Sun et al., 2020).

In order to make further improvements on attacking methods and to develop efficient defense methods, it is important to identify the patterns of the attacks. In this work, we focus on non-targeted structure attacks against GNNs. Particularly, we are interested in the following research questions:

- **RQ1** Why are graph structure attacks effective?
- **RQ2** Are there any drawbacks of the state-of-the-art algorithms? If so, is it possible to improve them?
- **RQ3** Do any theoretical limitations and unavoidable trade-offs exist in graph structure attacks?

In this paper, we make a leap towards the understanding of graph structure attacks. Conducting case studies in representative methods, we show that the perturbations chosen by gradient-based methods are highly correlated with the training set and they are distributed unevenly on the graph. As a result, we reveal that a simple defense strategy is to train the GNN with the validation set. Based on the findings, we propose the Black-Box Gradient Attack (BBGA) algorithm to evenly perturb the graphs without accessing any ground truth labels. Extensive experiments demonstrate the effectiveness of our proposed method. Based on our empirical studies, we analyze the advantages and drawbacks of various attacking strategies and we illustrate the existence of trade-offs in graph structure attacks. In a nutshell, our contributions are summarized below:

- **How state-of-the-art attacks work** We study the patterns of two representative non-targeted structure attacks, Mettack (Zügner and Günnemann, 2019) and PGD-Attack (Xu et al., 2019). Creating a taxonomy of the perturbations, we show that the perturbations are unevenly distributed and such unevenness breaks the i.i.d. assumption. Although achieving remarkable attacking performance, we show that such unevenness could also be easily utilized by defenders.
- **Black-box Gradient Attack (BBGA)** While it is believed that training surrogate models without ground-truth labels is impossible, we propose the first gradient-based black-box attacking method to our best knowledge. It is also the first non-targeted graph structure attack without permission to do black-box queries. Exploiting the spectral clustering, we train the surrogate model with pseudo-labels and then evenly distribute the perturbations via a novel k-fold training strategy.
- **Theoretical properties and trade-offs** Conducting both empirical studies and mathematical analysis, we reveal the existence of trade-offs in graph structure attacks. To our best knowledge, this is the first study on the theoretical limitations of graph structure attacks.

The rest of our paper is organized as follows. We list and review related works in Section 2, then we formally define the mathematical notations in Section 3. In Section 4, we analyze the patterns of notable attacks via a series of case studies. In Section 5, we introduce our proposed BBGA method, explain it in detail, and report the experimental results to demonstrate its effectiveness. The theoretical limitations and trade-offs are discussed in Section 6. Finally, we conclude the paper and discuss the future directions in Section 7.

2 Related works

We briefly review previous works on GNNs and graph adversarial attacks and defenses in this section.

2.1 Graph neural networks (GNNs)

Graph Neural Networks (GNNs) are deep learning methods that focus on graph data. In this paper, we mainly focus on Graph Convolutional Networks (GCNs) which utilize convolution operations to pass messages between the nodes (Zhang et al., 2018). Based on the methodologies of message-passing, GCNs could be divided into various families. *Spectral-based* GCNs utilize graph Fourier transform (GFT) to aggregate graph signals. It is

introduced by Bruna et al. (2014) at first. Then Defferrard et al. simplify spectral GCNs with Chebyshev polynomials (Defferrard et al., 2016). *Spatial-based* GCNs aggregate local information in the spatial domain. The representative GCN model proposed by Kipf and Welling (2017) is a hybrid model which could be considered as both spectral and spatial. Following that, a number of spatial methods are proposed (Velickovic et al., 2018; Hamilton et al., 2017; Chen et al., 2018). *Decoupled* GCNs originate from spatial-based GCNs but they decouple the convolution operation, which propagates information, with the feature transformation process. Notable decoupled GCNs include APPNP (Klicpera et al., 2019) and PTA (Dong et al., 2021). For a thorough review of GCNs, we refer the readers to recent surveys (Zhang et al., 2018; Zhou et al., 2020; Wu et al., 2021) and texts (Liu and Zhou, 2020; Ma and Tang, 2020).

2.2 Graph adversarial attacks and defenses

Extensive studies have revealed that GCNs are vulnerable to deliberate perturbations. Targeted attacks aim to fool the GCNs to misclassify certain target nodes. White-box approaches such as FGA (Chen et al., 2018) and IG-FGSM (Wu et al., 2019) are usually based on gradient information, which is considered to be inaccessible in real-life scenarios. Nettack, which is the most representative grey-box targeted attack algorithm, proposes to utilize a surrogate model to approximate the gradients (Zügner et al., 2018). Besides modifying features and edges, new attacking methods such as AFGSM (Wang et al., 2020), which injects vicious nodes to the graph, and backdoor attack (Zhang et al., 2021), which creates a "backdoor" subgraph for injection, have been proposed. In black-box situations, reinforcement learning is utilized by RL-S2V (Dai et al., 2018) while GF-Attack (Chang et al., 2019) provides a framework to utilize both the graph structure and feature information. TDGIA (Zou et al., 2021) is a black-box node injection attack. The recent development of targeted attacks tries to limit the number of perturbations to one node (Finkelshtein et al., 2020) and to provide universal attacks via preparing a set of attack nodes and fake nodes (Dai et al., 2022).

In this paper, we mainly focus on non-targeted attacks, which are less studied. Representative white-box methods include PGD Attack and Min-Max Attack (Xu et al., 2019) while in grey-box situations, Metattack (Zügner and Günnemann, 2019) utilizes a surrogate model and a meta-learning framework to modify edges according to the approximated gradients. NIPA (Sun et al., 2020) is a non-targeted node injection attacking algorithm. Zhang et al. (2020) degrades model performance by flipping the labels of training nodes. Feature-based black-box non-targeted attacks include RWCS (Ma et al., 2020) and InfMax (Ma et al., 2022). Geisler et al. discussed adversarial attacks against large-scale graphs (Geisler et al., 2021). Although several new attacking methods have emerged, Metattack is regarded as a state-of-the-art algorithm and is used as a primary baseline in various studies (Jin et al., 2020, 2021; Guo et al., 2022).

Due to the importance of safety in the real world, various defense methods have also been proposed. We categorize representative defense algorithms into three groups. *Attention-based* methods such as R-GCN (Zhu et al., 2019) and PA-GNN (Tang et al., 2020) utilize attention scores to control the propagation of adversarial information. *Preprocessing* methods such as GCN-Jaccard (Wu et al., 2019) and GCN-SVD (Entezari et al., 2020) purify the graph structure according to some criteria before the training of the model. *Robust structure learning* methods such as Pro-GNN (Jin et al., 2020) learn the optimal graph structure iteratively during the training. Besides the heuristic defense methods, a new

trend in defense research is to certify the robustness of the models. Zügner et al. Zügner and Günnemann (2019) provides a method to certify the robustness of nodes. Bojchevski et al. proposed a discrete certificate. Jin et al. (2020) utilizes Lagrange dualization and convex envelope to certificate the robustness of GCN under topological attacks. Reliable GNN (Geisler et al., 2020) is a certified defense method based on a novel Soft Medoid aggregation function. Recently, Schuchardt et al. (Schuchardt et al., 2021) proposed a collective certificate method. For a more comprehensive review of graph-based attacks and defenses, we refer the readers to various survey articles (Jin et al., 2020; Sun et al., 2018; Günnemann, 2022).

3 Preliminaries

In this section, we introduce the notations used in this paper as well as fundamental concepts and experimental environments.

3.1 Mathematical notations

In this paper, we denote a graph with N nodes as $G = (V, E)$ where $V = \{v_1, \cdots, v_N\}$ is the node set and $E = \{e_1, \cdots, e_E\} \subseteq V \times V$ is the edge set. The set of vertices V is usually divided into the training set V_{train}, the validation set V_{val} and the testing set V_{test}. $\mathbb{Z}_n$ is a quotient ring of integers modulo n and we use it to denote the set $\{0, 1, \cdots, n-1\}$. Specifically, $\mathbb{Z}_2 = \{0, 1\}$. We denote the structure of the G via the adjacency matrix $A \in \mathbb{Z}_2^{N \times N}$ whose element $A_{ij} = 1$ if and only if $\exists i \in [1, E]$ s.t. $e_i \in E$ connects v_i and v_j. Node features are stored in a matrix $X \in \mathbb{R}^{N \times F}$ where F is the number of dimensions of node features. Since the node features in graphs are often of the bag-of-words kind (Wu et al., 2019), we have $X \in \mathbb{Z}_2^{N \times F}$. The label set of a dataset is denoted by C while pseudo-labels used during training are denoted as C_p.

3.2 Graph convolutional networks (GCNs)

Despite many GCN models have been proposed, in this paper, we mainly consider the most representative one introduced by Kipf and Welling (2017). Each layer of a GCN aggregates messages according to the graph structure and then performs space transformations on the node features. Such a graph convolutional layer could be denoted as the following equation:

$$H^{(l+1)} = \sigma(\hat{A} H^{(l)} W^{(l)}), \tag{1}$$

where $\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix such that $\tilde{A} = A + I_N$, $\tilde{D}_{i,i} = \sum_j \tilde{A}_{i,j}$. σ is a non-linear activation function. A typical GCN network consists of two layers, the whole network is usually described as:

$$Z = f(X, A) = \text{softmax}(\hat{A} \sigma(\hat{A} X W^{(0)}) W^{(1)}). \tag{2}$$

The GCN network is usually trained with a cross-entropy loss. Noticing that two-layer GCNs aggregate information within 2-hop neighborhoods for each node, a simplified and linearized version is usually used in adversarial attacks:

$$Z = f(A, X) = \text{softmax}(\hat{A}^2 XW). \tag{3}$$

To utilize GCN in inductive settings, Hamilton et al. (2017) introduced a variant of GCN with a different normalization method:

$$h_v^{(l)} = \sigma\left(W_l \cdot \frac{1}{\tilde{D}_{(v,v)}} \sum_{u \in \tilde{\mathcal{N}}(v)} h_u^{(l-1)} \right), \tag{4}$$

where $h_v^{(l)}$ is the hidden representation of node v in the l^{th} layer and $\tilde{\mathcal{N}}(v)$ is the neighborhood of node v in the self-looped graph with $\tilde{A}$ as the adjacency matrix.

3.3 Metattack

Zügner and Günnemann (2019) is a gradient-based grey-box graph attacking algorithm. Being directly connected to the training loss, gradients are widely used in deep learning adversarial attacks (Yuan et al., 2019). Being denied to access model parameters, Metattack trains the surrogate model described in Eq. 3 to approximate the gradients of the GCN model. To find the best edge to perturb, the adjacency matrix is regarded as a hyperparameter and the meta-gradients are computed after the training of the surrogate model:

$$\nabla_G^{\text{meta}} := \nabla_G \mathcal{L}_{\text{atk}}(f_{\theta^*}(G)) \text{ s.t. } \theta^* = \text{opt}_\theta(\mathcal{L}_{\text{train}}(f_\theta(G))), \tag{5}$$

where $\mathcal{L}_{\text{atk}}$ is the target function that the attacker aims to optimize, opt is a training procedure and $\mathcal{L}_{\text{train}}$ is the training loss. It is revealed that when $\mathcal{L}_{\text{atk}} = -\mathcal{L}_{\text{self}}$, where $\mathcal{L}_{\text{self}}$ is the cross-entropy loss on the unlabelled nodes with predicted pseudo-labels, the algorithm reaches its best performance.

A greedy algorithm, which chooses exactly one edge a step, is employed to perform perturbations. The score of a node pair (u, v) is defined as:

$$S(u, v) = \nabla_{A_{uv}}^{\text{meta}} \cdot (-2 \cdot A_{uv} + 1), \tag{6}$$

where A is the adjacency matrix. The signs of meta-gradients are flipped for connected node pairs to yield the gradients for removing the edge. In each iteration, the algorithm picks the potential perturbation with the highest score.

The total number of perturbations is controlled by a budget constraint Δ, which is usually defined via a perturbation rate $\delta = \frac{\Delta}{|E|}$.

3.4 The set-cover problem

The Set-Cover problem (Slavík, 1997) is a classic NP-hard problem. Given a *universe* of m elements $\mathcal{U} = \{1, \cdots, m\}$ and k sets $\mathcal{S} = \{S_1, \cdots, S_k\}$ such that $\bigcup_{i=1}^k S_i = \mathcal{U}$, the optimization version of Set-Cover aims to find a cover $\mathcal{C} \subseteq \mathcal{S}$ with the fewest number of sets such that $\bigcup_{S \in \mathcal{C}} S = \mathcal{U}$. Let $x : \mathcal{S} \to \mathbb{Z}_2$ be an indicator function such that:

$$\forall S \in \mathcal{S}, x(S) = \begin{cases} 0, & S \notin \mathcal{C} \\ 1, & S \in \mathcal{C} \end{cases}. \tag{7}$$

Then the optimization problem is formulated as (Vazirani, 2013):

$$\begin{aligned} &\text{minimize} \sum_{S \in \mathcal{S}} x(S) \\ &\text{subject to } \forall e \in \mathcal{U}, \sum_{S : e \in S} x(S) \geq 1 \cdot \\ &\forall S \in \mathcal{S}, x(S) \in \mathbb{Z}_2 \end{aligned} \tag{8}$$

3.5 Experimental environments

This paper contains a series of case studies and experiments. In this subsection, we introduce the datasets and models used in the paper.

3.5.1 Datasets

This work mainly focuses on homogeneous graphs as both Zügner and Günnemann (2019) and our selected defense methods are designed for homogeneous graphs. Cora, Citesser, and Cora-ML, which are three commonly-used real-world benchmark datasets, are employed in our studies. Following (Zügner and Günnemann, 2019), we only consider the largest connected components of the graphs. The statistics of the datasets are summarized in Table 1. Following (Jin et al., 2020), we randomly pick 10% of nodes for training, 10% of nodes for validation, and the remaining 80% of nodes for testing.

3.5.2 Defense methods

When evaluating attacking performance, defense methods are employed. In our studies, both vanilla GCN and representative defense algorithms are utilized. For GCN and RGCN we use the official implementation along with its hyperparameters. For other models, we use the implementation and hyperparameters in Li et al. (2020).

- **GCN** (Kipf and Welling, 2017) In this work, we focus on attacking the representative Graph Convolutional Network (GCN).
- **GCN-Jaccard** (Wu et al., 2019) Noticing that existing graph adversarial attacks tend to connect dissimilar nodes, GCN-Jaccard removes the edges that connect nodes with Jaccard similarity scores lower than a threshold η before training the GCN.
- **R-GCN** (Zhu et al., 2019) R-GCN utilizes the attention mechanism to defend against perturbations. Modeling node features as normal distributions, the R-GCN assigns attention scores according to the variances of the nodes.
- **Pro-GNN** (Jin et al., 2020) Exploring the low rank and sparsity properties of adjacency matrices, Pro-GNN combines structure learning with the GNN in an end-to-end manner.

Table 1 Statistics of the datasets

Dataset	$\lvert V \rvert$	$\lvert E \rvert$	Classes	Features	Avg degree
Cora	2485	5069	7	1433	2.04
Citeseer	2110	3668	6	3703	1.74
Cora-ML	2810	7981	7	2879	2.84

- **GCN-SVD** (Entezari et al., 2020) A preprocessing method based on low-rank approximation of the graph adjacency matrices.

3.5.3 Attacking methods

Since we study the internal mechanisms of attacking methods and we propose BBGA, a novel attacking algorithm, we utilize a number of attacking methods as either investigating targets or baselines.

- **Metattack** Following previous studies (Jin et al., 2020), we choose Zügner and Günnemann (2019) as a primary object for our study. The perturbed graphs are attacked with the Meta-Self model without the log-likelihood constraint.
- **PGD Attack** We also choose PGD Attack (Xu et al., 2019) when investigating the patterns of graph structure attacks. PGD Attack is a gradient-based white-box non-targeted attacking method, since it's white-box and it requires even the gradient information, we only employ it as an investigating object.
- **Random Attack** The random attack is also investigated in this study. Since previous studies have revealed that attacking algorithms mostly choose to add edges (Wu et al., 2019), in our settings Random Attack is not allowed to delete edges.
- **DICE** DICE, a white-box baseline introduced by Zügner and Günnemann (2019), is based on Random Attack but it only links nodes of different labels.

3.5.4 Hardware and software packages

We conducted our experiments on an Ubuntu 16.04 LTS server with two E5-2650 CPUs and 4 GTX 1080Ti GPUs. Python packages we used include Pytorch 1.5.0, Scikit-Learn 0.23.1, NumPy 1.18.5, and SciPy 1.3.1. Tensorflow 1.15.0 is used in R-GCN.

4 Case studies on representative methods

We study the research question **RQ1** in this section. Adversarial attacks generate deliberate perturbations on graph data. Hence, to investigate why adversarial attacks work, it is necessary to analyze the perturbations made by attacking algorithms. Previous studies have found out that dissimilar nodes tend to be connected by attackers (Wu et al., 2019). In this work, we focus on the distributions of such carefully crafted perturbations.

Consisting of 2 graph convolutional layers, a typical GCN network is only able to aggregate information with the 2-hop neighborhood of each node. Xu et al. proved the following theorem:

Theorem 1 (Xu et al. (2018)) *If all paths in the computation graph of the model are activated with the same probability. Given an L-layer GCN with Eq. 4 as the normalization method, then* $\forall i, j \in V$, $\mathbb{E}[\frac{\partial H_j}{\partial X_i}]$ *is equivalent to the probability of reaching node j via a k-step random walk starting at node i.*

Inspired by Theorem 1, it is natural to raise the question that whether grey-box gradient attacks mainly perturb edges that are adjacent to the training set or not. To investigate the

patterns of the edges picked by representative methods, we create a taxonomy of edge perturbations and conduct a series of case studies.

4.1 A taxonomy of perturbations

We propose a taxonomy for graph structure perturbations. For each perturbation, which is a flip of edge (i, j), three possible types are defined:

- **Type 1** Both nodes i and j are inside the training set.
- **Type 2** Exactly one of i, j is inside the training set.
- **Type 3** Neither i nor j is inside the training set.

We compare four different attacking methods: Metattack, PGD Attack, Random Attack, and DICE. For all the methods, we use the implementations and hyperparameters in the DeepRobust library (Li et al., 2020). For each attacking method, we run the attacks 10 times on a random data split. For each attacking run, we also train the attacked graph with GCN once. Hence, for each attacking setting, 10 runs are conducted for both attack and defense. The Cora dataset is used in this case study.

The distributions of the 3 types of perturbations are reported in Table 2 and the attacking performance against GCN is reported in Table 3.

As revealed in the tables, gradient-based methods, which produce fewer type 3 perturbations, have better attacking performance than the randomized heuristics. Especially, more

Table 2 Distributions of perturbations by different attacks

Ptb(%)	Attack	Type 1 (%)	Type 2 (%)	Type 3 (%)
5%	Metattack	10.55%	88.97%	0.47%
	PGD	21.97%	77.18%	0.84%
	Random	0.99%	17.08%	81.94%
	DICE	1.11%	17.31%	81.58%
10%	Metattack	8.26%	90.47%	1.26%
	PGD	19.33%	78.58%	2.09%
	Random	0.93%	17.47%	81.60%
	DICE	1.01%	17.32%	81.68%
15%	Metattack	8.50%	89.64%	1.86%
	PGD	19.15%	75.94%	4.91%
	Random	0.91%	17.80%	81.29%
	DICE	0.87%	18.76%	80.37%
20%	Metattack	7.43%	90.51%	2.05%
	PGD	19.30%	73.85%	6.85%
	Random	0.97%	18.08%	80.95%
	DICE	1.03%	17.26%	81.72%
25%	Metattack	7.78%	89.91%	2.31%
	PGD	18.47%	72.29%	9.24%
	Random	1.19%	18.26%	80.54%
	DICE	1.15%	17.57%	81.28%

Table 3 GCN performance on Cora against the 3 methods (Accuracy ± Std)

Attack	5%	10%	15%	20%	25%
Metattack	75.97±1.54	71.33±1.81	63.73±2.44	57.15±4.42	51.99±4.97
PGD attack	80.45±1.69	76.88±1.87	73.53±1.55	72.76±2.29	70.01±2.32
Random attack	82.25±1.13	80.56±1.18	79.28±0.88	78.48±0.76	76.55±0.83
DICE	81.27±1.06	79.55±1.06	77.97±0.99	75.00±2.04	72.99±1.26

than 95% of perturbations generated by Metattack are of either type 1 or type 2. Another interesting finding is that although PGD Attack has more type 1 perturbations, which are intuitively most relevant to the training set, it's inferior to Metattack in terms of attacking performance.

Hence, current methods, which utilize the gradient information in the attacking procedures, perturb the graph unevenly such that they flip a much higher proportion of edges near the training set. *What are the effects of such unevenness? Is it related to how attacking algorithms work?* To answer the previous questions, we conduct an ablation study.

4.2 An ablation study on unevenness

We create two variants of Metattack. The first one is **R-Metattack**, which is a restricted version of Metattack such that no type 3 perturbations are allowed. The second variant is **3-Metattack**, in which a mask is utilized to prevent it from generating type 1 and type 2 perturbations. We adopt the implementation and hyperparameters from DeepRobust (Li et al., 2020) and three commonly-used datasets: Cora, Citeseer, and Cora-ML are used for comparison.

As reported in Table 4, Metattack and R-Metattack have similar performances in all situations. Metattack has better performance in 8 attacking settings while R-Metattack

Table 4 Performance of Metattack, R-Metattack, and 3-Metattack

Dataset	Ptb(%)	Metattack	R-Metattack	3-Metattack
Cora	5%	75.97%	76.95%	81.03%
	10%	71.33%	71.61%	78.84%
	15%	63.73%	65.26%	76.93%
	20%	57.15%	55.81%	75.02%
	25%	51.99%	51.57%	73.15%
Citeseer	5%	74.07%	73.60%	75.33%
	10%	70.47%	69.61%	73.70%
	15%	65.28%	64.99%	72.52%
	20%	62.03%	62.18%	71.11%
	25%	55.57%	55.47%	69.89%
Cora-ML	5%	78.55%	78.80%	81.94%
	10%	67.28%	67.81%	76.73%
	15%	58.47%	58.95%	74.45%
	20%	47.00%	47.46%	70.68%
	25%	42.63%	41.56%	66.52%

wins in the other 7. Using the Friedman Test (Zhou, 2016; Friedman, 1940), we have $N = 15, k = 2$, and the average ranks for Metattack and R-Metattack are $\frac{22}{15}$ and $\frac{23}{15}$ respectively. Hence,

$$\begin{aligned} \tau_{\chi^2} &= \frac{k-1}{k} \cdot \frac{12N}{k^2-1} \sum_{i=1}^{k} \left(r_i - \frac{k+1}{2} \right)^2 = \frac{1}{15}, \\ \tau_F &= \frac{(N-1)\tau_{\chi^2}}{N(k-1) - \tau_{\chi^2}} = \frac{1}{16} = 0.625, \end{aligned} \tag{9}$$

which is smaller than the critical value $c = 4.600 (\alpha = 0.05)$. This implies that Metattack and R-Metattack have similar performance. On the other hand, 3-Metattack performs significantly worse than the other two methods. Hence, type 1 and type 2 perturbations, which lead to the unevenness, play major roles in Metattack. This implies that Metattack mostly relies on type 1 and type 2 perturbations.

4.3 The distribution shift

In this subsection, we make one step further to show that the unevenness created by type 1 and type 2 perturbations leads to a shift of the distributions of node purities.

Definition 1 *(node purity)* The purity $\mathcal{P}_v$ of a node $v \in V$ is the proportion of its neighbors with the same label of v. Formally, if $| \{ j \mid j \in V \wedge A_{vj} = 1 \} | \neq 0$, $\mathcal{P}_v = \frac{|\{j | j \in V \wedge A_{vj}=1 \wedge L_v = L_j\}|}{|\{j | j \in V \wedge A_{vj}=1\}|}$, if $| \{ j \mid j \in V \wedge A_{vj} = 1 \} | = 0$, we define $\mathcal{P}_v = 0$.

Since perturbations generated by Metattack and PGD Attack are highly correlated to the training set, we study the purities of training nodes separately. The mean training purity $\bar{\mathcal{P}}_{\text{train}}$ and mean non-training purity $\bar{\mathcal{P}}_{\text{non}}$ are defined as:

$$\bar{\mathcal{P}}_{\text{train}} = \frac{\sum_{i \in V_{\text{train}}} \mathcal{P}_i}{| V_{\text{train}} |}, \tag{10}$$

$$\bar{\mathcal{P}}_{\text{non}} = \frac{\sum_{i \in V_{\text{val}}} \mathcal{P}_i + \sum_{i \in V_{\text{test}}} \mathcal{P}_i}{| V_{\text{val}} \cup V_{\text{test}} |}. \tag{11}$$

For the Cora dataset, we calculate both $\bar{\mathcal{P}}_{\text{train}}$ and $\bar{\mathcal{P}}_{\text{non}}$ for 10 different attacked graphs and the results are averaged. The results for gradient-based attacks and randomized heuristics are reported in Figure 1.

As shown in Figure 1, Metattack and PGD Attack, which are gradient-based methods relying on uneven perturbations, severely shift the purity distributions of the training nodes. Metattack, which produces extremely uneven perturbations, influences the distributions most significantly. When the perturbation rates are 25%, the average value of $\bar{\mathcal{P}}_{\text{train}}$ is only 42.38%, while the average value of $\bar{\mathcal{P}}_{\text{non}}$ is at 70.34%. Although PGD Attack has a higher proportion of type 1 flips, it only shifts $\bar{\mathcal{P}}_{\text{train}}$ to 62.22%. In statistical learning, the nodes in the training set are usually sampled dependently and identically from a distribution (Shalev-Shwartz and Ben-David, 2014). However, the figures show that under Metattack and PGD Attack, the purities of training nodes and testing nodes are no longer

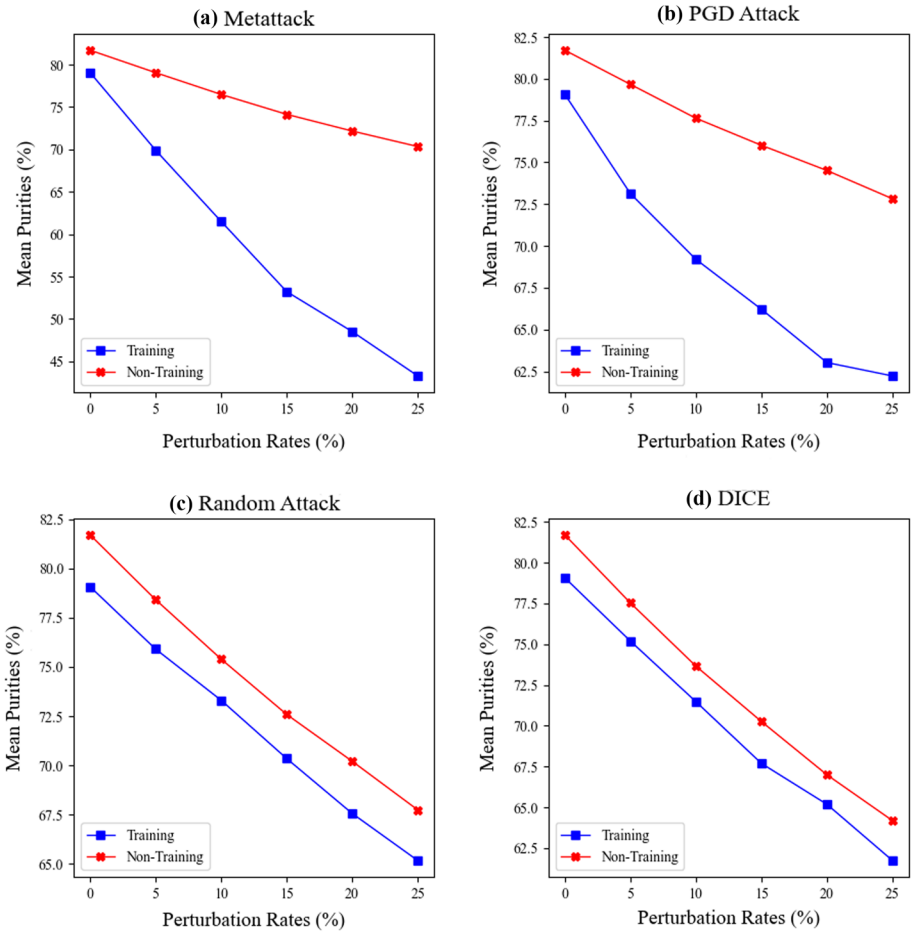

Fig. 1 The purity distributions of both training and non-training nodes after being attacked

independently and identically distributed. Hence we identify the internal mechanism of gradient-based graph structure attacks such as Metattack.

5 Black-box gradient attack

5.1 Vulnerability of existing methods

Although the distribution shift helps in attacking, it could be easily utilized by the defender. Since it works by changing the distribution of the training set, a natural idea is to use another training set. We develop **Flip-GCN**, which is a training strategy that trains the GCN with the validation set, to defend against Metattack. The network structure of Flip-GCN is the same as the GCN and we adopt the hyperparameters in the official implementation of GCN (Kipf and Welling, 2017). The only difference is that we exchange the training

set and the validation set. Following the experimental settings of Jin et al. (2020), we choose 5 perturbation rates ranging from 5 to 25% and we randomly select 10% of nodes as the training set, 10% for validation, and the remaining 80% as the testing set. The graph is perturbed by Metattack without the log-likelihood restraint. As reported in Table 5, when V_{train} and V_{val} are flipped, the testing accuracy increased dramatically. Hence, such uneven attacks are not robust enough and they are only effective for certain training sets.

To increase the robustness of adversarial attacks, the perturbations should be unbiased to any data splits. However, existing even attacks such as Random Attack and DICE have quite limited attacking performance. Hence, to alleviate the problem, we are supposed to have an advanced black-box attacking method, which has no access to the training data in order to prevent such biases.

While most black-box attacks are based on reinforcement learning (Dai et al., 2018; Ma et al., 2019), random walk-based attacks such as RWCS are also promoted recently to avoid model inquiries (Ma et al., 2020). However, RWCS focuses on perturbing features instead of graph structures. To our best knowledge, no existing non-targeted black-box structure attack works without black-box inquiries. Meanwhile, gradients are not exploited in black-box attacks since a surrogate model is needed. However, gradients are powerful tools to pick edges for perturbations. In this paper, by disengaging gradients from any specific training set, we propose the novel Black-Box Gradient Attack (BBGA) algorithm to answer **RQ2**.

5.2 Attack condition

In this subsection, we introduce the attacking condition of the BBGA algorithm.

Goal As a non-targeted attack, the attacker's goal is to decrease the classification accuracies of GNNs.

Knowledge As a black-box attack, access to model parameters, training data, and ground-truth labels are denied. The graph $G = (V, E)$ and the node features are considered to be accessible.

Constraints The number of changes is restricted by a budget Δ such that $\|A' - A\|_0 \leq 2\Delta$. Here A' is the modified adjacency matrix and we have 2Δ due to the symmetry of adjacency matrices. In addition, various defense algorithms have taken advantage of the connections between dissimilar nodes (Wu et al., 2019; Jin et al., 2020). We believe that such easily detected modifications are supposed to be restricted since they could be easily eliminated by a preprocessing algorithm. Noticing that node features in common graph tasks are encoded in the one hot manner, we propose a similarity constraint such that for any pair of nodes $v_1, v_2 \in V$, if their Jaccard similarity score J_{v_1,v_2} is smaller than a threshold value η, the connection of v_1 and v_2 is disallowed. The definition of the Jaccard similarity score is:

Table 5 The experimental results of the flip strategy (Accuracy ± Std)

Ptb Rate	GCN	Flip-GCN
5%	76.82±1.10	81.76±1.23
10%	71.59±1.79	81.19±0.89
15%	63.61±2.54	80.36±1.33
20%	55.88±5.03	80.50±0.69
25%	52.21±4.71	79.26±1.10

$$J_{v_1,v_2} = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}, \tag{12}$$

where $M_{ab}(a, b \in \{0, 1\})$ represents the number of features which have value a in node v_1 and value b in node v_2. The constraints are summarized as a function $\phi(G) : G \to \mathcal{P}(V \times V)$, where G is the graph to attack. The function ϕ maps the graph to a set of valid node pairs.

5.3 Pseudo-label and surrogate model

Since no ground-truth labels are accessible, we need to have pseudo-labels to train the surrogate model. We utilize spectral clustering to generate pseudo-labels (Pedregosa et al., 2011). Parameters of the spectral clustering algorithm are chosen according to the Calinski-Harabasz Score.

Since it's not accurate to approximate the gradients of GNNs with pseudo-labels generated by a spectral clustering algorithm, a simplified GCN described in 3 is employed as the surrogate model. The surrogate model is trained with the pseudo-labels on a random training set, which has no relation to the training set of the defense model.

5.4 k-fold greedy attack

As demonstrated in Section 2, a main drawback of Metattack is the uneven distribution of the modifications. In the black-box scenario in which the training set is not accessible, the attacker is supposed to distribute its modifications in the whole graph. In this paper, we proposed a novel k-fold greedy algorithm to solve this problem.

We divide the node set V into k partitions $V_1, V_2, \cdots, V_k$. For the i^{th} partition $V_i (i \in [1, k])$, we predict $\hat{C}_i$, which is the labels of nodes in $V - V_i$, with a GCN trained on it. The attacker loss function is defined as:

$$\mathcal{L}^i_{\text{atk}} = -\mathcal{L}(V - V_i, \hat{C}_i), \tag{13}$$

where $\mathcal{L}$ is the cross-entropy loss.

With the attacker's loss, we compute the meta-gradients (gradients w.r.t. hyperparameters):

$$\nabla^i_G := \nabla_G \mathcal{L}^i_{\text{atk}}(f_{\theta^*}(G)) \text{ s.t. } \theta^* = \text{opt}_\theta(\mathcal{L}_i(f_\theta(G))), \tag{14}$$

where $\mathcal{L}_i$ is the cross-entropy loss on partition V_i. Similar to Metattack, the partition score function $S^i : V \times V \to \mathbb{R}$ on the i^{th} partition is defined as:

$$S^i(u, v) = \nabla^i_{u,v} \cdot (-2 \cdot A_{uv} + 1). \tag{15}$$

For each pair of nodes (u, v), we compute σ_{uv}, which is the standard deviation of its k partition scores. The greedy score for (u, v) is defined as:

$$S(u, v) = \begin{cases} \sum_{i=1}^{k} S^i(u, v), & \sigma_{uv} < \tilde{\sigma} \\ 0, & (\sigma_{uv} >= \tilde{\sigma}) \vee (u = v) \end{cases} \tag{16}$$

where $\tilde{\sigma}$ is the median of all σ_{uv}s. The definition eliminates all self-loops and ensures that the gradients of the chosen perturbation on the k partitions are not too diverse.

In each step, we greedily pick exactly one perturbation $e' = (u', v')$ with the highest score:

$$e' = \underset{e=(u,v)\in\phi(G)}{\arg\max}\ S(u, v), \tag{17}$$

where $e \in \phi(G)$ ensures that the modification does not conflict with the attack constraints. Then the algorithm updates G according to $e' = (u', v')$ by flipping the value of $A_{u'v'}$.

5.5 Algorithm

Algorithm 1 Black-Box Gradient Attack (BBGA)

Require: $G = (V, E)$, node features X, attack budget Δ, constraint $\phi(G)$, training iteration T, partition $\{V_1, \cdots, V_k\}$.
Ensure: Modified graph $G' = (V, E')$.

$C_p \leftarrow$ SpectralClustering(G, X).
$C_s \leftarrow$ training surrogate model with C_p and a random training set.
$A' \leftarrow A$
while $\|A' - A\|_0 < 2\Delta$ **do**
  for $i \in [1, k]$ **do**
    randomly initialize θ_0;
    $\theta_T \leftarrow$ update parameters for T iterations;
    $\nabla_G^i \leftarrow \nabla_G \mathcal{L}_{\text{atk}}^i(f_{\theta^T}(G))$;
    $S_i \leftarrow \nabla_G^i \odot (-2A' + 1)$;
  end for
  $S \leftarrow$ according to Eq. 16;
  $e' \leftarrow$ the index of the maximum element in S that are in $\phi(G)$;
  $A' \leftarrow$ flip edge e'.
end while
$G' \leftarrow$ use A' as the new adjacency matrix;
return: G'.

Following the detailed description, we now present the pseudo-code of the BBGA algorithm in Algorithm 1.

The meta-gradients of all the N^2 pairs of nodes will be computed in the algorithm, the computing of meta-gradients has T steps due to the chain rule and the meta-gradients are calculated for each of the k partitions. Thus, the computational complexity for the attacking procedure itself is bounded by $O(k \cdot T \cdot N^2)$. Considering the computational efforts needed to find the pseudo-labels (Yan et al., 2009), the overall computational complexity of BBGA is $O(k \cdot T \cdot N^2) + O(N^3)$. The $O(N^3)$ part is brought by the spectral clustering process as we apply the classic version of the spectral clustering algorithm in our proposed method. It's an open problem to improve efficiency of spectral clustering. A number of studies have been conducted to utilize spectral clustering on large-scale datasets (Liu et al., 2013; Cai and Chen, 2015; He et al., 2019; Yang et al., 2019, 2020; Song et al., 2021). Since spectral clustering is not the main focus of this work, we utilize the classic spectral clustering algorithm, which is $O(N^3)$, in our experiments. The application of fast spectral clustering algorithms in BBGA remains as a future work.

5.6 Experiments

We evaluate our proposed BBGA algorithm against both vanilla GCN and several defense methods. By conducting various experiments, we aim to answer the following empirical questions:

- **EQ1** How does BBGA works without accessing the training set of the model?
- **EQ2** Are the attacked graph as expected such that the perturbations are not distributed mainly around the training set?
- **EQ3** How do different components and hyperparameters affect the performance of BBGA?

5.6.1 Experimental settings

To demonstrate the effectiveness of our proposed BBGA method, we compare BBGA with three baselines: Random Attack, DICE, and Metattack. Since DICE and Metattack need label information, spectral clustering is also utilized to derive the pseudo labels. The defense algorithms used in the experiment include GCN, GCN-Jaccard, R-GCN, Pro-GNN, and GCN-SVD. We utilize scikit-learn (Pedregosa et al., 2011) for spectral clustering with parameter $\gamma = 0.001$. Following (Li et al., 2020) we set $\eta = 0.01$. For other hyperparameters, we set $k = 5$ and $T = 100$.

For each perturbation rate, we ran the experiments 10 times. To verify that our proposed BBGA algorithm is not engaged with any specific training set, we randomly altered the dataset splits with scikit-learn (Pedregosa et al., 2011) each time before the training of the defense algorithms. Following previous works, (Zügner and Günnemann, 2019; Jin et al., 2020) we choose accuracy rate as the primary evaluation metric.

5.6.2 Attack performance

To answer **EQ1**, we report the results when attacking against the original GCN in Table 6. The best results are highlighted in bold. As shown in the table, our proposed method outperforms the baselines in all situations when attacking against the original GCN model. Especially, BBGA outperforms the baselines by 20% in several experimental settings.

The performance when attacking against various defense methods is reported in Figs. 2, 3, and 4. As shown in the figures, our proposed BBGA algorithm achieves the best misclassification rates in most situations. The only exception is GCN-SVD. The experiments show that our proposed method is able to achieve higher misclassification rates when the training set is not accessible.

To further verify the effectiveness of the BBGA method, we employ both Friedman Test and Nemenyi Post-hoc Test (Zhou, 2016). The ranking distribution of the 4 attacking algorithms is reported in Table 7. Having $k = 4$ algorithms and $N = 60$ different experimental conditions, we have:

Table 6 Performance of attacking methods when attacking against the GCN (Misclassification rates±Std)

Dataset	Attack	20%	40%	60%	80%
Cora	BBGA	**23.90±1.28**	**28.98±1.69**	**33.48±2.01**	**37.12±2.09**
	DICE-BB	21.42± 1.16 (−10.37%)	25.69±1.19 (−10.71%)	29.43±1.31 (−12.10%)	33.14±1.52 (−10.72%)
	Random	21.34±1.74 (−10.71%)	25.77±1.19 (−11.08%)	28.43±1.71 (−15.08%)	32.73±1.76 (−11.83%)
	Metattack	22.58±1.19 (−5.23%)	26.82±1.57 (−7.45%)	28.88±1.26 (−13.74%)	32.71±1.51 (−11.88%)
Citeseer	BBGA	**29.44±1.22**	**32.17±1.15**	**35.69±1.24**	**39.46±2.01**
	DICE-BB	27.92±0.85 (−5.16%)	30.85±1.41 (−4.10%)	33.91±1.53 (−4.99%)	37.22±2.21 (−5.68%)
	Random	27.25±1.10 (−7.44%)	29.96±1.48 (−6.87%)	32.12±0.85 (−10.00%)	33.68±1.52 (−14.65%)
	Metattack	27.19±0.76 (−7.64%)	30.55±1.29 (−5.04%)	32.59±1.69 (−8.69%)	36.70±1.67 (−6.99%)
Cora-ML	BBGA	**30.29±3.03**	**45.10±3.58**	**52.69±5.26**	**60.36±4.96**
	DICE-BB	25.81±2.47 (−14.79%)	34.10±4.35 (−24.39%)	42.27±7.37 (−19.78%)	55.77±5.60 (−7.60%)
	Random	24.79±1.64 (−18.16%)	30.78±2.40 (−31.75%)	40.74±3.79 (−22.68%)	47.74±3.90 (−20.91%)
	Metattack	28.07±1.79 (−7.33%)	36.81±3.11 (−18.38%)	42.96±2.93 (−18.47%)	46.92±2.93 (−22.27%)

$$\tau_{\chi^2} = \frac{k-1}{k} \cdot \frac{12N}{k^2-1} \sum_{i=1}^{k} \left(r_i - \frac{k+1}{2} \right)^2 = 60.98,$$
$$\tau_F = \frac{(N-1)\tau_{\chi^2}}{N(k-1) - \tau_{\chi^2}} = 30.08, \tag{18}$$

which exceeds the critical value $c \approx 2.6556(\alpha = 0.05)$. Hence, according to the Friedman Test (Friedman, 1940), the performances of the 4 algorithms are significantly dissimilar. Using Nemenyi Post-hoc Test (Nemenyi, 1963), we have $q_\alpha \approx 2.5690(\alpha = 0.05, k = 4)$ and

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \approx 0.6055. \tag{19}$$

Since $|\frac{97}{60} - \frac{144}{60}| = \frac{47}{60} \approx 0.78 > 0.6055$, we reject the hypothesis that BBGA and Metattack have similar performance. However, as $|\frac{144}{60} - \frac{152}{60}| = \frac{8}{60} \approx 0.13 < 0.6055$, the performance of Metattack and DICE don't differ significantly. Hence, we conclude that in black-box environments, BBGA outperforms all the baselines significantly while Metattack doesn't show notable improvements over DICE.

5.6.3 Case study on perturbations

In this subsection, we answer the research question **EQ2** with a case study on the perturbations generated by BBGA. For a random split of the Cora dataset, we reveal distributions

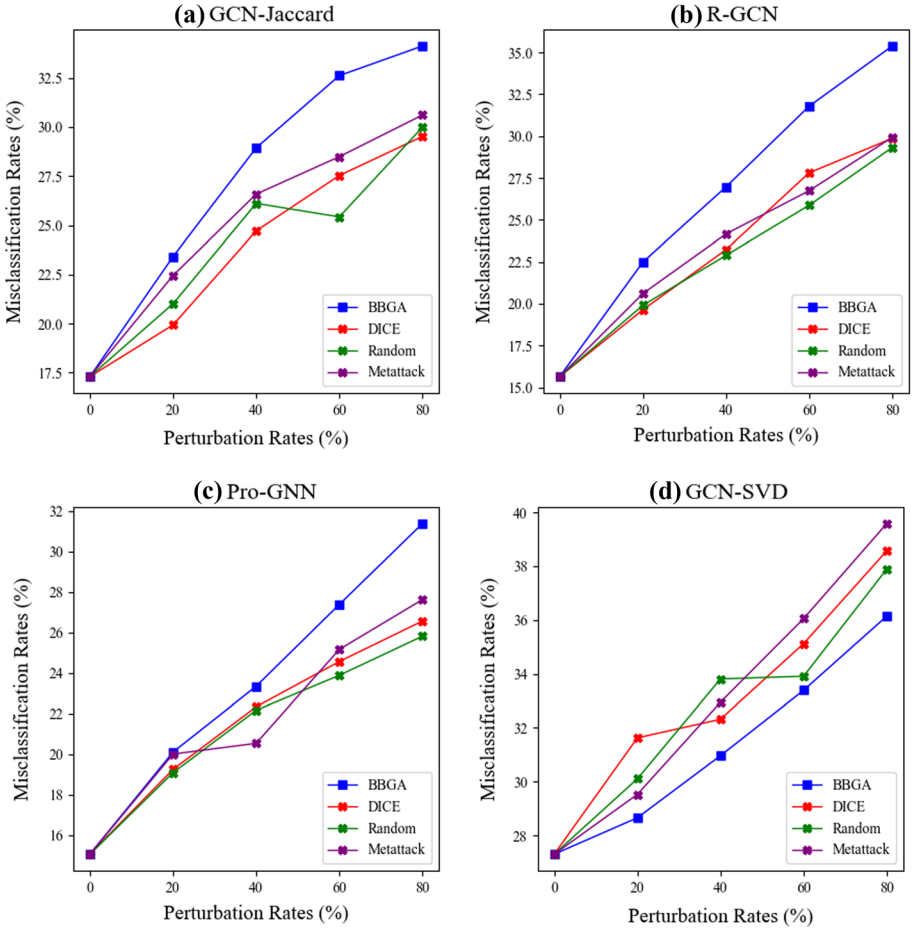

Fig. 2 Misclassification rates when attacking against defense algorithms on Cora: **a** GCN-Jaccard, **b** R-GCN, **c** Pro-GNN, **d** GCN-SVD

of BBGA's perturbations in Table 8. As illustrated in the table, the perturbations are not biased toward the training set as the proportion of Type 3 flips is around 80%. This demonstrates the evenness of the modifications of our proposed method. It explains the reason why our BBGA algorithm works without accessing the training set.

5.6.4 Ablation study and parameter analysis

To answer the research question **EQ3**, we conducted ablation studies and parameter analysis. For the ablation study, we created two variants of our method. BBGA-α removes the filter of low-variance node pairs in Eq. 16. BBGA-β removes the k-fold greedy choice procedure and in each step, exactly one partition is chosen randomly to compute the meta-gradient. We took Cora and the GCN model as an example. As revealed in Fig. 5a, BBGA has the best performance while BBGA-β performs the worst. This indicates that both

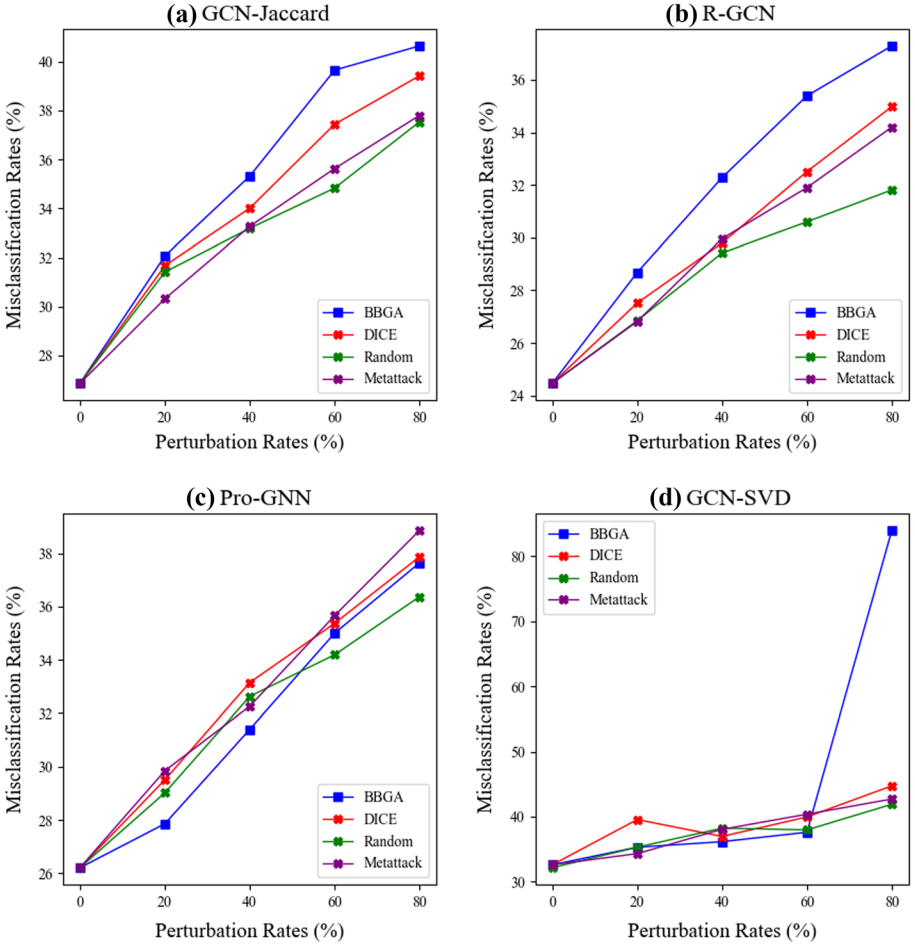

Fig. 3 Misclassification rates when attacking against defense algorithms on Citeseer: **a** GCN-Jaccard, **b** R-GCN, **c** Pro-GNN, **d** GCN-SVD

considering multiple partitions and filtering low-variance node pairs help in increasing the attack performance.

For parameter analysis, we varied the number of partitions k. Taking 80% perturbation rates on Cora as an example, it is revealed that the attack performance grows as k increases in general. However, since a larger k implies a longer training time, we suggest using the hyperparameter $k = 5$. The results of the parameter analysis are reported in Figure 5b.

6 Limitations and trade-offs

As revealed in Section V, the misclassification rate of GCNs can be significantly improved by the BBGA algorithm. However, a higher perturbation rate is generally required in black-box scenarios compared to gray-box ones. Also, another discovery is that BBGA is relatively ineffective when attacking GCN-SVD, which is a defense

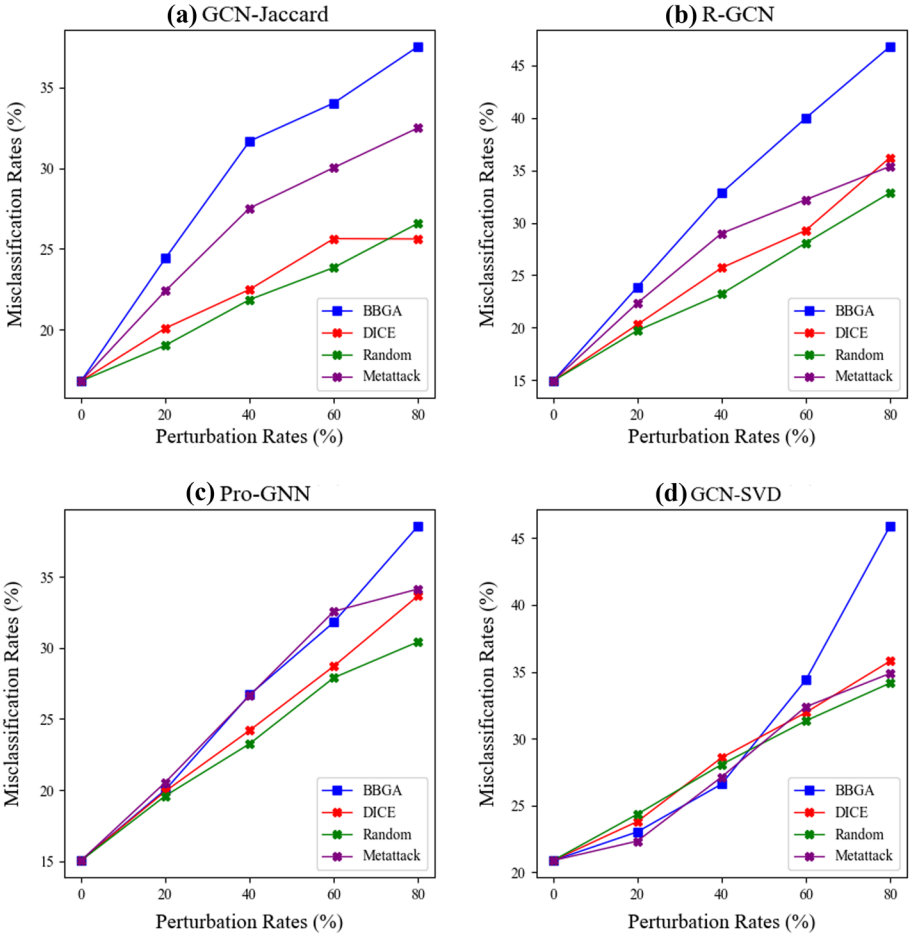

Fig. 4 Misclassification rates when attacking against defense algorithms on Cora-ML: **a** GCN-Jaccard, **b** R-GCN, **c** Pro-GNN, **d** GCN-SVD

Table 7 The ranking distribution of the 4 attacking methods

Algorithm	Rank 1	Rank 2	Rank 3	Rank 4	Avg. ranking
BBGA	45	2	4	9	97/60
DICE	4	26	24	6	152/60
Random	3	4	16	37	207/60
Metattack	8	28	16	8	144/60

algorithm utilizing low-rank approximation (Entezari et al., 2020). In this section, we analyze such discoveries and address the research question **RQ3**. Limitations are often reflected as trade-off points. For instance, in reinforcement learning, there is a trade-off between exploration and exploitation (Sutton and Barto, 2005). In the area of deep learning robustness, Hermann et al. discovered a trade-off between ImageNet top-1

Table 8 Distributions of perturbations by BBGA

Ptb Rate	Type 1 (%)	Type 2 (%)	Type 3 (%)
20%	1.02%	18.30%	80.68%
40%	1.07%	17.89%	81.05%
65%	1.00%	18.14%	80.86%
80%	1.04%	18.23%	80.73%

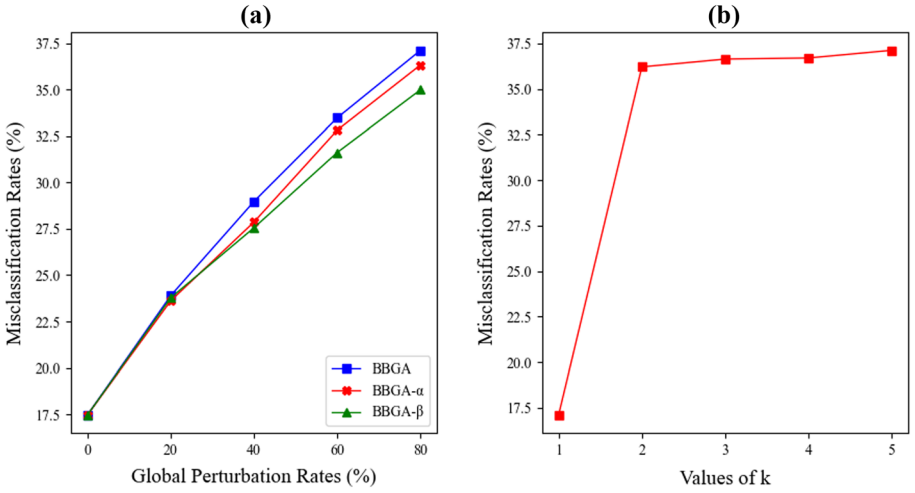

Fig. 5 Results of the ablation study and parameter analysis

accuracy and shape bias (Hermann et al., 2020). We conduct theoretical analysis in two trade-offs:

- **The robustness-budget trade-off** When the distribution shift mechanism is disallowed, the attacking algorithm is supposed to propagate adversarial information among the whole graph. In this situation, a relatively high attacking budget is expected.
- **The distribution-eigenvalue trade-off** Evenly-distributed attacks generally have lower influences on the leading singular values of the graph adjacency matrices. This explains why BBGA is relatively ineffective against GCN-SVD while significantly outperforming the baselines when attacking other models.

6.1 The robustness-budget trade-off

Since GNN models suffer from over-smoothing (Li et al., 2018), many graph neural networks, such as GCN (Kipf and Welling, 2017), GAT (Velickovic et al., 2018), and R-GCN (Zhu et al., 2019), contain only 2 layers. As a result, the impact range of an adversarial edge (i, j) is limited to 1-hop neighborhoods of nodes i and j. To discuss the trade-off mathematically, we define the following concepts at first.

Definition 2 *(influence)* In a 2-layer GNN, when adding an edge (i, j), all nodes adjacent to either v_i or v_j are **influenced**. Mathematically, $\forall k \in V$ s.t. $(k, i) \in E \vee (k, j) \in E$, k is influenced.

Definition 3 (the Node-Influence problem) An edge-adding sequence $\mathcal{E}$ of length k is a sequence $e_1, e_2, \cdots, e_k$ ($\forall i \in [1, k], e_i \in P$ where P is the set of allowed perturbations). Let $I_1, I_2, \cdots, I_k$ be the sets of nodes influenced by adding $e_1, e_2, \cdots, e_k$ in sequence. The Node-Influence optimization problem aims to find the shortest possible edge-adding sequence $\mathcal{E}_{\text{OPT}} = e_1, e_2, \cdots, e_k$ such that $\bigcup_{i=1}^{k} I_i = V$.

Based on Definition 2 and Definition 3, we aim to show that the length of the shortest possible edge-adding sequence is related to the Set-Cover problem. More specifically, we show that Node-Influence is equivalent to Set-Cover in terms of computational complexity via polynomial-time reducibility. For an in-depth review of reductions, we refer the readers to classic papers (Cook, 1971) and texts (Sipser, 2013; Goldreich, 2008) in theoretical computer science.

Theorem 2 *When the victim GCN model has only* 2 *layers, and the attacker is only allowed to add edges, the Node-Influence problem is equivalent to the Set-Cover problem.*

Proof We show the equivalence by constructing polynomial-time reductions.

Part 1 Node-Influence **is polynomial-time reducible to** Set-Cover.

Let $\mathcal{U} = V$ and k be the total number of allowed flips. Consider the i^{th} possible perturbation (s_i, t_i) such that $s_i, t_i \in V$. Let $S_i = \mathcal{N}_{s_i} \cup \mathcal{N}_{t_i}$ where $\mathcal{N}_{s_i} = \{v \mid v \in V \wedge A_{s_i v} = 1\} \cup \{s_i\}$ and $\mathcal{N}_{t_i} = \{v \mid v \in V \wedge A_{t_i v} = 1\} \cup \{t_i\}$. The collection of sets is constructed as $\mathcal{S} = \{S_1, \cdots, S_k\}$.

We inductively show that when $e_i = (s_i, t_i) \in \mathcal{E}$ is added during the attacking procedure, the set of newly influenced nodes $\bar{I}_i = I_i - \bigcup_{j=1}^{i-1} I_j$ is always a subset of S_i.

Induction basis This obviously holds for the first step since the $\mathcal{S}$ is constructed on the original graph and by the definition S_1 and $\bar{I}_1$, $S_1 = \bar{I}_1$.

Inductive step When adding the i^{th} adversarial edge $e_i = (s_i, t_i)$, there are two possible situations:

(1) If no edge incident to s_i or t_i has been added, adding $e_i = (s_i, t_i)$ will only influence their 1-hop neighbors since the GCN has 2 layers. Hence by definitions we have $I_i = S_i$. Thus, $\bar{I}_i = I_i - \bigcup_{j=1}^{i-1} I_j \subseteq S_i$.

(2) Without the loss of generality, if any edge $e_k = (s_i, j)$, which is incident to s_i, has been added ($(s_i, j) \in \{e_1, \cdots e_{i-1}\}$), then adding (s_i, t_i) will also influence j. In this situation, we have $j \in I_i$ but $j \notin S_i$. However, in this situation we must have $j \in I_k (k \in [1, i-1])$ since it has been covered by a previous set when (s_i, j) is added. Hence, we must have $\bar{I} = I_i - \bigcup_{j=1}^{i=1} I_j \subseteq S_i$.

Inductive conclusion Thus, Node-Influence is reducible to Set-Cover.

When constructing $\mathcal{S}$, the algorithm needs to traverse all the $N \times N$ pairs of nodes in the graph. For each node, the algorithm needs linear time to visit its neighbors. Hence, the overall time complexity for the reduction is $O(N^3)$ and it's a polynomial-time reduction.

Part 2 Set-Cover **is polynomial-time reducible to** Node-Influence.

We prove it by constructing the reduction algorithm directly. For a Set-Cover problem with n elements in the universe $\mathcal{U}$ and k sets in $\mathcal{S}$, we construct n element-level nodes $v_{e1}, v_{e2}, \cdots, v_{en}$ and k set-level nodes $v_{s1}, v_{s2}, \cdots, v_{sk}$ corresponding to the sets in $\mathcal{S}$. Then we add a starting node v_0. In the graph, each set-level node v_{si} is adjacent to its elements. Also, every pair of set-level nodes is connected to each other. Then we mark all node pairs $(v_0, v_{si}), i \in [1, k]$ as allowed perturbations. The pseudocode of the construction algorithm is reported in Algorithm 2.

When the first perturbation (v_0, v_{sa}) is selected, all set-level nodes are influenced. Also, since v_{sa} is adjacent to the nodes representing elements in S_a, all elements in S_a are influenced. After the first perturbation, each allowed perturbation adds a new edge v_{si} and influences the nodes representing elements in S_i. Hence, a set of perturbations that influences the whole graph exactly corresponds to a set cover of the universe $\mathcal{U}$. Thus, Set-Cover can be reduced to Node-Influence.

The reduction algorithm traverses all elements in the Set-Cover problem in $O(n)$ time. For each set in the Set-Cover problem, its elements are traversed in $O(nk)$ time. Finally, for pairs of sets, edges are constructed in $O(k^2)$ time. Hence, the overall time complexity for the reduction algorithm is $O(n + nk + k^2)$, which is polynomial.

Since the two problems can be reduced to each other, they are equivalent. □

Algorithm 2 The Construction Algorithm

Require: $\mathcal{U}, \mathcal{S}$.
Ensure: $G = (V, E)$, set of allowed perturbations P.

for $e_i \in \mathcal{U}$ **do**
  Construct $v_{ei} \in V$
end for
for $S_i \in \mathcal{S}$ **do**
  Construct $v_{si} \in V$
  for $e_i \in S_i$ **do**
    Construct $(v_{ei}, v_{si}) \in E$
  end for
end for
Construct v_0
for $v_{si} \in \{v_{s1}, \cdots, v_{sk}\}$ **do**
  Add (v_0, v_{si}) to P
  for $v_{sj} \in \{v_{s1}, \cdots, v_{sk}\} - \{v_{si}\}$ **do**
    Construct $(v_{si}, v_{sj}) \in E$
  end for
end for
return: $G = (V, E), P$.

Algorithm 3 The Greedy Approximation Algorithm

Require: A (graph adjacency matrix), r (maximum degree for a node to be attackable), N (the number of nodes), V (vertex set).

Ensure: p (approximate number of perturbations needed to influence the whole graph).

$\mathcal{S} \leftarrow \emptyset$;
for $i \in [0, N-1)$ **do**
  for $j \in [i+1, N)$ **do**
    if Both v_i and v_j have degrees no more than r. **then**
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathcal{N}_{v_i} \cup \mathcal{N}_{v_j}\}$;
    end if
  end for
end for
$p \leftarrow 0$;
while $\exists v \in V$ s.t. $\deg(v) \leq r$ **do**
  Select $S_i \in \mathcal{S}$ which maximizes $\mid S_i \cap V \mid$;
  $V \leftarrow V - S_i$;
  $p \leftarrow p + 1$;
end while
return: p.

Inspired by Theorem 2, the minimum number of perturbations needed to cover the whole graph could be estimated by solving the Set-Cover problem. Although being NP-Hard, the Set-Cover problem could be approximated with a greedy algorithm, which selects the set that contains the most uncovered elements in each step. When the maximum size of the sets in $\mathcal{S}$ is $d = \max_{i \in [1,k]} |S_i|$, the approximation ratio of the greedy algorithm is bounded by the harmonic function $H(d) = \sum_{i=1}^{d} \frac{1}{i}$ (Kleinberg and Tardos, 2005).

We approximate the minimum number of perturbations needed with Algorithm 3. Datasets used in this study include Cora, Citeseer, and Cora-ML. In reality, it is believed that it's not realistic to perturb nodes with large degrees (Ma et al., 2020). High-degree nodes are also reported to be less vulnerable (Wang et al., 2019). Hence, we introduce a limit r on nodes such that any node whose degree is larger than r is ignored in the algorithm. This not only makes the approximation more realistic, but also improves the approximation bound. In the experiments, we choose $r = 5$. Thus the largest possible cardinality of sets in $\mathcal{S}$ is $d = 12$. The approximation ratio is bounded by:

$$H(12) = \sum_{i=1}^{12} \frac{1}{i} \approx 3.10. \tag{20}$$

The approximated numbers of perturbations are reported in Table 9. We report both the approximated and minimum possible number of perturbations.

- p: The approximated numbers of perturbations returned by Algorithm 3.
- $p/\mid E \mid$: The perturbation ratios needed to achieve p perturbations.
- p/3.10: The estimated minimum numbers of perturbations estimated by Equation (20). The values are rounded to the nearest integers.

Table 9 Approximated number of perturbations

Dataset	p	$p/\mid E \mid$	$p/3.10$	$p/3.10 \mid E \mid$
Cora	702	13.85%	226	4.46%
Citeseer	604	16.47%	195	5.32%
Cora-ML	1006	12.60%	325	4.07%

- $p/3.10 \mid E \mid$: The estimated minimum perturbation ratio needed to influence the whole graph.

As revealed in Table 9, a relatively high proportion of perturbations is needed for sparse graphs. Since Set-Cover is NP-Hard and several works such as Lund and Yannakakis (1994) and Dinur and Steurer (2014) have addressed its hardness of approximation theoretically, it is not likely to reach the minimum perturbation ratio. Also, in the reality, it's not enough to make GCN misclassify a node simply by covering it with any arbitrary adversarial perturbation. Hence, a relatively high perturbation ratio is expected when solely relying on the message propagation mechanism.

6.2 The distribution-eigenvalue trade-off

In this subsection, we show the existence of the distribution-eigenvalue trade-off under two assumptions. Consider an evenly distributed attack and a biased attack. The even attack is similar to Random Attack in the distributions of flips while the biased attack has a much higher proportion of type 1 and type 2 flips. Let A' be the attacked adjacency matrix and we denote $\Delta A = A' - A$ as the attack matrix. We assume that the only difference between the two methods is the evenness and both methods flip ζ edges, i.e. the attacking budget is ζ.

The attack matrix of the even attack is denoted as E while the other is denoted as U. Noticing that the squares of the singular values of E and U are exactly the eigenvalues of $E^T E$ and $U^T U$, we denote $F \triangleq E^T E, W \triangleq U^T U$. The eigenvalues of F and W are denoted as $\lambda_{f1}, \cdots, \lambda_{fN}$ and $\lambda_{w1}, \cdots, \lambda_{wN}$ respectively. For the simplicity of discussion, we focus on the eigenvalues of F and W instead of the singular values of E and U in the following proofs.

Based on previous findings that attacking algorithms tend to add edges instead of deleting ones (Wu et al., 2019), we assume that both attacks are not allowed to delete edges.

Assumption 1 $\forall i, j \in [1, N]$, we have $E_{ij} \geq 0 \wedge U_{ij} \geq 0$.

When an edge (i, j) is added by the algorithm, a perturbation is *incident* to both node i and node j. $F_{ii} = \sum_{j=1}^{N} E_{ij} E_{ji}$ and $W_{ii} = \sum_{j=1}^{n} U_{ij} U_{ji}$ are the numbers of perturbations incident to node i under the two attacks respectively.

When adversarial edges (i, j) and (i, k) are added, nodes j and k have a *common incident*. For $i \neq j$, $F_{ij} = \sum_{k=1}^{N} E_{ik} E_{kj}$ and $W_{ij} = \sum_{k=1}^{N} U_{ik} U_{kj}$ specify the number of their common incidents under the two attacks respectively.

We assume that under the even attack, each node is more likely to be incident to the same number of perturbations. Similarly, each pair of nodes is more likely to have the same number of common incidents. This assumption is formally defined as:

Assumption 2 Let $F_{\text{diag}} = \{F_{ii} \mid i \in [1,N]\}, W_{\text{diag}} = \{W_{ii} \mid i \in [1,N]\}, F_{\text{co}} = \{F_{ij} \mid i,j \in [1,N] \wedge i \neq j\}$ and $W_{\text{co}} = \{W_{ij} \mid i,j \in [1,N] \wedge i \neq j\}$, we have:

$$\text{Var}(F_{\text{diag}}) < \text{Var}(W_{\text{diag}}), \tag{21}$$

$$\text{Var}(F_{\text{co}}) < \text{Var}(W_{\text{co}}). \tag{22}$$

Based on the assumptions, the trade-off is mathematically stated as:

Theorem 3 (The Distribution-Eigenvalue Trade-off) *For the eigenvalues of $F = E^T E$ and $W = U^T U$, we have $\sum_{i=1}^{n} \lambda_{fi} = \sum_{i=1}^{n} \lambda_{wi}$ and $\text{Var}(\lambda_f) < \text{Var}(\lambda_w)$.*

To prove the theorem, we state and prove several lemmas at first.

Lemma 1 *For two sets of numbers $X = \{x_1, \cdots, x_n\}$ and $Y = \{y_1, \cdots, y_n\}$, if $\sum_{i=1}^{n} x_i \leq \sum_{i=1}^{n} y_i$ and $\text{Var}(X) < \text{Var}(Y)$, we have $\sum_{i=1}^{n} x_i^2 < \sum_{i=1}^{n} y_i^2$.*

Proof Notice that $\text{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$,we have:

$$\begin{aligned} \sum_{i=1}^{n} x_i^2 &= n\mathbb{E}[X^2] = n\text{Var}(X) + n(\mathbb{E}[X])^2 \\ \sum_{i=1}^{n} y_i^2 &= n\mathbb{E}[Y^2] = n\text{Var}(Y) + n(\mathbb{E}[Y])^2 \end{aligned}. \tag{23}$$

Since $\sum_{i=1}^{n} x_i \leq \sum_{i=1}^{n} y_i$, $\mathbb{E}[X] \leq \mathbb{E}[Y]$. As $\text{Var}(X) < \text{Var}(Y)$, we have $n\text{Var}(X) + n(\mathbb{E}[X])^2 < n\text{Var}(Y) + n(\mathbb{E}[Y])^2$. Hence $\sum_{i=1}^{n} x_i^2 < \sum_{i=1}^{n} y_i^2$. □

Lemma 2 $\lambda_{f1} + \cdots + \lambda_{fN} = \lambda_{w1} + \cdots + \lambda_{wN}$.

Proof For all $i \in [1,N]$, the values of F_{ii} and W_{ii} are the number of perturbations incident to node i under the two attacks respectively. Since both attacks have the same budget ζ, we have $\text{tr}(F) = \text{tr}(W) = 2\zeta$. Noticing that $\text{tr}(F) = \lambda_{f1} + \cdots + \lambda_{fN}, \text{tr}(W) = \lambda_{w1} + \cdots + \lambda_{wN}$, we have $\lambda_{f1} + \cdots + \lambda_{fN} = \lambda_{w1} + \cdots + \lambda_{wN}$. □

Now we consider the elements in F and W. Let $\tilde{F}$ and $\tilde{W}$ be the squared sum of their diagonal elements, $\hat{F}$ and $\hat{W}$ be the squared sum of their non-diagonal elements. Formally, $\tilde{F} = \sum_{i=1}^{N} F_{ii}^2, \tilde{W} = \sum_{i=1}^{N} W_{ii}^2$ and $\hat{F} = \sum_{i \neq j} F_{ij}^2, \hat{W} = \sum_{i \neq j} W_{ij}^2$.

Lemma 3 $\tilde{F} < \tilde{W}$.

Proof In the proof of Lemma 2, we have shown that $\mathrm{tr}(F) = \mathrm{tr}(W)$. Hence $\sum_{i=1}^{N} F_{ii} = \sum_{i=1}^{N} W_{ii}$. According to Assumption 2, $\mathrm{Var}(F_{ii}) < \mathrm{Var}(W_{ii})$. By Lemma 1, $\tilde{F} = \sum_{i=1}^{N} F_{ii}^2 < \sum_{i=1}^{N} W_{ii}^2 = \tilde{W}$. □

Lemma 4 $\hat{F} < \hat{W}$.

Proof F_{ij} is a combinatorial counting of the number of common incidents between i and j under the even attack. $\sum_{i \neq j} F_{ij}$ is twice the total number of all common incidents caused by the attack. From another perspective, an arbitrary node i is incident to F_{ii} adversarial edges. Each pair of adversarial edges incident to node i is exactly one distinct common incident and node i contributes to $\binom{F_{ii}}{2}$ common incidents. Hence, the total number of common occurrences is $\sum_{i=1}^{N} \binom{F_{ii}}{2}$. By a combinatorial argument, we have:

$$\begin{aligned} \sum_{i \neq j} F_{ij} &= 2 \times \sum_{i=1}^{N} \binom{F_{ii}}{2} \\ &= \sum_{i=1}^{N} 2 \times \frac{F_{ii}(F_{ii}-1)}{2} \\ &= \sum_{i=1}^{N} (F_{ii}^2 - F_{ii}) \\ &= \sum_{i=1}^{N} F_{ii}^2 - \sum_{i=1}^{N} F_{ii} \end{aligned} . \tag{24}$$

Similarly, we have:

$$\sum_{i \neq j} W_{ij} = \sum_{i=1}^{N} W_{ii}^2 - \sum_{i=1}^{N} W_{ii}. \tag{25}$$

According to Lemma 3, $\sum_{i=1}^{N} F_{ii}^2 < \sum_{i=1}^{N} W_{ii}^2$. Noticing that $\sum_{i=1}^{N} F_{ii} = \sum_{i=1}^{N} W_{ii}$, we have the following inequality:

$$\sum_{i \neq j} F_{ij} < \sum_{i \neq j} W_{ij}. \tag{26}$$

According to Assumption 2, $\mathrm{Var}(F_{ij}) < \mathrm{Var}(W_{ij})$. Thus, by Lemma 1, we immediately have:

$$\hat{F} = \sum_{i \neq j} F_{ij}^2 < \sum_{i \neq j} W_{ij}^2 = \hat{W}. \tag{27}$$

□

Now, we prove the existence of the evenness-eigenvalue trade-off.

Proof of Theorem 3 $\sum_{i=1}^{n} \lambda_{fi} = \sum_{i=1}^{n} \lambda_{wi}$ is proven by Lemma 2.

We now consider the variances. Since E and U are symmetric, F and W are also symmetric. Hence $F^2 = F^T F$, $W^2 = W^T W$. We have:

$$\begin{aligned} \mathrm{tr}(F^T F) = \mathrm{tr}(F^2) = \|F\|_F^2 = \tilde{F} + \hat{F} \\ \mathrm{tr}(W^T W) = \mathrm{tr}(W^2) = \|W\|_F^2 = \tilde{W} + \hat{W} \end{aligned}. \tag{28}$$

Noticing that for any matrix M and its eigenvalue λ, λ^2 is an eigenvalue of M^2. Thus we have:

$$\begin{aligned} \mathrm{tr}(F^2) = \lambda_{f1}^2 + \cdots + \lambda_{fN}^2 \\ \mathrm{tr}(W^2) = \lambda_{w1}^2 + \cdots + \lambda_{wN}^2 \end{aligned}. \tag{29}$$

By Lemma 3 and Lemma 4, we have:

$$\tilde{F} + \hat{F} < \tilde{W} + \hat{W}. \tag{30}$$

Combining Equations (28), (29) and (30), we have:

$$\lambda_{f1}^2 + \cdots + \lambda_{fN}^2 < \lambda_{w1}^2 + \cdots + \lambda_{wN}^2. \tag{31}$$

Hence $\mathbb{E}[\lambda_f^2] = \frac{1}{n}(\sum_{i=1}^N \lambda_{fi}^2) = \frac{1}{n}(\sum_{i=1}^N \lambda_{wi}^2) < \mathbb{E}[\lambda_w^2]$. By Lemma 2, $\mathbb{E}[\lambda_f] = \frac{1}{n}(\sum_{i=1}^N \lambda_{fi}) = \frac{1}{n}(\sum_{i=1}^N \lambda_{wi}) = \mathbb{E}[\lambda_w]$. Thus we have:

$$\begin{aligned} \mathrm{Var}(\lambda_f) &= \mathbb{E}[\lambda_f^2] - (\mathbb{E}[\lambda_f])^2 \\ &< \mathbb{E}[\lambda_w^2] - (\mathbb{E}[\lambda_w])^2 \\ &= \mathrm{Var}(\lambda_w). \end{aligned} \tag{32}$$

□

Hence, we proved the existence of the distribution-eigenvalue trade-off under Assumption 1 and Assumption 2.

Theorem 4 (Weyl's inequality for singular values (Horn and Johnson, 1991)) *For $m \times n$ matrices A, B, we have $\sigma_j(A) + \sigma_k(B) \geq \sigma_{j+k-1}(A + B)$, in which $\sigma_i(A)$ denotes the i^{th} largest singular value of matrix A.*

Theorem 3 shows that even attacks tend to have smaller leading singular values of the attack matrix. According to the Weyl's inequality (Theorem 4), the uneven attack will have a stronger attack ability since it will have a higher upper bound on the leading singular values of the perturbed adjacent matrix. Since GCN-SVD is based on low-rank approximation such that small singular values are discarded, this explains the experimental results that the BBGA algorithm is relatively ineffective against GCN-SVD.

7 Conclusions

In this work, we develop deeper insights into graph structure attacks and defenses. We identify an internal mechanism of the existing representative graph structure attacking algorithms via various case studies. Revealing the advantages and drawbacks of the distribution

shift mechanism, we propose the novel Black-Box Gradient Attack (BBGA) algorithm. To the best of our knowledge, this is the first gradient-based black-box graph attack and the first non-targeted structure attack that doesn't require any inquiry. Inspired by the experimental result, we also conduct theoretical analysis to reveal two trade-off points in graph structure attacks.

Robustness is critical to the real-world applications of GNNs. This work provides insights for further studies in both graph adversarial attacks and graph defenses. At first, the theoretical properties found in our work could be utilized by both attacking methods and defense algorithms. Secondly, this work primarily focuses on small datasets. To apply BBGA on large-scale graphs, we aim to investigate the applications of fast spectral clustering methods in our proposed method. Finally, it will be very meaningful to investigate black-box adversarial attacks against certificate robustness methods as well as adversarial attacks on heterogeneous graphs.

Author Contributions XP motivated this work. Both authors contributed the ideas and HZ conducted the experiments. The article is drafted by HZ and edited by both authors together.

Funding No funds, grants, or other support were received for conducting this study.

Data Availability The datasets used in this paper are publicly available at https://github.com/danielzuegner/nettack/tree/master/data and https://github.com/klicperajo/ppnp/tree/master/ppnp/data

Code Availability The codes of BBGA will be open-sourced after the publication of this paper.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethics Approval Not applicable.

Consent to Participate Not applicable.

Consent for Publication Not applicable.

References

Athalye, A., Carlini, N., & Wagner, D.A. (2018). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, pp. 274–283.

Bruna, J., Zaremba, W., Szlam, A.D., & LeCun, Y. (2014). Spectral networks and locally connected networks on graphs. CoRR **abs/1312.6203**.

Cai, D., & Chen, X. (2015). Large scale spectral clustering via landmark-based sparse representation. *IEEE Transactions on Cybernetics, 45,* 1669–1680.

Carlini, N., & Wagner, D.A. (2017). Adversarial examples are not easily detected: Bypassing ten detection methods. In Thuraisingham, B.M., Biggio, B., Freeman, D.M., Miller, B., Sinha, A. (eds.) Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, (pp. 3–14).

Chang, H., Rong, Y., Xu, T., Huang, W., Zhang, H., Cui, P., Zhu, W., & Huang, J. (2019). The general black-box attack method for graph neural networks. arXiv:abs/1908.01297.

Chen, J.J., Ma, T., & Xiao, C. (2018). Fastgcn: Fast learning with graph convolutional networks via importance sampling. arXiv:abs/1801.10247.

Chen, J., Wu, Y., Xu, X., Chen, Y., Zheng, H., & Xuan, Q. (2018). Fast gradient attack on network embedding. arXiv:abs/1809.02797

Chen, Y.-C., Lu, P.-E., Chang, C.-S., & Liu, T.-H. (2020). A time-dependent sir model for covid-19 with undetectable infected persons. *IEEE Transactions on Network Science and Engineering, 7*(4), 3279–3294.

Cook, S.A. (1971). The complexity of theorem-proving procedures. Proceedings of the third annual ACM symposium on Theory of computing.

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., & Song, L. (2018). Adversarial attack on graph structured data. In Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, (pp. 1123–1132).

Dai, J., Zhu, W., & Luo, X. (2022). A targeted universal attack on graph convolutional network. arXiv:abs/2011.14365

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, (pp. 3837–3845).

Dinur, I., & Steurer, D. (2014). Analytical approach to parallel repetition. In Shmoys, D.B. (ed.) Symposium on Theory of Computing, STOC 2014, (pp. 624–633).

Dong, H., Chen, J., Feng, F., He, X., Bi, S., Ding, Z., & Cui, P. (2021). On the equivalence of decoupled graph convolution network and label propagation. In Leskovec, J., Grobelnik, M., Najork, M., Tang, J., Zia, L. (eds.) WWW '21: The Web Conference 2021, (pp. 3651–3662).

Entezari, N., Al-Sayouri, S.A., Darvishzadeh, A., & Papalexakis, E.E. (2020). All you need is low (rank): Defending against adversarial attacks on graphs. In Caverlee, J., Hu, X.B., Lalmas, M., Wang, W. (eds.) WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, (pp. 169–177).

Finkelshtein, B., Baskin, C., Zheltonozhskii, E., & Alon, U. (2020). Single-node attack for fooling graph neural networks. arXiv:abs/2011.03574.

Fricke, S. (2018). Semantic scholar. *Journal of the Medical Library Association : JMLA, 106,* 145–147.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics, 11,* 86–92.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F., & Brendel, W. (2019). Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. arXiv:abs/1811.12231.

Geisler, S., Schmidt, T., Şirin, H., Zügner, D., Bojchevski, A., & Günnemann, S. (2021). Robustness of graph neural networks at scale. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, & J. W. Vaughan (Eds.), *Advances in Neural Information Processing Systems* (Vol. 34, pp. 7637–7649). Red Hook, NY, USA: Curran Associates Inc.

Geisler, S., Zügner, D., & Günnemann, S. (2020). Reliable graph neural networks via robust aggregation. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems* (Vol. 33, pp. 13272–13284). Red Hook, NY, USA: Curran Associates Inc.

Goldreich, O. (2008). *Computational Complexity: A Conceptual Perspective*. Cambridge: Cambridge University Press.

Goodfellow, I.J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. CoRR arxiv:abs/1412.6572.

Günnemann, S. (2022). In Wu, L., Cui, P., Pei, J., Zhao, L. (eds.) Graph Neural Networks: Adversarial Robustness, (pp. 149–176). Springer, Singapore.

Guo, J., Li, S., Zhao, Y., & Zhang, Y. (2022). Learning robust representation through graph adversarial contrastive learning. DASFAA 2022, Virtual Event, April 11–14, 2022, Proceedings, Part I*Database Systems for Advanced Applications: 27th International Conference* (pp. 682–697). Berlin, Heidelberg: Springer.

Hamilton, W.L., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, pp. 1024–1034

He, L., Ray, N., Guan, Y., & Zhang, H. (2019). Fast large-scale spectral clustering via explicit feature mapping. *IEEE Transactions on Cybernetics, 49,* 1058–1071.

Hermann, K.L., Chen, T., & Kornblith, S. (2020). The origins and prevalence of texture bias in convolutional neural networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, (pp. 19000–19015).

Horn, R. A., & Johnson, C. R. (1991). *Topics in Matrix Analysis*. New York: Cambridge University Press.

Jin, W., Derr, T., Wang, Y., Ma, Y., Liu, Z., & Tang, J. (2021). Node similarity preserving graph convolutional networks. In Lewin-Eytan, L., Carmel, D., Yom-Tov, E., Agichtein, E., Gabrilovich, E. (eds.)

WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, (pp. 148–156).
Jin, W., Li, Y., Xu, H., Wang, Y., & Tang, J. (2020). Adversarial attacks and defenses on graphs: A review and empirical study. arXiv:abs/2003.00653.
Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., & Tang, J. (2020). Graph structure learning for robust graph neural networks. In Gupta, R., Liu, Y., Tang, J., Prakash, B.A. (eds.) KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, (pp. 66–74).
Jin, H., Shi, Z., Peruri, V. J. S. A., & Zhang, X. (2020). Certified robustness of graph convolution networks for graph classification under topological attacks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems* (Vol. 33, pp. 8463–8474). Red Hook, NY, USA: Curran Associates Inc.
Kipf, T., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. arXiv:abs/1609.02907.
Kleinberg, J., & Tardos, É. (2005). *Algorithm design*. Boston: Pearson.
Klicpera, J., Bojchevski, A., & Günnemann, S. (2019). Predict then propagate: Graph neural networks meet personalized pagerank. In ICLR.
Li, Q., Han, Z., & Wu, X. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. In McIlraith, S.A., Weinberger, K.Q. (eds.) Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), (pp. 3538–3545).
Li, Y., Jin, W., Xu, H., & Tang, J. (2020). Deeprobust: A pytorch library for adversarial attacks and defenses. arXiv:abs/2005.06149.
Liu, Z., & Zhou, J. (2020). Introduction to graph neural networks. In Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, San Rafael.
Liu, J., Wang, C., Danilevsky, M., & Han, J. (2013). Large-scale spectral clustering on graphs. In IJCAI.
Liu, S., Li, G., Tran, T., & Jiang, Y. (2016). Preference relation-based Markov random fields for recommender systems. *Machine Learning, 106,* 523–546.
Lund, C., & Yannakakis, M. (1994). On the hardness of approximating minimization problems. *Journal of ACM, 41,* 960–981.
Ma, J., Deng, J., & Mei, Q. (2022). Adversarial attack on graph neural networks as an influence maximization problem. WSDM '22, (pp. 675–685). Association for Computing Machinery, New York, NY, USA.
Ma, J., Ding, S., & Mei, Q. (2020). Towards more practical adversarial attacks on graph neural networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020.
Ma, Y., Wang, S., Wu, L., & Tang, J. (2019). Attacking graph convolutional networks via rewiring. arXiv:abs/1906.03750.
Ma, Y., & Tang, J. (2020). *Deep Learning on Graphs*. Cambridge: Cambridge University Press.
Nemenyi, P. B. (1963). *Distribution-free Multiple Comparisons*. Princeton: Princeton University.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research, 12,* 2825–2830.
Ron, D., & Shamir, A. (2013). Quantitative analysis of the full bitcoin transaction graph. Lecture Notes in Computer Science. In A. Sadeghi (Ed.), *Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1–5, 2013, Revised Selected Papers* (Vol. 7859, pp. 6–24). Heidelberg: Springer.
Rossetti, G., Pappalardo, L., Pedreschi, D., & Giannotti, F. (2016). Tiles: An online algorithm for community discovery in dynamic social networks. *Machine Learning, 106,* 1213–1241.
Schuchardt, J., Bojchevski, A., Gasteiger, J., & Günnemann, S. (2021). Collective robustness certificates: Exploiting interdependence in graph neural networks. In: International Conference on Learning Representations.
Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding Machine Learning—From Theory to Algorithms*. Cambridge: Cambridge University Press.
Sipser, M. (2013). *Introduction to the Theory of Computation* (3rd ed.). Boston: Cengage Learning.
Slavík, P. (1997). A tight analysis of the greedy algorithm for set cover. *Journal of Algorithms, 25,* 237–254.
Song, K., Yao, X., Nie, F., Li, X., & Xu, M. (2021). Weighted bilateral k-means algorithm for fast co-clustering and fast spectral clustering. *Pattern Recognition, 109,* 107560.
Sun, L., Dou, Y., Yang, C., Wang, J., Yu, P.S., & Li, B. (2018). Adversarial attack and defense on graph data: A survey. arXiv preprint arXiv:1812.10528.
Sun, Y., Wang, S., Tang, X., Hsieh, T.-Y., & Honavar, V. (2020). Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In Huang, Y., King, I., Liu, T.-Y., van

Steen, M. (eds.) Proceedings of The Web Conference 2020, (pp. 673–683). Association for Computing Machinery, New York, NY, USA.

Sutton, R., & Barto, A. (2005). Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks, 16,* 285–286.

Tang, X., Li, Y., Sun, Y., Yao, H., Mitra, P., & Wang, S. (2020). Transferring robustness for graph neural network against poisoning attacks. In: Caverlee, J., Hu, X.B., Lalmas, M., Wang, W. (eds.) WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, (pp. 600–608).

Vazirani, V. V. (2013). *Approximation Algorithms*. Heidelberg: Springer.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. arXiv:abs/1710.10903.

Wang, X., Liu, X., & Hsieh, C.-J. (2019). Graphdefense: Towards robust graph convolutional networks. arXiv:abs/1911.04429.

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

Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., & Zhu, L. (2019). Adversarial examples for graph data: Deep insights into attack and defense. In Kraus, S. (ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, (pp. 4816–4823).

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems, 32,* 4–24.

Xu, K., Chen, H., Liu, S., Chen, P., Weng, T., Hong, M., & Lin, X. (2019). Topology attack and defense for graph neural networks: An optimization perspective. In Kraus, S. (ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, (pp. 3961–3967).

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., & Jegelka, S. (2018). Representation learning on graphs with jumping knowledge networks. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, pp. 5449–5458.

Yan, D., Huang, L., & Jordan, M.I. (2009). Fast approximate spectral clustering. In IV, J.F.E., Fogelman-Soulié, F., Flach, P.A., Zaki, M.J. (eds.) Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (pp. 907–916).

Yang, L., Liu, X., Nie, F., & Liu, M. (2019). Large-scale spectral clustering based on representative points. *Mathematical Problems in Engineering, 2019,* 1–7.

Yang, X., Yu, W., Wang, R., Zhang, G., & Nie, F. (2020). Fast spectral clustering learning with hierarchical bipartite graph for large-scale data. *Pattern Recognition Letters, 130,* 345–352.

Yuan, X., He, P., Zhu, Q., & Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems, 30,* 2805–2824.

Zhang, Z., Cui, P., & Zhu, W. (2018). Deep learning on graphs: A survey. arXiv:abs/1812.04202.

Zhang, M., Hu, L., Shi, C., & Wang, X. (2020). Adversarial label-flipping attack and defense for graph neural networks. In Plant, C., Wang, H., Cuzzocrea, A., Zaniolo, C., Wu, X. (eds.) 20th IEEE International Conference on Data Mining, ICDM 2020, (pp. 791–800).

Zhang, Z., Jia, J., Wang, B., & Gong, N.Z. (2021). Backdoor attacks to graph neural networks. In Lobo, J., Pietro, R.D., Chowdhury, O., Hu, H. (eds.) SACMAT '21: The 26th ACM Symposium on Access Control Models and Technologies, (pp. 15–26).

Zhou, Z.-H. (2016). *Machine learning*. Singapore: Springer.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., et al. (2020). Graph neural networks: A review of methods and applications. *AI Open, 1,* 57–81.

Zhu, D., Zhang, Z., Cui, P., & Zhu, W. (2019). Robust graph convolutional networks against adversarial attacks. In Teredesai, A., Kumar, V., Li, Y., Rosales, R., Terzi, E., Karypis, G. (eds.) Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, (pp. 1399–1407).

Zou, X., Zheng, Q., Dong, Y., Guan, X., Kharlamov, E., Lu, J., & Tang, J. (2021). Tdgia: Effective injection attacks on graph neural networks. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, (pp. 2461–2471). Association for Computing Machinery, New York, NY, USA

Zügner, D., & Günnemann, S. (2019). Adversarial attacks on graph neural networks via meta learning. In 7th International Conference on Learning Representations, ICLR 2019.

Zügner, D., & Günnemann, S. (2019). Certifiable robustness and robust training for graph convolutional networks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '19, pp. 246–256. Association for Computing Machinery, New York, NY, USA.

Zügner, D., Akbarnejad, A., & Günnemann, S. (2018). Adversarial attacks on neural networks for graph data. In Guo, Y., Farooq, F. (eds.) Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, (pp. 2847–2856).

Zügner, D., Borchert, O., Akbarnejad, A., & Günnemann, S. (2020). Adversarial attacks on graph neural networks: Perturbations and their patterns. ACM Trans. Knowl. Discov. Data **14**(5).