

Minimum Topology Attacks for Graph Neural Networks

Mengmei Zhang
Beijing University of Posts and
Telecommunications
China
zhangmm@bupt.edu.cn

Xiao Wang
Beijing University of Posts and
Telecommunications
China
xiaowang@bupt.edu.cn

Chuan Shi*
Beijing University of Posts and
Telecommunications
China
shichuan@bupt.edu.cn

Lingjuan Lyu
Sony AI
Japan
lingjuan.lv@sony.com

Tianchi Yang
Beijing University of Posts and
Telecommunications
China
yangtianchi@bupt.edu.cn

Junping Du
Beijing University of Posts and
Telecommunications
China
junpingdu@126.com

ABSTRACT

With the great popularity of Graph Neural Networks (GNNs), their robustness to adversarial topology attacks has received significant attention. Although many attack methods have been proposed, they mainly focus on fixed-budget attacks, aiming at finding the most adversarial perturbations within a fixed budget for target node. However, considering the varied robustness of each node, there is an inevitable dilemma caused by the fixed budget, i.e., no successful perturbation is found when the budget is relatively small, while if it is too large, the yielding redundant perturbations will hurt the invisibility. To break this dilemma, we propose a new type of topology attack, named minimum-budget topology attack, aiming to adaptively find the minimum perturbation sufficient for a successful attack on each node. To this end, we propose an attack model, named MiBTack, based on a dynamic projected gradient descent algorithm, which can effectively solve the involving non-convex constraint optimization on discrete topology. Extensive results on three GNNs and four real-world datasets show that MiBTack can successfully lead all target nodes misclassified with the minimum perturbation edges. Moreover, the obtained minimum budget can be used to measure node robustness, so we can explore the relationships of robustness, topology, and uncertainty for nodes, which is beyond what the current fixed-budget topology attacks can offer.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; • **Networks** → **Network reliability**.

KEYWORDS

graph, graph neural networks, adversarial attacks

*Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WWW '23, May 1–5, 2023, Austin, TX, USA

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9416-1/23/04...\$15.00
<https://doi.org/10.1145/3543507.3583509>

ACM Reference Format:

Mengmei Zhang, Xiao Wang, Chuan Shi, Lingjuan Lyu, Tianchi Yang, and Junping Du. 2023. Minimum Topology Attacks for Graph Neural Networks. In *Proceedings of the ACM Web Conference 2023 (WWW '23)*, May 1–5, 2023, Austin, TX, USA. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3543507.3583509>

1 INTRODUCTION

Graph is commonly used to model many real-world relationships, such as social networks, citation networks, and e-commerce networks. Recently, there has been a surge of interest in Graph Neural Networks (GNNs) for representation learning of graphs, which combine node feature and topology with neural networks and have achieved outstanding performance in various application areas [44].

Despite the great success, GNNs have been proved to often suffer from adversarial attacks [7, 47], especially for topology attacks where the attacker tries to slightly manipulate topology to modify the decision of GNNs. Recent research has developed stronger topology attack methods [9, 29, 33, 38, 43] to explore the robustness of GNNs. Specifically, existing works all belong to the fixed-budget topology attacks, i.e., for each node, the attacker finds the most adversarial perturbations within a given budget (i.e., a fixed number of perturbed edges). Usually, the budget of each node is heuristically specified as the same value or based on the node degree, and then it will keep unchanged once specified.

However, in the real-world graph, each node has unique feature and topology, and thus has varying adversarial robustness. It naturally raises a fundamental question: *Are the previous fixed-budget topology attacks truly suitable for evaluating the robustness of GNNs?* On the one hand, usually, an effective topology attack should lead to the misclassification of the target node and keeps it invisible, e.g., the topology cannot be modified too much. On the other hand, considering varied distribution of each node, evaluating the robustness of GNNs on node level can help better understand the robustness. Unfortunately, the fixed-budget perturbation cannot satisfy the above requirements. First, for each node, the heuristically specified budget is usually not optimal: no successful perturbation can be found when the specified budget is too small, while for the large one, the redundant perturbations will hurt the invisibility. As illustrated in Fig. 1, given blue (class c_1) target node v , existing fixed-budget topology attacks aim to maximally change the prediction (color) of v by modifying a fixed number of edges. Only perturbing one edge

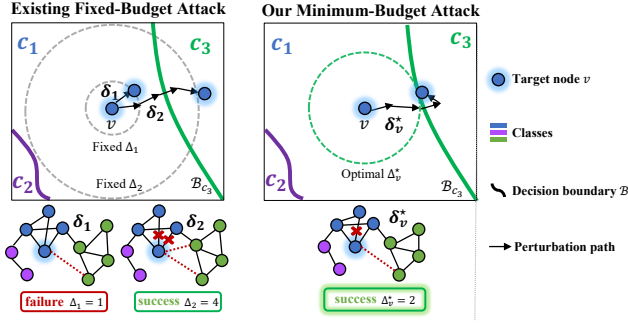


Figure 1: Illustrations of the existing fixed-budget topology attacks which may fail to cross decision boundary or cross too much, and our MiBTack can exactly cross the closest green decision boundary \mathcal{B}_{c_3} with no waste.

(budget $\Delta_1 = 1$), the attacker fails to find a successful perturbation δ_1 to cross purple (class c_2) or green (class c_3) decision boundary. While perturbing four edges ($\Delta_2 = 4$) will cross the boundary too much, hurting the invisibility of the attack. The above analysis shows that there is a dilemma between a fixed budget and an effective attack. Second, given the fixed budget, the total accuracy of all target nodes under attacks can be used to evaluate the robustness of GNNs, while evaluating the adversarial robustness of a sample (e.g., a node) amounts to finding the minimum perturbation for misclassification of it [25].

Clearly, the fixed budget restricts the potential of attack methods due to the different robustness of each node. Henceforth, we propose a new orthogonal minimum-budget topology attack, which aims at adaptively finding the minimum budget that is sufficient for a successful attack for each node. Thus the attacker can adapt to the robustness of each node as the green dotted line in Fig. 1. Moreover, if we can adaptively find the budget for each node, we can provide a better understanding or more insights on node robustness, uncertainty, etc.

In essence, the minimum-budget attacks are inherently different from the fixed-budget. The attacker with a fixed budget in Fig. 1 aims to flip Δ_2 edges to maximally cross the decision boundary. Obviously, this is a combinatorial optimization problem constrained by Δ_2 . To solve it, existing topology attacks often greedily flip edges for Δ_2 times [5, 47, 48]. Further, some works use Projected Gradient Descent (PGD) to project the combination of perturbed edges within the Δ_2 -limited space (grey dotted circle) [27, 38, 41]. While the attacker with an adaptive budget will learn a minimum perturbation (green dotted circle Δ_v^*) for crossing the decision boundary, which is challenging. With misclassification guarantee of non-convex GNN, the minimum-budget topology attack is essentially a non-convex constrained optimization problem, where PGD-based model can not be directly used [23]. And the greedy-based model is too myopic to find a good solution [20]. Grid search or binary search on budget for each node is also unbearable in practice.

In this paper, we propose a novel **Minimum-Budget Topology attack** method for GNNs named **MiBTack** based on dynamic projected gradient descent, in order to solve the above challenges. Specifically, we decouple the budget and perturbation, turning the

non-convex constrained optimization to alternatively solve the easier convex constrained optimization and update the budget separately. In each iteration, we search for the most adversarial topology attacks under the current budget with PGD and then the budget is enlarged or reduced based on whether these attacks succeed. Through repeatedly crossing the green decision boundary, MiBTack can find the optimal budget Δ_v^* , namely the green dotted circle centered at node v as shown in Fig. 1, which is tangent to the green decision boundary. Finally, our MiBTack can successfully attack (0 ACC) all target nodes with minimum budget.

The contributions of this paper are summarized as follows:

- We highlight the inherent bottleneck of existing fixed-budget topology attacks, then we propose an orthogonal type of topology attack for GNN, which aims to adaptively find minimum budget for a successful attack. The proposed attacks are highly threatening and can be used to quantify the node-level adversarial robustness.
- We propose an effective minimum-budget attack model MiBTack based on a differentiable dynamic projected gradient descent module, in order to solve the involving non-convex constraint on discrete topology.
- Extensive experiments on four real-world datasets and three GNNs show that our MiBTack can produce successful attacks with minimum perturbed edges. With the obtained minimum budgets, we also explore the relationships between robustness, topology and uncertainty.

2 RELATED WORK

Graph Neural Networks. Graph analysis has attracted considerable attention as graphs exist in various complex systems [28, 30]. GNNs [12, 16, 17, 44] on graph structured data have shown outstanding results in various tasks. A special form of GNNs is graph convolutional networks (GCNs) [12], which learn on graph structures using convolution operations and achieve state-of-the-art performance. To further improve GCNs, [13] derives an improved propagation scheme based on personalized PageRank, which further leverages a large, adjustable neighborhood for classification, and [35] further simplifies GCN by removing the non-linearities between GCN layers, etc. In this paper, we mainly focus on attacking GCN and its variants.

Topology Attacks. With the wide applications of GNNs, their robustness to adversarial attacks has received increasing attention, especially for topology attacks [5, 7, 9, 11, 21, 22, 29, 33, 38, 39, 42, 42, 47]. They try to find the most adversarial perturbations within a fixed budget Δ . To avoid the combinatorial search in discrete space, a basic solution is to greedily select perturbed edges with top- Δ adversarial scores [36, 47]. The advanced works suggest utilizing gradient to search for optimal combinations. Specifically, they often use projected gradient descent (PGD) to project the updated perturbations into the Δ -constrained space [9, 27, 38]. However, they mainly focus on fixed-budget topology attacks. We introduce the first study on a new minimum-budget topology attack. In addition, different from the previous simple convex constraint on budget, we need to solve an intractable non-convex constraint on GNN which cannot be directly solved by PGD [23].

Minimum-Norm Attacks. Adversarial attacks on machine learning can be categorized into maximum-confidence attacks and

minimum-norm attacks [23]. Given the target image, the former tries to find the perturbation δ within a given bound of the δ 's L_p norm to maximize the confidence of misclassification, e.g, FGSM [10]. Recently, the latter is proposed to find the perturbation with minimum L_p norm sufficient for misclassification [3, 23, 25]. In contrast to the former attacks that can only be used to evaluate model robustness, the minimum-norm attacks can also be used to measure an image's robustness, i.e., the distance of the image to the decision boundary. Intuitively, a more robust image requires a larger perturbation to cross the decision boundary [24]. Thus the minimum-norm attacks are often used to better understand the adversarial robustness and its inherent relations to underlying data distribution [2, 24]. However, there are no existing minimum-norm attacks for discrete graph data, and this paper sheds the first light on this important problem.

3 PRELIMINARIES

In this section, we briefly introduce the backgrounds and preliminaries of GNNs and existing topology attacks. We summarize the related works in Appendix 2.

3.1 Graph Neural Networks

We consider attacking GNNs on the semi-supervised node classification task. Formally, we define an undirected graph as $G = (\mathcal{V}, \mathcal{E}, X)$, where \mathcal{V} is the set of N nodes, \mathcal{E} represents the set of edges, and $X = [\mathbf{x}_1; \mathbf{x}_2; \dots; \mathbf{x}_N] \in \mathbb{R}^{N \times D}$ is the node feature matrix. We denote $\mathbf{a}_v = [a_{v1}; a_{v2}; \dots; a_{vN}] \in \{0, 1\}^N$ as the adjacency vector of node v . Let d_v and $\mathcal{N}(v)$ represent the degree and the neighborhood set of v , and the \tilde{d}_v and $\tilde{\mathcal{N}}(v)$ are the ones including self-loop (v, v). In this paper, we focus on node classification, where each node $v \in \mathcal{V}_L$ in training set $\mathcal{V}_L \subset \mathcal{V}$ has a label $y_v \in \mathcal{Y}$ and $\mathcal{Y} = \{1, 2, \dots, C\}$. Given \mathcal{V}_L , the goal of node classification task is to predict the class of node in unlabeled data \mathcal{V}_U . Taking the representative GCN [12] as an example, GCN uses convolution operations to aggregate neighboring nodes as follows:

$$\mathbf{h}_v^{(l)} = \text{ReLU}(\mathbf{W}^{(l)} (\sum_{u \in \tilde{\mathcal{N}}(v)} \tilde{d}_u^{-1/2} \tilde{d}_v^{-1/2} \mathbf{h}_u^{(l-1)})), \quad (1)$$

where $\mathbf{h}_v^{(l)}$ is the representation of node v at the l -th layer, and $\mathbf{h}_v^{(0)} = \mathbf{x}_v$ for initialization. $\mathbf{W}^{(l)}$ is the trainable weight matrix in the l -th layer, and we set $\theta = \{\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(L)}\}$ as all model parameters. The output of node v is $f_\theta(\mathbf{a}_v) = \text{softmax}(\mathbf{h}_v^{(L)}) \in [0, 1]^C$, where $f_\theta(\mathbf{a}_v)_c$ indicates the probability of node v being classified into class c , i.e., confidence. Then, the model parameters θ are learned by minimizing the cross-entropy loss on the outputs of the training nodes \mathcal{V}_L as follows:

$$\theta^* = \arg \min_{\theta} - \sum_{v \in \mathcal{V}_L} \ln f_\theta(\mathbf{a}_v)_{y_v}. \quad (2)$$

3.2 Adversarial Topology Attacks

Here we focus on adversarial topology attacks under the following settings:

- **Attack Goal.** Our attacker's goal is to lead GNN misclassify the target node v by manipulating its adjacency vector \mathbf{a}_v .

- **Attack Type.** We mainly focus on evasion attacks, where the parameters of the trained model are assumed to be fixed when being attacked. We also perform experiments for the challenging poisoning case, where the model is retrained after the attack.

- **Attack Knowledge.** We consider the worst case, in which an attacker can access the internal configurations (i.e., the learned parameters) of the targeted GNN model. This assumption ensures reliable vulnerability analysis in the worst case.

Formally, given a target node v , the attackers under these settings aim to decrease the performance of the well-trained GNN f_{θ^*} on node v . They optimize a topology perturbation vector $\delta_v \in [\delta_{v1}, \delta_{v2}, \dots, \delta_{vN}] \in \{0, 1\}^N$ for perturbing adjacency vector \mathbf{a}_v , where the u -th element $\delta_{vu} = 1$ means flipping the status of edge (v, u) (adding or deleting). Specifically, we add perturbation to \mathbf{a}_v by $\mathbf{a}'_v = \mathbf{a}_v + \mathcal{T}(\delta_v) = \mathbf{a}_v + (1 - 2\mathbf{a}_v) \odot \delta_v$, where \odot denotes element-wise multiplication. So element $\delta_{vu} = 1$ indicates adding edge (v, u) when $a_{vu} = 0$, deleting edge (v, u) when $a_{vu} = 1$.

Fixed-budget Topology Attacks. Existing topology attacks mainly focus on fixed-budget topology attacks. Given a target node v , they often specify the maximum amount of perturbation edges in advance, then try to find a perturbation vector δ_v to maximally degrade GNN's performance on v with a constraint of $\|\delta_v\|_0 \leq \Delta_v$ as follows:

$$\begin{aligned} \delta_v^* &= \underset{\delta_v}{\text{argmin}} \quad \mathcal{L}(\mathbf{a}'_v), \\ \text{s.t. } \|\delta_v\|_0 &\leq \Delta_v, \quad \delta_v \in \{0, 1\}^N, \end{aligned} \quad (3)$$

where the attack loss \mathcal{L} is the Carilli-Wagner (CW) loss following [38]:

$$\mathcal{L}(\mathbf{a}'_v) = f_{\theta^*}(\mathbf{a}'_v)_{y_v} - \max_{c \neq y_v} f_{\theta^*}(\mathbf{a}'_v)_c, \quad (4)$$

where the smaller $\mathcal{L}(\mathbf{a}'_v)$ indicates the stronger attacks. Clearly, as the constraints in Eq. (3) showed, the fixed-budget attacks try to search the most adversarial Δ_v perturbation edges, which is inherently a combinatorial optimization problem. Existing works solve this problem mainly by two ways: (1) A straight solution is greedy search: the attacker greedily flips the edge which degrades loss \mathcal{L} most and repeats it for Δ_v times. Doubtless, the greedy solver is myopic and will neglect some flipping actions that might be better in the long run [20]. (2) So the recent works [20, 38] tend to relax $\delta_v \in \{0, 1\}^N$ to its convex hull $\delta_v \in [0, 1]^N$, yielding a continuous optimization problem. Then they can optimize δ_v with gradient-based solvers which are proved to be much better than the classical greedy method. Specifically, they often use projected gradient descent (PGD) to solve the convex constraint $\delta_v \in [0, 1]^N$.

As shown in the above objective, it is hard to generate both successful and unnoticeable attacks by the fixed budget attacks. Given a small budget, the perturbed target node may not be misclassified ($\mathcal{L}(\mathbf{a}'_v) < 0$), while a large budget will hurt the invisibility of attacks.

4 MINIMUM-BUDGET TOPOLOGY ATTACK

4.1 Attack Objective

Considering the mentioned inherent dilemma of existing fixed-budget topology attacks, we first propose an orthogonal minimum-budget topology attack for GNNs, which aims to adaptively find the

minimum perturbation that is sufficient for the misclassification of target node. Formally, the objective of our attacks can be formulated as:

$$\begin{aligned} \delta_v^* &= \operatorname{argmin}_{\delta_v} \|\delta_v\|_0, \\ \text{s.t. } \mathcal{L}(\mathbf{a}'_v) &< \gamma, \quad \delta_v \in \{0, 1\}^N, \end{aligned} \quad (5)$$

where $\mathcal{L}(\mathbf{a}'_v)$ is CW attack loss defined in Eq. (4) and $\mathbf{a}'_v = \mathbf{a}_v + \mathcal{T}(\delta_v)$. γ ($\gamma > 0$) is the confidence level of misclassification, and a higher γ will lead to crossing decision boundary more. In this paper, we set $\gamma = 0$ by default, namely, generate the attacks which just cross the decision boundary exactly. So we can break the dilemma of fixed-budget about effectiveness and invisibility. Besides, δ_v^* can be used to evaluate the robustness of each node. Intuitively, if a node requires more perturbations to be successfully attacked, the node has a larger distance to the decision boundary and is more robust for topology attacks. Here we define the robustness ρ_v of node v as follows:

DEFINITION 1. Node Robustness. Given a node v and a graph neural network f_{θ^*} , the node robustness ρ_v is defined as the amount of perturbed edges which are sufficient to make f_{θ^*} misclassify node v with minimum perturbation.

Based on our attack objective, the node robustness can be calculated by $\rho_v = \|\delta_v^*\|_0$. Then we can use it to explore the relationships between robustness and data distribution, then provide a better understanding or more insights on node robustness.

Unfortunately, effectively and efficiently optimizing δ_v^* is challenging. Compared to the fixed-budget attack in Eq. (3), we can observe that our attack objective has an extra non-convex constraint $\mathcal{L}(\mathbf{a}'_v) < \gamma$ (the non-convexity of GNN model f_{θ^*}). Solving such non-convex constrained optimization problem is harder. The advanced PGD-based model can only solve the convex constraint like $\|\delta_v\|_0 \leq \Delta_v$, and is not readily applied in our attacks. Doubtlessly, the greedy-based model is too myopic to find a good solution.

4.2 The Proposed Model MiBTack

Next, we describe the proposed minimum-budget attack model, MiBTack, which solves the non-convex constrained optimization problem effectively. MiBTack is based on a dynamic PGD, which is mainly a projected gradient descent (PGD) method with a dynamic budget, so that the non-convex constrained optimization can be turned to alternatively solving the easier convex constrained optimization and updating budget separately. Fig. 2 shows the overall framework of MiBTack, which consists of two components: (1) We fix current budget Δ_v and search for δ_v under Δ_v which is a convex constrained optimization and can be solved by PGD. (2) If δ_v can lead to a misclassification of target node v (i.e., $\mathcal{L}(\mathbf{a}'_v) < 0$), the current Δ_v will be decreased for next iteration, otherwise increased. Thus the dynamic PGD can converge to $\mathcal{L}(\mathbf{a}'_v) < 0$ and lead to a finer search of a minimal budget by repeatedly crossing the decision boundary.

Updating Perturbation. In this step, we aim to keep the budget fixed and update perturbation with the constraint of current budget. Specifically, for i -iteration, we fix $\Delta_v^{(i)}$ as a constant and update perturbations $\delta_v^{(i-1)}$ by minimizing current attack loss $\mathcal{L}(\mathbf{a}'_v^{(i-1)})$

within budget $\Delta_v^{(i)}$:

$$\begin{aligned} \min_{\delta_v^{(i)}} \mathcal{L}(\mathbf{a}'_v^{(i-1)}), \\ \text{s.t. } \|\delta_v^{(i)}\|_0 \leq \Delta_v^{(i)}, \delta_v^{(i)} \in [0, 1]^N, \end{aligned} \quad (6)$$

where $\mathbf{a}'_v^{(i-1)} = \mathbf{a}_v + \mathcal{T}(\delta_v^{(i-1)})$. Following [38], we also relax the discrete $\delta_v^{(i)} \in \{0, 1\}^N$ to continue $[0, 1]^N$ for ease of gradients. As we can see, this is the easier convex constrained problem which can be solved through PGD, which contains gradient descent and projection. The idea of PGD is as follows: if the perturbation variable after the gradient descent update is out of current budget $\Delta_v^{(i)}$, PGD will project it back to the $\Delta_v^{(i)}$ -constrained space. In gradient descent, we calculate the gradient \mathbf{g} of the attack loss w.r.t. $\delta_v^{(i-1)}$:

$$\mathbf{g} \leftarrow \nabla_{\delta_v^{(i-1)}} \mathcal{L}(\mathbf{a}'_v^{(i-1)}). \quad (7)$$

Given \mathbf{g} , we update $\delta_v^{(i-1)}$ with the normalized gradient descent:

$$\tilde{\delta}_v^{(i)} \leftarrow \delta_v^{(i-1)} + \alpha \cdot \mathbf{g} / \|\mathbf{g}\|_2, \quad (8)$$

where α is the step size of gradient descent. Once $\delta_v^{(i-1)}$ is updated to $\tilde{\delta}_v^{(i)}$, to keep $\|\delta_v^{(i)}\|_0 \leq \Delta_v^{(i)}$ and $\delta_v^{(i)} \in [0, 1]^N$, we project $\tilde{\delta}_v^{(i)}$ via a projection operator:

$$\delta_v^{(i)} \leftarrow \operatorname{Proj}_{[0,1]^N} [\tilde{\delta}_v^{(i)} - \mu \mathbf{1}], \quad (9)$$

where the perturbation vector $\tilde{\delta}_v^{(i)}$ encodes the score of flipping edges of v . For u -th element, a larger value of $\tilde{\delta}_{vu}^{(i)}$ indicates a stronger attack effect of flipping edge (v, u) . To fulfill the constraint of $\Delta_v^{(i)}$, we denote the $(\Delta_v^{(i)} + 1)$ -th largest value of $\tilde{\delta}_v^{(i)}$ as a scalar μ , so only the most $\Delta_v^{(i)}$ perturbation edges are kept non-negative values in $\tilde{\delta}_v^{(i)} - \mu \mathbf{1}$, otherwise negative. Then we use clip operation $\operatorname{Proj}_{[0,1]^N}$ to set these negative elements as zero. Thus, after projection, $\delta_v^{(i)}$ can fulfill the constraints $\|\delta_v^{(i)}\|_0 \leq \Delta_v^{(i)}$ and $\delta_v^{(i)} \in [0, 1]^N$ in Eq. (6), and can be used to perturb the topology of v by $\mathbf{a}'_v^{(i)} \leftarrow \mathbf{a}_v + \mathcal{T}(\delta_v^{(i)})$.

Updating Budget. In this step, based on the above updated perturbation $\delta_v^{(i)}$, we will dynamically adjust the current budget $\Delta_v^{(i)}$ depending on whether the current perturbation can successfully attack.

To identify whether the current perturbation can succeed, one direct way is to test the predicted label of the perturbed adjacency vector $\mathbf{a}'_v^{(i)} \in [0, 1]^N$. While the real-world graph topology and its perturbations are discrete, so a more precise way is to project $\mathbf{a}'_v^{(i)}$ from continuous space to discrete space $\operatorname{Proj}_{\{0,1\}^N} [\mathbf{a}'_v^{(i)}] \in \{0, 1\}^N$ first. The projection method we use is straightforward: we choose the non-zero cells in perturbation $\delta_v^{(i)}$ and turn them into 1. Next, if the current perturbation $\delta_v^{(i)}$ is sufficient to cross decision boundary (i.e., $\mathcal{L}(\operatorname{Proj}_{\{0,1\}^N} [\mathbf{a}'_v^{(i)}]) < 0$), it can be determined that the current budget is large enough, and the optimal budget must be no more than $\Delta_v^{(i-1)}$, so we will decrease the budget $\Delta_v^{(i)}$ for next iteration with step size β by:

$$\Delta_v^{(i+1)} \leftarrow \min(\Delta_v^{(i)} - 1, \Delta_v^{(i)}(1 - \beta)), \quad (10)$$

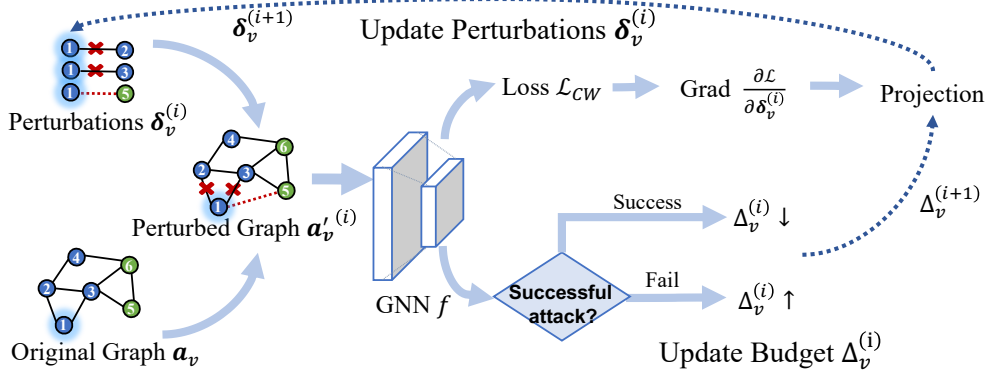


Figure 2: The overall framework procedure of our proposed MiBTack. In i -th iteration, MiBTack alternatively updates the current perturbation $\delta_v^{(i)}$ and $\Delta_v^{(i)}$ with a dynamic PGD algorithm.

otherwise, we increase current $\Delta_v^{(i)}$ by:

$$\Delta_v^{(i+1)} \leftarrow \max(\Delta_v^{(i)} + 1, \Delta_v^{(i)}(1 + \beta)). \quad (11)$$

Thus the dynamic PGD can lead to a finer search of a minimal budget by repeatedly crossing the decision boundary.

Initialization of δ_v . Here we point that the attack performance of our MiBTack may rely on the initial status $\mathbf{a}_v^{(0)} = \mathbf{a}_v + \delta_v^{(0)} = \mathbf{a}_v$ with $\delta_v^{(0)} = \mathbf{0}$. Since the existing iterative attacks often add noises monotonically along the direction of gradient [26], resulting in a dependence of initial direction. While in multi-classification, the initial attack direction may not be optimal: when minimize CW-based loss $\mathcal{L}(\mathbf{a}_v^{(0)})$, the yielded perturbation often move node to the wrong class c which has the largest confidence ($\max_{c \neq y_v} f_{\theta^*}(\mathbf{a}_v)_c$).

We find that it often fails to indicate the closest decision boundary in multi-class classification. Clearly, precisely estimating the closest decision boundary beforehand and along its direction guiding the later attack generation can help generate a more unnoticeable attack. Here we solve this problem by performing topology attacks with one-step towards each wrong class, and choose the class with the largest descent of \mathcal{L} , then use it to initialize δ_v . The details can be found in Appendix A.

Convergence. We alternatively update perturbation and budget until target node reaches the decision boundary, then we are only allowed to iterate for a given patience P times. To accelerate convergence and dampen oscillations around the boundary in last P iterations [23], we start to use cosine annealing to reduce the step sizes, including step size α for updating $\delta_v^{(i-1)}$ in Eq.8 and β for updating $\Delta_v^{(i)}$ in Eq.10 and Eq.11. Finally, the perturbed target node will reach a point where the (green) decision boundary is tangent to the (dotted green) sphere of Δ_v^* as shown in Fig. 1, yielding the optimal δ_v^* with minimum budget Δ_v^* . We summarize the pseudo-code in Appendix B for the whole process of MiBTack. Moreover, the space consumption analysis is provided in Appendix C.

5 EXPERIMENTS

5.1 Experimental Setup

Datasets. We employ the widely-used citation networks (Cora, Citeseer, Pubmed) as in [47], and a social network Polblogs in [1]. We conduct the experiments with 250 randomly selected nodes in the test set as the targeted nodes that are to be attacked following [32]. More details of dataset are shown in Appendix D.1.

Baselines. Since existing topology attack methods are all fixed-budget attacks, here we adapt the state-of-the-art baselines to minimum-budget topology attack. For greedy-based attacks, **Rand** randomly flips the edges in \mathbf{a}_v for target node v . **DICE** [34] randomly disconnects the links of v to the neighbors with the same label y_v and connects the v to the nodes with class $c \neq y_v$. **DICE-t** extends DICE by adding the edges (v, u) where u belongs to the target class c^* and $c^* = \operatorname{argmax}_{c \neq y_v} f_{\theta^*}(\mathbf{a}_v)$. **FGA** [9] flips one edge

at a time by performing gradient update along the direction of the sign of gradients of loss function w.r.t. each adjacency matrix. **Nettack** [47] generates perturbation edges greedily by exploiting the properties of the linearized GCN surrogate. We can easily adapt above greedy-based attacks by consistently adding perturbations until target node v is misclassified or the number of perturbed edges is more than 1000. For PGD-based attacks, **PGD** [38] uses projected gradient descent to project the perturbation edges into the space of given budget Δ . **PRBCD** [9] generates Δ perturbation edges for large-scale graph by projected randomized block coordinate descent. Above PGD-based algorithms model the fixed budget attacks as a convex constrained optimization, and cannot be directly used for our minimum-budget attacks which is essentially a non-convex constrained optimization problem. Following [9], we use the degree of the target node that we currently attack as budget. More implementation details of baselines and our MiBTack, e.g., hyperparameters, are provided in Appendix D.2 and D.3.

Target Models. To validate the generalization ability of our proposed attacker, we choose three popular graph neural networks¹: (1) GCN [12] is a representative GNN and learns on graph structures

¹The code of GNNs can be found in <https://github.com/BUPT-GAMMA/GammaGL>.

Table 1: Attack performance. The lower classification accuracy (ACC) of all attacked target nodes indicates a better attack performance. The lower total budget (TB) hints the better invisibility.

Datasets	GNNs	Metrics	PGD	PRBCD	Rand	DICE	DICE-t	FGA	Nettack	MiBTack
Cora	GCN	ACC	0.032	0.020	0.112	0.000	0.052	0.000	0.000	0.000
		TB	765	779	10349	4057	1171	384	357	330
	SGC	ACC	0.020	0.008	0.104	0.000	0.000	0.000	0.000	0.000
		TB	787	792	6774	4927	1155	437	443	385
	APPNP	ACC	0.256	0.100	0.152	0.000	0.000	0.000	0.000	0.000
		TB	777	796	2919	2518	1237	683	679	507
Citeseer	GCN	ACC	0.100	0.092	0.132	0.000	0.000	0.000	0.000	0.000
		TB	691	697	4290	4292	1343	484	438	444
	SGC	ACC	0.048	0.028	0.144	0.000	0.000	0.000	0.000	0.000
		TB	689	693	6362	3461	1002	411	414	386
	APPNP	ACC	0.172	0.136	0.064	0.000	0.016	0.000	0.000	0.000
		TB	650	696	7386	2201	949	494	551	412
Polblogs	GCN	ACC	0.000	0.004	0.380	0.000	0.000	0.000	0.000	0.000
		TB	7112	7105	18891	7261	7261	2038	2011	1961
	SGC	ACC	0.036	0.064	0.468	0.000	0.000	0.000	0.000	0.000
		TB	7027	7062	19250	7304	7304	3518	3660	3328
	APPNP	ACC	0.396	0.132	0.528	0.000	0.000	0.000	0.000	0.000
		TB	7125	7125	14551	7543	7543	5619	5739	5494
Pubmed	GCN	ACC	0.108	0.016	0.292	0.000	0.000	0.000	0.000	0.000
		TB	742	770	2935	4857	1386	357	354	348
	SGC	ACC	0.196	0.020	0.284	0.000	0.000	0.000	0.000	0.000
		TB	683	772	6740	2540	1346	368	364	359
	APPNP	ACC	0.296	0.148	0.284	0.000	0.000	0.000	0.000	0.000
		TB	720	744	2111	2340	1241	529	514	527

using convolution operations. We train a 2-layer GCN with learning rate 0.01, where the number of units in hidden layer is 16. In addition, the dropout rate is 0.5, weight decay is $5e-4$. (2) SGC [35] simplifies GCNs through successively removing nonlinearities and collapsing weight matrices between consecutive layers. For SGC, the learning rate is 0.01, the number of units is 16, the dropout rate is 0.5, and weight decay is $5e-6$. (3) APPNP [13] improves GCN by leveraging residual connection to preserve the information of raw features. The learning rate of APPNP is 0.01, the number of units in hidden layer is 64, the dropout rate is 0.5, weight decay is $5e-6$.

Metrics. Here we utilize two metrics to evaluate the minimum-budget topology attacks. (1) Accuracy (ACC): We use the accuracy of GNN model on all attacked target nodes to show whether the attack models can make all nodes misclassification. The zero value of ACC indicates 100% attack successful rate. (2) Total Budget (TB): Total budgets is the amount of perturbation edges for attacking all target nodes. A lower TB hints a more unnoticeable attack.

5.2 Attack Effectiveness

Here we evaluate the effectiveness of our model against all baselines for minimum-budget topology attacks, under four datasets and three GNNs. The overall results are presented in Table 1, where we have the following observations:

- Our MiBTack can outperform all baselines in most scenarios, yielding minimum attacks with the guarantee of misclassification

of all nodes. First, the PGD-based attacks PGD and PRBCD are hard to achieve 0 accuracy and need large budgets, since they predefine the node budget by its degree, which is usually not optimal. Based on random strategies, Rand, DICE and DICE-t, need large amounts of perturbation edges. Then Nettack and FGA have better performance than above methods, since they greedily flip the adversarial edges with the highest gradient w.r.t. attack loss. Finally, our MiBTack can generate the minimum attacks with the guarantee of 0 accuracy, averagely saving at least 60 perturbed edges. Compared to myopic greedy search, MiBTack can search the better combinatorial perturbed edges based on our dynamic PGD.

- Specifically, we observe that our MiBTack has the largest improvement on Polblogs. This is because that the topology connections in Polblogs are much more dense than other datasets, thus the nodes in Polblogs require significantly larger budget to successfully attack, namely more iterations, where greedy based baselines will accumulate more errors than gradient based methods.

- We also observe that, under most scenarios, APPNP needs higher TB than GCN and SGC, indicating more adversarial robustness of APPNP. The reason may be that APPNP, which leverages residual connection to preserve the information of raw features, may be less dependent on topology and thus have better robustness to topology attacks. Besides, GNNs are more robust on Polblogs due to the relatively more dense graph of Polblogs.

To better evaluate how effective is our model, take the more robust Polblogs dataset as an example, we plot the classification margins of GCN and APPNP in Fig. 3. Each point in the plot represents one target node v . The classification margin of v is $f_{\theta}(\mathbf{a}_v)_{y_v} - \max_{c \neq y_v} f_{\theta}(\mathbf{a}_v)_c$ where y_v is the ground truth class, $f_{\theta}(\mathbf{a}_v)_{y_v}$ is the probability of node v being classified into class y_v . The positive classification margin value of node v indicates a failed attack, where v fails to cross the decision boundary. For the negative value, v is misclassified and a lower value indicates that v crosses the decision boundary more. Compared to the clean scenario, we find that PRBCD with node degree as budget, can strongly affect GNNs most, but fail to lead the misclassification for all target nodes. On the contrast, our MiBTack and the greedy-based attacks, FGA and Nettack, affect more slightly but successfully attack all target nodes. Most remarkably, our MiBTack achieves higher margin value than FGA and Nettack, indicating that the nodes, attacked by MiBTack, cross the decision boundary less. This is because that the dynamic PGD in our MiBTack can converge to decision boundary and lead to a finer search of a minimal combination of perturbation edges by repeatedly crossing the decision boundary.

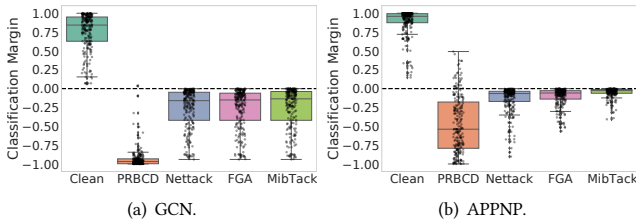


Figure 3: The node classification margins on Polblogs. Our MiBTack leads all target nodes misclassified and affects them most slightly by finding minimum and successful attacks.

5.3 Transferability of Attacks

We mainly focus on evasion (test time) attacks against GNNs, but we also evaluate the transferability of our MiBTack under poisoning (train time) setting and defended setting.

Poisoning Attacks. Following [9], we retrain GNNs on the perturbed graph of an evasion attack for poisoning. Note that poisoning attack is a more challenging scenario since retraining GNNs will introduce much uncertainty to attackers. Fortunately, our MiBTack can handle such uncertainty by improving the confidence level of misclassification γ . Take Cora and Citeseer as examples, we gradually increase γ and present the ACC of attacked target nodes in Fig. 4 (a) and (b). As seen, for each GNN model, the accuracy of target nodes is close to 0 and shows the similar declining trend with the increasing $\gamma \in \{0.00, 0.05, 0.10, 0.15, 0.20\}$. So MiBTack with larger γ tends to generate stronger attack under poisoning scenarios. But there is no free lunch: it is expected that the needed total budget will grow, as showed in Fig. 4 (c) and (d). In conclusion, our MiBTack can achieve powerful attacks under challenging poisoning attacks.

Attacks against Defended GNNs. Some countermeasures are proposed to improve the robustness of GNNs [4, 6, 14, 15, 18, 19, 37, 40, 46]. Besides vanilla GNNs, we also consider these defended GNNs: RGCN [45] adopts a variance-based attention mechanism to

Table 2: The total budget of our attack models with/without the proposed dynamic PGD and Initialization.

Dataset	W/o DPGD	W/o Init	MiBTack
Cora	385	364	330
Citeseer	484	459	444
Polblogs	2038	1967	1961
Pubmed	357	353	348

remedy the propagation of adversarial attacks. JacGCN [36] filters edges based on attribute Jaccard similarity (here threshold is 0.03). With no surprise, directly transferring attacks from vanilla GNNs to defended GNNs will lead to a huge drop of attack performance as shown in Fig. 5. The accuracy of target nodes attacked by Nettack, FGA and our MiBTack significantly increase, e.g., from accuracy 0 of GCN to around 0.7 of JacGCN on Cora dataset. However, we can easily extend MiBTack against defended GNNs. For JacGCN, we restrict the search space with Jaccard similarity when generating perturbation edges, yielding **MiBTack-J**. For RGCN, we replace the target GNN in the attack model by RGCN, named **MiBTack-R**. As seen, the accuracy of JacGCN and RGCN (i.e., blue bars) under MiBTack-J and MiBTack-R dramatically drop even reach 0, costing only a little bit more budgets (i.e., pink bars). Doubtlessly, if the defended GNNs are complex even black-box, the gradient of adjacency matrix is unavailable, and the straight solution in MiBTack-J or MiBTack-R is not feasible. We leave designing effective attack models against black-box GNNs as our future work.

5.4 Model Analysis

Ablation Study. We conduct an ablation study to evaluate the necessity of the components of our attack model. Take GCN as example, we report the total budget of our dynamic projected gradient descent and exploration operation as shown in Table 2 (ACC are all 0). Specifically, we compare our full model **MiBTack** with the variant (**W/o DPGD**) using greedy method to replace the dynamic PGD and the model without initialization (**W/o Init**). One can observe that full model **MiBTack** significantly behaves better than W/o DPGD, suggesting that our dynamic PGD can improve the combinatorial optimization problem and the challenge of non-convex constraint optimization under PGD can be alleviated by our designs. Meanwhile, compared to W/o Init, it can be seen that our customized initialization also benefits the proposed **MiBTack**.

Impact of P (patience). Fig. 6 demonstrates how the proposed attack methods perform against three GNNs when patience P increases under three datasets. One can observe that the proposed model obtains lower minimum budget with the increasing of patience P , and only requires a few of the trials to converge. This is because P is the number of maximal iteration after crossing the decision boundary, and a larger P will lead to a better adjusting of Δ_v iteratively. Meanwhile, our model can quickly converge below 200 iterations, indicating the efficiency of our model.

Impact of α . Hyperparameter α is the step size for updating δ_v in Eq. 8. Take the datasets Cora and Citeseer as examples, the results are reported in Fig. 7. As seen, there exists an optimal α that delivers the minimum budget. This is because our methods with too large step size α may fail to converge to a good solution. When step size α is too small, the elements of flipping operation

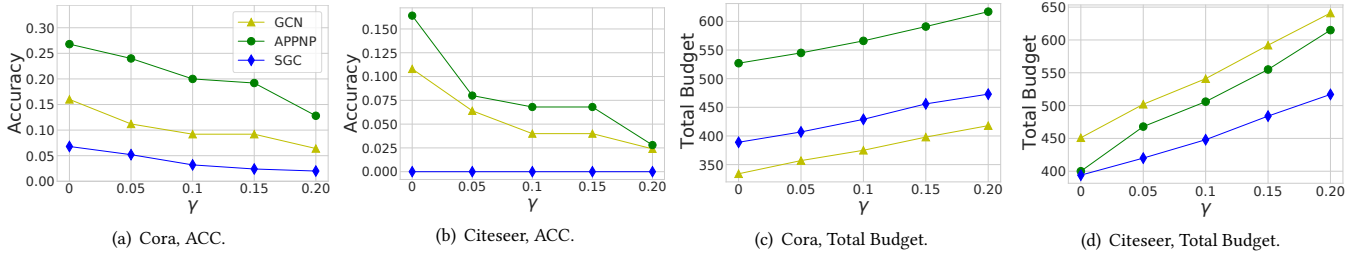


Figure 4: Performance of poisoning attacks on Cora and Citeseer.

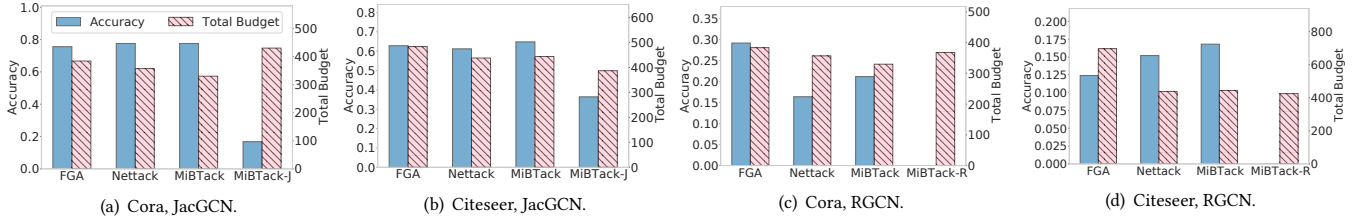


Figure 5: Performance of attacks against defended GNNs (i.e., JacGCN and RGCN). We report the total budget and the accuracy of target nodes of defended GNNs under FGA, Nettack, our MiBTack, our variants MiBTack-J and MiBTack-R.

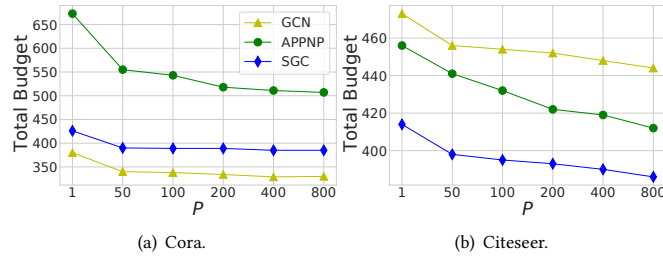


Figure 6: Impact of P .

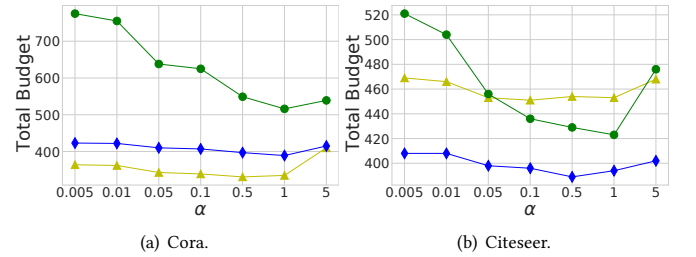


Figure 7: Impact of α .

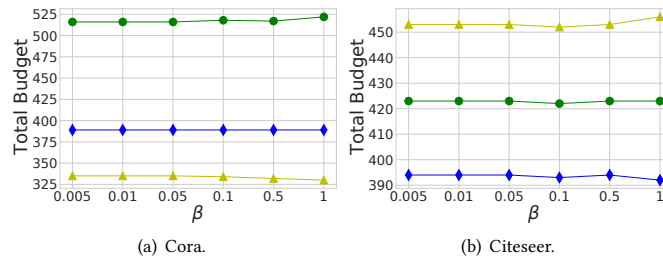


Figure 8: Impact of β .

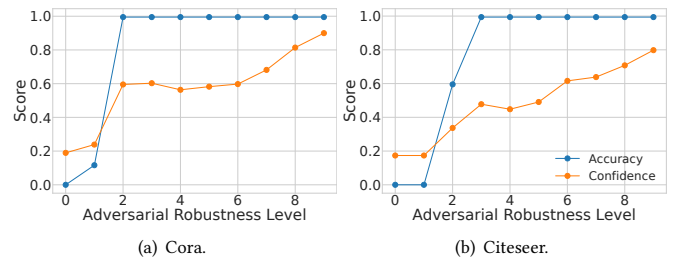


Figure 10: Robustness versus uncertainty.

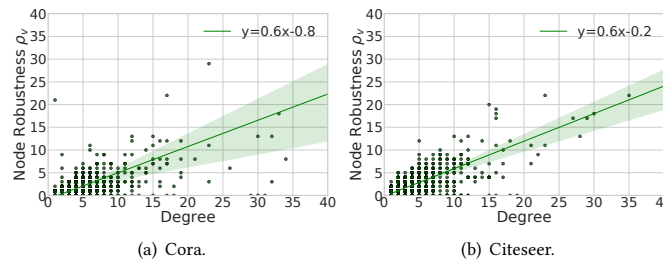


Figure 9: Robustness versus degree.

vector have small values in continuous space and can only change the adjacency vector \mathbf{a}_v slightly, which often leads to sub-optimal results due to the discrete structure of graph data.

Impact of β . We test the impact of hyperparameter β , which is the step size for updating budget Δ_v . As shown in Fig. 8, we can observe that, basically, our framework is stable when β is within the range from $\{0.005, 0.01, 0.05, 0.1, 0.5, 1\}$. This is because that the step size of Δ_v is expected to only affect the time of convergence.

5.5 Node Robustness Analysis

For each node v , we use the minimum budget Δ_v^* generated by our MiBTack as node robustness ρ_v .

Relationships of node robustness and degree. We analyze how the structure of the target node, i.e., its degree, affects the robustness. For each target node v in Cora and Citeseer, we plot its degree and the node robustness ρ_v found by MiBTack, then plot the corresponding linear regression line and equation as shown in Fig. 9. There are some observations: First, there is a positive correlation between degree and ρ_v as expected in existing works [47]. The high degree nodes are harder to be perturbed to cross decision boundaries. Second, there exists a mass of nodes under regression line, they only need the perturbation edges with fewer than half of neighbors to mislead GCN, showing high vulnerability.

Relationships of node robustness and uncertainty. The uncertainty of GNNs' predictions, indicating how much we should trust our GNN, is crucial for their deployment in many risk-sensitive applications. Existing works often use the predicted probability (i.e. confidence) as the uncertainty indicator. Here we study the relationship between adversarial robustness and confidence together with accuracy. Following [24], we rank the input node according to their node robustness (ρ_v) and then divide the dataset into 10 equally-sized subsets. For each adversarial robustness subset, we compute accuracy and the average confidence score of the predicted class as shown in Fig. 10. One can clearly see that both accuracy and confidence increase with the adversarial robustness of the input data. With the increase of adversarial robustness level, compared to accuracy, the confidence becomes higher first then consistently lower. This indicates that GCN tends to give over-confident predictions for the easily attacked nodes, but give under-confident predictions otherwise, which fits nicely with the conclusion of existing work (GNNs are under-confident) [31]. Moreover, it makes a new interesting observation: There may exist a turning point for GNNs from over-confident to under-confident. This observation may help improve the confidence calibration for GNNs.

6 CONCLUSIONS

In this paper, we introduce the first study on the minimum-budget topology attack on GNNs, to find the smallest perturbation for successful attack. We propose an effective attack model MiBTack based on the differentiable dynamic projected gradient descent (PGD). Our MiBTack can effectively find the minimum perturbations for cross above decision boundary by differentiable dynamic PGD, solving the inherent intractable non-convex constrained optimization. The experimental results show that MiBTack can achieve 100% attack success rate with minimum perturbation edges. Besides, we use these obtained minimum budget to study the relationships between robustness, topology and uncertainty. An interesting direction for future work is to extend our MiBTack to black-box setting.

ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China (No. U20B2045, 62192784, 62172052, 62002029, U1936104).

REFERENCES

- [1] Lada A. Adamic and Natalie S. Glance. 2005. The political blogosphere and the 2004 U.S. election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery, LinkKDD 2005, Chicago, Illinois, USA, August 21-25, 2005*. ACM, 36–43.
- [2] Nicholas Carlini, Úlfar Erlingsson, and Nicolas Papernot. 2019. Distribution Density, Tails, and Outliers in Machine Learning: Metrics and Applications. *ArXiv abs/1910.13427* (2019).
- [3] Nicholas Carlini and David A. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. *2017 IEEE Symposium on Security and Privacy (SP)* (2017), 39–57.
- [4] Heng Chang, Yu Rong, Tingyang Xu, Yatao Bian, Shi-Yao Zhou, Xin Wang, Junzhou Huang, and Wenwu Zhu. 2021. Not All Low-Pass Filters are Robust in Graph Convolutional Networks. In *NeurIPS*.
- [5] Heng Chang, Yu Rong, Tingyang Xu, Wenbing Huang, Honglei Zhang, Peng Cui, Wenwu Zhu, and Junzhou Huang. 2020. A Restricted Black-box Adversarial Framework Towards Attacking Graph Embedding Models. AAAI.
- [6] Liang Chen, Jintang Li, Qibiao Peng, Yang Liu, Zibin Zheng, and Carl Yang. 2021. Understanding Structural Vulnerability in Graph Convolutional Networks. In *IJCAI*.
- [7] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial Attack on Graph Structured Data. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 1123–1132.
- [8] Boyuan Feng, Yuke Wang, Xu Li, and Yufei Ding. 2020. Scalable Adversarial Attack on Graph Neural Networks with Alternating Direction Method of Multipliers. *ArXiv abs/2009.10233* (2020).
- [9] Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. 2021. Robustness of graph neural networks at scale. *Advances in Neural Information Processing Systems* 34 (2021), 7637–7649.
- [10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. *CoRR abs/1412.6572* (2015).
- [11] Hussain Hussain, Tomislav Duricic, E. Lex, D. Helic, Markus Strohmaier, and Roman Kern. 2021. Structack: Structure-based Adversarial Attacks on Graph Neural Networks. *Proceedings of the 32nd ACM Conference on Hypertext and Social Media* (2021).
- [12] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.
- [13] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.
- [14] Kuan Li, Yang Liu, Xiang Ao, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2022. Reliable Representations Make A Stronger Defender: Unsupervised Structure Refinement for Robust GNN. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (2022).
- [15] Ao Liu, Beibei Li, Tao Li, Pan Zhou, and Rui Wang. 2022. AN-GCN: An Anonymous Graph Convolutional Network Against Edge-Perturbing Attacks. *IEEE transactions on neural networks and learning systems* PP (2022).
- [16] Juncheng Liu, Bryan Hooi, Kenji Kawaguchi, and X. Xiao. 2023. MGNNI: Multi-scale Graph Neural Networks with Implicit Layers. In *NeurIPS*.
- [17] Juncheng Liu, Kenji Kawaguchi, Bryan Hooi, Yiwei Wang, and Xiaokui Xiao. 2021. EIGNN: Efficient Infinite-Depth Graph Neural Networks. In *NeurIPS*. 18762–18773.
- [18] Xiaorui Liu, Jiayuan Ding, Wei Jin, Han Xu, Yao Ma, Zitao Liu, and Jiliang Tang. 2021. Graph Neural Networks with Adaptive Residual. In *NeurIPS*.
- [19] Xiaorui Liu, Wei Jin, Yao Ma, Yaxin Li, Hua Liu, Yiqi Wang, Ming Yan, and Jiliang Tang. 2021. Elastic Graph Neural Networks. In *ICML*.
- [20] Xuanqing Liu, Si Si, Xiaojin Zhu, Yang Li, and Cho-Jui Hsieh. 2019. A Unified Framework for Data Poisoning Attack to Graph-based Semi-supervised Learning. In *NeurIPS*.
- [21] Zihan Liu, Yun Luo, Zelin Zang, and Stan Z. Li. 2022. Surrogate Representation Learning with Isometric Mapping for Gray-box Graph Adversarial Attacks. *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining* (2022).
- [22] Yao Ma, Suhang Wang, Lingfei Wu, and Jiliang Tang. 2019. Attacking Graph Convolutional Networks via Rewiring. *CoRR abs/1906.03750* (2019).
- [23] Maura Pintor, Fabio Roli, Wieland Brendel, and Battista Biggio. 2021. Fast Minimum-norm Adversarial Attacks through Adaptive Norm Constraints. *ArXiv abs/2102.12827* (2021).
- [24] Yao Qin, Xuezhi Wang, Alex Beutel, and Ed H. Chi. 2020. Improving Calibration through the Relationship with Adversarial Robustness.
- [25] Jérôme Rony, Luiz G. Hafemann, Luiz Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. 2019. Decoupling Direction and Norm for Efficient Gradient-Based L2 Adversarial Attacks and Defenses. *CVPR* (2019), 4317–4325.
- [26] Yucheng Shi, Siyu Wang, and Yahong Han. 2019. Curls & Whey: Boosting Black-Box Adversarial Attacks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 6512–6520.
- [27] Mingjie Sun, Jian Tang, Huichen Li, Bo Li, Chaowei Xiao, Yao Chen, and Dawn Xiaodong Song. 2018. Data Poisoning Attack against Unsupervised Node Embedding Methods. *ArXiv abs/1810.12881* (2018).

- [28] Xiyang Sun and Fumiyasu Komaki. 2023. BHGNN-RT: Network embedding for directed heterogeneous graphs. *ArXiv abs/2311.14404* (2023). <https://api.semanticscholar.org/CorpusID:265445839>
- [29] Binghui Wang, Youqin Li, and Pan Zhou. 2022. Bandits for Structure Perturbation-based Black-box Attacks to Graph Neural Networks with Theoretical Guarantees. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), 13369–13377.
- [30] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding. In *AAAI Conference on Artificial Intelligence*. <https://api.semanticscholar.org/CorpusID:29154877>
- [31] Xiao Wang, Hongrui Liu, Chuan Shi, and Cheng Yang. 2021. Be Confident! Towards Trustworthy Graph Neural Networks via Confidence Calibration. *ArXiv abs/2109.14285* (2021).
- [32] Yiwei Wang, Shenghua Liu, Minji Yoon, Hemank Lamba, Wei Wang, Christos Faloutsos, and Bryan Hooi. 2020. Provably Robust Node Classification via Low-Pass Message Passing. In *ICDM*. 621–630.
- [33] Zhengyi Wang, Zhongkai Hao, Ziqiao Wang, Hang Su, and Jun Zhu. 2022. Cluster Attack: Query-based Adversarial Attacks on Graph with Graph-Dependent Priors. In *IJCAI*.
- [34] Marcin Wanek, Tomasz P. Michalak, Talal Rahwan, and Michael Wooldridge. 2017. Hiding individuals and communities in a social network. *Nature Human Behaviour* 2 (2017), 139–147.
- [35] Felix Wu, Tianyi Zhang, Amauri H. de Souza, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. *ArXiv abs/1902.07153* (2019).
- [36] Huijun Wu, Chen Wang, Yu. O. Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial Examples for Graph Data: Deep Insights into Attack and Defense. In *IJCAI*.
- [37] Hui Xu, Liyao Xiang, Jiahao Yu, Anqi Cao, and Xinbing Wang. 2021. Speedup Robust Graph Structure Learning with Low-Rank Information. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (2021).
- [38] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. 2019. Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective. *ArXiv abs/1906.04214* (2019).
- [39] Runze Yang and Teng Long. 2021. Derivative-free optimization adversarial attacks for graph convolutional networks. *PeerJ Computer Science* 7 (2021).
- [40] Baoliang Zhang, Xiaoxi Guo, Zhenchuan Tu, and Jia Zhang. 2022. Graph alternate learning for robust graph neural networks in node classification. *Neural Comput. Appl.* 34 (2022), 8723–8735.
- [41] He Zhang, Bang Wu, Xiangwen Yang, Chuan Zhou, Shuo Wang, Xingliang Yuan, and Shirui Pan. 2021. Projective Ranking: A Transferable Evasion Attack Method on Graph Neural Networks. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (2021).
- [42] Sixiao Zhang, Hongxu Chen, Xiangguo Sun, Yicong Li, and Guandong Xu. 2022. Unsupervised Graph Poisoning Attack via Contrastive Loss Back-propagation. *Proceedings of the ACM Web Conference 2022* (2022).
- [43] Qinkai Zheng, Xu Zou, Yuxiao Dong, Yukuo Cen, Da Yin, Jiarong Xu, Yang Yang, and Jie Tang. 2021. Graph Robustness Benchmark: Benchmarking the Adversarial Robustness of Graph Machine Learning. *ArXiv abs/2111.04314* (2021).
- [44] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2020. Graph Neural Networks: A Review of Methods and Applications. *ArXiv abs/1812.08434* (2020).
- [45] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust Graph Convolutional Networks Against Adversarial Attacks. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019).
- [46] Jun Zhuang and Mohammad al Hasan. 2022. Defending Graph Convolutional Networks against Dynamic Graph Perturbations via Bayesian Self-supervision. In *AAAI*.
- [47] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial Attacks on Neural Networks for Graph Data. In *KDD*. 2847–2856.
- [48] Daniel Zügner and Stephan Günnemann. 2019. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019*. OpenReview.net.

A INITIALIZATION

The attack performance of our MiBTack may rely on the initial status $\mathbf{a}'_v^{(0)} = \mathbf{a}_v + \delta_v^{(0)}$. While in multi-classification, the initial attack direction may not be optimal, namely dose not point to the closest decision boundary. Specifically, in the first iteration, the initial adjacency $\mathbf{a}'_v^{(0)} = \mathbf{a}_v$ often has small and similar prediction confidence on all wrong classes $c (c \neq y_v)$, e.g., node 352 in Cora dataset has $f_{\theta^*}(\mathbf{a}'_v^{(0)})$ [0.0133, 0.0039, 0.0215, 0.0361, 0.0130, 0.0550] for $c \neq y_v$ and 0.8571 for y_v . The \mathcal{L} will update the perturbation vector along the direction to the wrong class with the largest prediction confidence $\max_{c \neq y_v} f_{\theta^*}(\mathbf{a}'_v^{(0)})$. Obviously, the wrong class with largest confidence (0.0550) may only have little superiority than other wrong classes so it probably fails to indicate the closest decision boundary in multi-class classification. Note that the superiority of the wrong class picked in first iteration can be inherited under \mathcal{L} , guiding target node to cross a non-closest decision boundary with more perturbation edges wasted.

However, it is time-consuming to precisely search the closest decision boundary, since the attacker needs to compute the distance (i.e., minimum budgets) of starting point to each decision boundary \mathcal{B}_c for class $c \in \{1, \dots, C\} \setminus \{y_v\}$. Here we estimate the closest decision boundary in multi-classification by a fast one-step attack instead of the time-consuming brute force, to trade off performance and efficiency. Then we initialize the starting point by slightly moving to the specified closest decision boundary, aiming to guide the generated attack trajectory to cross this boundary.

Clearly, this reminds us that precisely estimating the closest decision boundary beforehand and specifying it to guide the later attack generation can help generate a more unnoticeable attack. However, given target node v , the precise way to search the closest decision boundary is to perform minimum-budget attacks towards each decision boundary $\mathcal{B}_c (c \in \{1, \dots, C\} \setminus \{y_v\})$, and choose the boundary with minimal budget as the closest one \mathcal{B}_{c^*} , which is time-consuming. To trade off performance and efficiency, we simplify the minimum-budget attack for each class c by limiting only flipping one edge, namely one-step attack, yielding a one-step perturbation δ_v^c towards c as:

$$\begin{aligned} \delta_v^c &= \underset{\delta_v}{\operatorname{argmin}} \quad \mathcal{L}_{CW}(\mathbf{a}_v + \mathcal{T}(\delta_v), c), \\ &= \underset{\delta_v}{\operatorname{argmin}} \quad f_{\theta^*}(\mathbf{a}_v + \mathcal{T}(\delta_v))_{y_v} - f_{\theta^*}(\mathbf{a}_v + \mathcal{T}(\delta_v))_c, \\ &\text{s.t.} \quad \|\delta_v\|_0 \leq 1. \end{aligned} \quad (12)$$

where $\delta_v^c \in [\delta_{v1}^c, \delta_{v2}^c, \dots, \delta_{vN}^c] \in [0, 1]^N$ can be found by minimizing the difference of confidence between y_v and c . Then we approximately choose the most potential class c^* by comparing the decrease loss under each c :

$$c^* = \underset{c}{\operatorname{argmax}} \quad \mathcal{L}_{CW}(\mathbf{a}_v + \delta_v^c, c). \quad (13)$$

Then we initialize the perturbation $\delta_v^{(0)}$ as $\delta_v^{c^*}$, leading to a better starting point $\mathbf{a}'_v^{(0)} = \mathbf{a}_v + \mathcal{T}(\delta_v^{(0)})$ which tends to have a significantly higher confidence on c^* .

Table 3: Dataset statistics.

	Polblogs	Cora	Citeseer	Pubmed
#Nodes	1,222	2,485	2,110	19,717
#Edges	16,724	5,069	3,668	44,325
#Features	-	1,433	3,703	500
#Classes	2	7	6	3

B ALGORITHM

Here we provide the pseudo-code in Algorithm 1 for the whole process of MiBTack. Given a target node v , we initialize the starting adjacency vector $\mathbf{a}'_v^{(0)}$ by one-step attack for each class $c \in \{1, \dots, C\}$ in Line 2-3. The training procedure of the perturbation vector δ_v is presented in Line 4-11, which learns δ_v with PGD (Line 5-7), then dynamically adjusts Δ_v and other parameters (Line 8-11). Lastly, we output the attacks when $P \leq 0$ or $\|\delta_v^*\|_0 \leq 1$. Intuitively, before attack, we first estimate the closest decision boundary, and then use it to initialize the starting point to guide the attack trajectory to cross the closest decision boundary. Then we search for the most adversarial topology attacks under the current budget with PGD and then the budget is enlarged or reduced based on whether these attacks succeed. Through repeatedly crossing the green decision boundary, MiBTack can find the optimal budget Δ_v^* , namely the green dotted circle centered at node v , which is tangent to the green decision boundary.

C SPACE CONSUMPTION ANALYSIS

With the burden of a dense adjacency matrix, the gradient-based topology attacks, including our MiBTack, usually have the space complexity $O(N^2)$ [9], where N is the number of nodes. Fortunately, there are several works overcome this limitation [8, 9]. For example, PRBCD [9] based on Randomized Block Coordinate Descent (R-BCD) only has linear complexity w.r.t. the budget Δ . It worth pointing out that the space complexity of our MiBTack can be further reduced by combining with these methods. We leave extending our MiBTack with Randomized Block Coordinate Descent as our future work.

D FUTURE DETAILS ON EXPERIMENT

D.1 Dataset

The dataset characteristics are shown in Table 3, and we only consider the largest connected component as in [47]. We split the network into labeled (20%) and unlabeled nodes (80%). We further equally split the labeled nodes into training and validation sets to train our surrogate model. The datasets used in this paper can be found in <https://github.com/DSE-MSU/DeepRobust>.

D.2 Implementations of Attack Models

All baselines are initialized with same parameters suggested by their papers and we also further carefully turn parameters to get optimal performance. For the greedy-based baselines (i.e., Rand, DICE, DICE-t, FGA and Nettack), we set the maximum epoch as 1000, where they can consistently add perturbations until target node v is misclassified or the number of perturbed edges is more

Algorithm 1 MiBTack

Require: The target node v , the trained GNN model f , patience P , the initial step size α and β .

Ensure: The minimal-budget adversarial perturbations δ_v^* .

```

1:  $i \leftarrow 1, \Delta_v^{(0)} = 1$ .
2: Approximate the closet boundary  $c^*$  via Eq. 13, then initialize
    $\delta_v^{(0)}$  with  $\delta_v^{c^*}$ .
3: while  $P \leq 0$  do
4:   Update  $\delta_v^{(i-1)}$  to  $\tilde{\delta}_v^{(i)}$  with gradient descent with Eq. 7 and
     8.
5:   Project  $\tilde{\delta}_v^{(i)}$  for the constraint of  $\|\delta_v^{(i)}\|_0 \leq \Delta_v^{(i)}$  by Eq. 9.
6:   Obtain the perturbed adjacency vector by  $\mathbf{a}'_v^{(i)} \leftarrow \mathbf{a}_v +$ 
      $\mathcal{T}(\delta_v^{(i)})$ .
7:   if  $\mathcal{L}(\text{Proj}_{\{0,1\}^N}[\mathbf{a}'_v^{(i)}]) < 0$  then
8:     Reduce budget to  $\Delta_v^{(i+1)}$  with Eq. 10.
9:     if  $\|\delta_v^{(i)}\|_0 < \|\delta_v^*\|_0$  then
10:       $\delta_v^* \leftarrow \delta_v^{(i)}$ .
11:     end if
12:   else
13:     Enlarge budget to  $\Delta_v^{(i+1)}$  with Eq. 11.
14:   end if
15:   Reduce patience  $P$  and step size  $\alpha$  and  $\beta$  if the perturbed
     node has reached decision boundary.
16: end while
17: return Minimum perturbation  $\delta_v^*$  and  $\Delta_v^* = \|\delta_v^*\|_0$ .

```

than 1000. For our model, we set the patience P as 800, where the step size α for updating $\delta_v^{(i-1)}$ is 1.0 and β for updating $\Delta_v^{(i)}$ is 0.1.

D.3 Experiments Settings

All experiments are conducted with following setting:

- Operating system: CentOS Linux release 7.7.1908(Core)
- CPU: Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz
- GPU: GeForce RTX 2080 Ti
- Software versions: Python 3.8.5; Pytorch 1.8.0; Numpy 1.19.2; SciPy 1.5.4; NetworkX 2.5; Scikit-learn 0.23.2; dgl 0.5.3; torchsparse 0.6.12;