SAM: Query-Efficient Adversarial Attacks Against Graph Neural Networks

CHENHAN ZHANG, University of Technology Sydney, Australia
SHIYAO ZHANG, Southern University of Science and Technology, China
JAMES J.Q. YU, University of York, United Kingdom
SHUI YU, University of Technology Sydney, Australia

Recent studies indicate that Graph Neural Networks (GNNs) are vulnerable to adversarial attacks. Particularly, adversarially perturbing the graph structure, e.g., flipping edges, can lead to salient degeneration of GNNs' accuracy. In general, efficiency and stealthiness are two significant metrics to evaluate an attack method in practical use. However, most prevailing graph structure-based attack methods are query-intensive, which impacts their practical use. Furthermore, while the stealthiness of perturbations has been discussed in previous studies, the majority of them focus on the attack scenario targeting a single node. To fill the research gap, we present a global attack method against GNNs, **S**aturation adversarial **A**ttack with **M**eta-gradient (**SAM**), in this paper. We first propose an enhanced meta-learning-based optimization method to obtain useful gradient information concerning graph structural perturbations. Then, leveraging the notion of saturation attack, we devise an effective algorithm to determine the perturbations based on the derived meta-gradients. Meanwhile, to ensure stealthiness, we introduce a similarity constraint to suppress the number of perturbed edges. Thorough experiments demonstrate that our method can effectively depreciate the accuracy of GNNs with a small number of queries. While achieving a higher misclassification rate, we also show that the perturbations developed by our method are not noticeable.

CCS Concepts: • **Security and privacy** → *Social network security and privacy*; • **Information systems** → Graph-based database models; • **Computing methodologies** → *Neural networks.*

Additional Key Words and Phrases: Adversarial attack, poisoning attack, graph neural network, topology attack.

ACM Reference Format:
Chenhan Zhang, Shiyao Zhang, James J.Q. Yu, and Shui Yu. 2023. SAM: Query-Efficient Adversarial Attacks Against Graph Neural Networks. *J. ACM* 1, 1, Article 111 (January 2023), 19 pages. https://doi.org/nn.nnnn/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recent years have witnessed significant successes brought by deep learning (DL)-based models on graph data, and the most attractable branches are graph neural networks (GNN) [33]. Compared with conventional approaches, GNNs can utilize advanced learning techniques, such as graph

This research was supported by the Australian Research Council (ARC) DP200101374. James J.Q. Yu is also with the Southern University of Science and Technology.

Authors' addresses: Chenhan Zhang, chenhan.zhang@student.uts.edu.au, University of Technology Sydney, 15 Broadway, Ultimo 2007, Sydney, NSW, Australia; Shiyao Zhang, syzhang@ieee.org, Southern University of Science and Technology, 1088 Xueyuan Blvd, Nanshan Qu, Shenzhen Shi, China; James J.Q. Yu, yujq3@sustech.edu.cn, University of York, Heslington, York YO10 5DD, York, United Kingdom; Shui Yu, shui.yu@uts.edu.au, University of Technology Sydney, 15 Broadway, Ultimo NSW 2007, Sydney, NSW, Australia.

0004-5411/2023/1-ART111 $15.00
https://doi.org/nn.nnnn/nnnnnnn.nnnnnnn

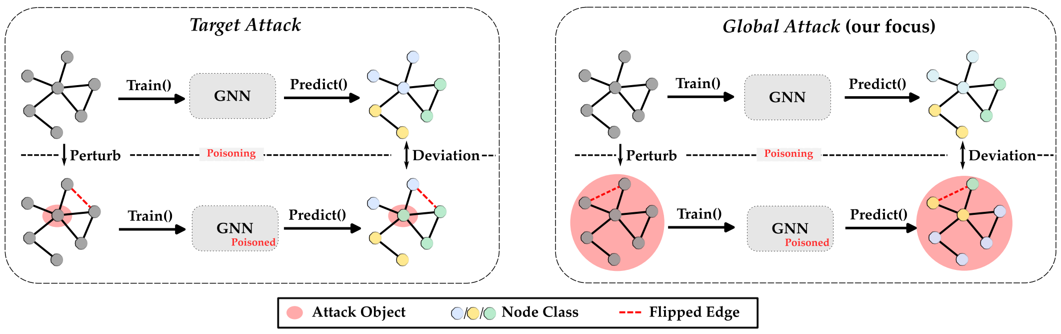


Fig. 1. Graph structure-based poisoning attacks against node classification tasks: *Target* vs *Global*.

convolution, to extract latent information from nodes' neighbors, achieving higher accuracy on downstream tasks such as node classification [34].

The DL nature of GNNs makes them vulnerable to adversarial attacks, regardless of their powerful learning and reasoning capability over graph data — which raises potential security concerns when applying GNNs to security-critical tasks [23, 33]. In general, the adversarial attacks on graph data can be categorized into *attribute perturbation-based* and *structural perturbation-based* methods. Attribute perturbation-based attacks follow the adversarial attacks on images [8] that attempt to perturb the attributes of nodes or edges to make GNNs misclassify [25]. However, the limitation is obvious that such attacks apply only to attributed graphs that have features associated with each node or edge. Differently, structural perturbation-based attacks modify the graph structure. As graph-based methods usually utilize the nodes' relational knowledge, the change of graph structure have been demonstrated to be effective in attacking conventional graph-based methods for graph tasks, such as collective classification-based methods [28]. Such a vulnerability has also been found in GNNs — their training efforts or reasoning performance can be deteriorated by the topology change. This discovery prompts a number of studies following the paradigm of structural perturbation-based attacks [36, 38, 42].

In this work, we investigate the vulnerability of GNNs to structural perturbation-based adversarial attacks from a poisoning attack perspective. The majority of existing studies center on the *target attack* that only attempts to compromise the classification accuracy on a single node [15, 38, 42]. Differently, we focus on the *global attack*, wherein the objective is to undermine the node classification performance of the models across all nodes in a graph (see Figure 1). We particularly consider the *grey-box setting*, where the attacker has limited knowledge of the model, posing more severe threats to model security.

Motivations. Query limitation is unheeded in the majority of existing studies in terms of graph-based adversarial attacks [36, 42, 43]. Unlike white-box settings, where attackers have full knowledge of the target model assets, black- and grey-box settings typically require query-intensive attacks to gather useful information [39]. However, many online systems impose restrictions on the number of queries that can be submitted within a given time frame; for example, Google's cloud vision API only allows approximately 1800 queries per minute [3]. Inefficient query leads to unit-time inefficiency and weakens attackers' large-scale attacks. Furthermore, these systems usually tend to flag the behavior of submitting a large number of similar queries within a short period as a *malicious* one [6]. These realistic limitations compel attackers to devise query-efficient strategies — launch effective attacks within a limited number of queries.

On the other hand, adversarial attacks are developed to be inconspicuous that can keep undiscovered and thus survive. As aforementioned, the majority of works in this domain focus on target node attacks: the attack only changes the local topology (within one or several hops of the target node), which is usually unnoticeable [38]. However, in terms of global attacks, a greater number of topological perturbations are usually involved to ensure an effective attack. Naturally, more concerns are here: how to make the global attack as stealthy as possible. So far, the stealthiness of global attacks is still an open problem to address [23].

Our Contributions. To address the above challenges, we propose **S**aturation adversarial **A**ttack with **M**eta-gradient (SAM) for adversarial topology attacks. The term "saturation attack" originates from a military tactic that aims to gain an advantage through massive and overwhelming actions within short timeframes. SAM is designed with the concepts of meta-learning [20] and saturation attack [30], and efficiency and stealthiness are the main advantages of SAM. We first propose an enhanced meta-learning-based optimization approach to develop meta-gradients associated with the graph structure, aligned with our attack objectives. Particularly, the optimization approach for meta-gradients is designed towards three goals— high training loss, high fooling capacity, and high smoothness of graph data's features and structure. Second, we design a Top-k Flipping algorithm inspired by the concept of saturation attack to decide the edges to be flipped based on the derived meta-gradients. Additionally, considering the attacking stealthiness, we incorporate a similarity constraint in the Top-k Flipping algorithm to limit the number of perturbed changes and preserve the connectivity of feature-similar nodes as much as possible. By combining the strengths of optimization and heuristic algorithms, our proposed method encompasses the advantages of both approaches. Our main contributions are summarized as follows:

- We propose a meta-learning-based approach to the global attack against graph node classification. We refine the optimization strategy of graph structures' meta-gradients, which can derive meaningful meta-gradients for constructing effective graph structural perturbations for the attack goal.
- We leverage the concept of saturation attack and design an edge flipping algorithm named Top-k Flipping. This algorithm flips multiple edges upon the computed meta-gradients in one query to improve attack efficiency. To make the attack less noticeable, we incorporate a similarity-based constraint to suppress the number of flipping edges and preserve the connectivity of nodes with similar features.
- We design a comprehensive evaluation procedure on three real-world graph datasets to show the efficacy of the proposed method. The proposed method achieves more query-efficient attacks than state-of-the-art, and the perturbations to graph structures are unobvious meanwhile.[1]

The remainder of this paper is organized as follows. In Section 2, we review related literature about adversarial attacks on graph data. We introduce some preliminary knowledge and the threat model in Section 3. In Section 4, we elaborate on the proposed approach. Section 5 presents the experimental results. We conclude in Section 7 with a discussion of future work.

2 RELATED WORK

This section first reviews the related work on adversarial attacks on graph data. Then the difference between existing works and our work is also discussed.

[1]The source code repository of this work is available at https://github.com/CHZ-democode/SAM.

2.1 Adversarial Attacks

The adversarial attack is recognized as a big threat to the security of deep models. The earlier studies around neural networks' robustness laid a foundation for the recent development of adversarial attacks. Szegedy *et al.* [24] indicated that the specifically fabricated inputs with a small magnitude of perturbations could markedly degrade the accuracy of a well-trained deep neural network. Goodfellow *et al.* [8] investigated the phenomenon and proposed a gradient-based method to generate adversarial image examples. Moosavi-Dezfooli *et al.* [16] proposed to develop a minimal norm adversarial perturbation, and they further found that there exist "universal" adversarial perturbations that can fool a model on different data samples. Myriads of adversarial attack models or strategies have then followed up to demonstrate the vulnerabilities of DNNs in various settings and applications [21].

2.2 Adversarial Attacks on Graph Data

Recent years have witnessed the increasing popularity of deep graph learning such as Graph Neural Networks (GNNs) due to the capacity to exploit complicated relationships in graph data [35]. The popularization of GNNs in the industry raise the security and privacy concerns about them [7, 10]. The seminal work of extending the notion of adversarial attack to the graph domain can be referred to [4, 43]. Reference [43] investigated both the poisoning and evasion attacks on node classifications by proposing a greedy incremental computation-based approach via modifying the graph structure to increase the loss. Comparatively, Reference [4] only considered the evasion attack and devised a reinforcement learning-based approach to downgrade the test accuracy. [1] proposed a poisoning attack against random walk-based models and demonstrated the transferability of the attack across different models. Reference [29] identified the limitation to attack multi-layer GNNs and proposed a corresponding optimization-based attack method. Reference [43] leveraged meta-learning and treated the graph structure as a hyperparameter to assist in seeking the graph perturbation. Reference [36] treated graph structural perturbation as continuous variables by using convex relaxation to optimize the perturbation.

Our proposed method builds upon the concept of meta gradient-based attacks in [43] but offers refinement from different perspectives to improve the efficiency and stealthiness, including the design of objective function design, the adoption of a saturated strategy of perturbation decision, and the incorporation of similarity constraint. Besides, our proposed method differs from works such as [38] and [36], which directly apply convex relaxation to optimize graph structure perturbations. The practicality of these approaches may be debatable and can depend on the quality of their implementation.

3 PRELIMINARY

In this section, we first introduce related definitions of graph data, tasks, and GNNs. Then, we give the basic definition of graph structure-based poisoning attacks. Lastly, we define the threat model to be investigated in this work.

3.1 Graph Data, Node Classification Task, and GNNs

Formally, let $G = (\mathcal{V}, \mathcal{E})$ denote an undirected and unweighted graph, where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the edge set. The adjacency matrix of G is $\mathbf{A} = \{a_{ij}\} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ where we have the entry $a_{ij} = 1$ if edge $(i, j) \in \mathcal{E}$ and $a_{ij} = 0$ otherwise.

We study the semi-supervised node classification task in this work. The graph G is assumed to be attributed, and we denote the node attribute (feature) matrix by $\mathbf{X} = \{x_i\} \in \mathbb{R}^{|\mathcal{V}|\times D}$ where D is the dimension of the node feature vector. Thus, the attributed graph G can be also denoted by

Table 1. Summary of Symbols

Symbol	Explanation
G	Original graph.
G'	Perturbed graph.
$v, \mathcal{V}$	Node set.
$\mathcal{E}$	Edge set.
(i, j)	The edge connecting node i and j.
$\mathbf{A}$	Adjacency matrix.
$\mathbf{A}'$	Perturbed adjacency matrix.
$\mathbf{P}$	Perturbation matrix.
$\nabla_{\mathbf{A}}^{\text{meta}}$	Meta-gradient of adjacency matrix.
$\mathbf{X}$	Node feature (attribute) matrix.
$\mathbf{W}$	Weight matrix of GNN.
α	Balanced coefficient α for training loss on labeled and unlabeled nodes (learnable).
β	Coefficient for fooling loss term (learnable).
γ	Coefficient for similarity loss term (learnable).
$\underline{\varrho}, \overline{\varrho}$	Lower and upper bounds for learnable coefficients.
$\mathbf{S}$	Similarity matrix.
T	Number of queries.
E	Number of training epochs for GNN.
k	Number of flipped edges.
ζ, λ	Threshold and its determining hyperparameter.
ι	Budget limiting the number of flipped edges.

$G = (\mathbf{A}, \mathbf{X})$. Given the set of labeled nodes $\mathcal{V}_{\text{lab}}$ and the labels $\mathbf{Y} \in \mathbb{R}^{|\mathcal{V}_{\text{lab}}|}$, the task aims to learn a GNN f that can map the class of each unlabeled node to the exact one. Particularly, the GNN in this work is set as a 2-layer GCN model [12], which can be formulated as

$$Z = f(\mathbf{A}, \mathbf{X}) = \text{softmax}\left(\hat{\mathbf{A}}\,\text{ReLU}\left(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(1)}\right)\mathbf{W}^{(2)}\right), \tag{1}$$

where $\hat{\mathbf{A}}$ is the normalized Laplacian adjacency matrix; $\mathbf{W}^{(l)}$ is the trainable weight matrix at layer (l); Z is the output of GCN, which indicate the probability of mapping the node to a specific class. The training loss is computed by cross-entropy between the true labels and the predicted labels [13].

3.2 Graph Structure-based Poisoning Attack

In this work, we investigate poisoning attacks using structural perturbations. In particular, edge flipping is leveraged to perturb the graph topology; therefore, we introduce a perturbation matrix $\mathbf{P}$ to perturb the original adjacency matrix used for training. $\mathbf{P} = \{p_{ij}\} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is a boolean symmetric matrix where $p_{ij} = 1$ means edge (i, j) is flipped and otherwise remains unchanged. The perturbed adjacency matrix can be computed by

$$\mathbf{A}' = (\mathbf{1} - \mathbf{P}) \odot \mathbf{A} + \mathbf{P} \odot (\mathbf{1}_0 - \mathbf{A}), \tag{2}$$

where $\odot$ represents the Hadamard product; $\mathbf{1} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the all-ones matrix and $\mathbf{1}_0$ is the all-ones matrix with zeros on the main diagonal.

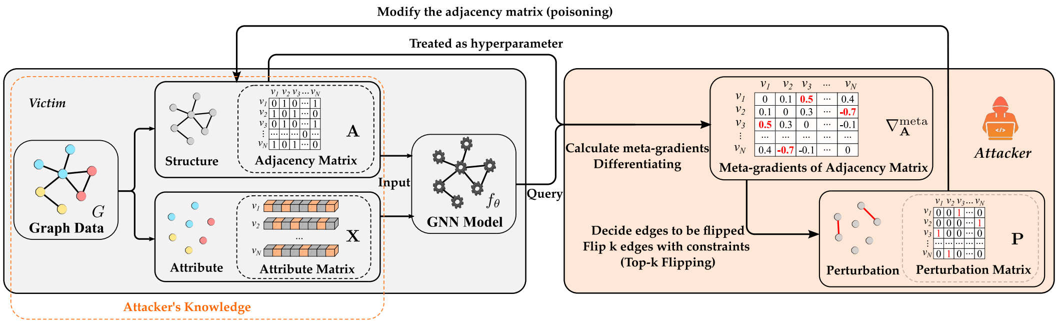


Fig. 2. Schematic of SAM. The left block presents the paradigm of GNN training. The right block describes the pipeline of graph structure-based poisoning attacks.

We recognize that optimization-based attacks can result in a noticeable decline in classification performance. Therefore, we follow [18] that the poisoning attack is formulated as a bilevel optimization problem, which can be defined as

$$\begin{aligned} \underset{\Theta}{\textbf{minimize}} \quad & \mathcal{L}_{\text{atk}}\left(f_{\theta^*}(G')\right) \\ \textbf{subject to} \quad & \theta^* = \underset{\theta}{\arg\min}\ \mathcal{L}_{\text{tra}}\left(f_{\theta}(G')\right), \end{aligned} \tag{3}$$

where $\mathcal{L}_{\text{atk}}$ denotes the objective function that the attacker tries to optimize and $\mathcal{L}_{\text{tra}}$ denotes the training loss; G' represents the perturbed graph data and we only consider the topology attack in this work, therefore, we have $G' = (\mathbf{A}', \mathbf{X})$ where $\mathbf{X}$ is fixed during the optimization process; θ denotes all learnable parameters of the model, i.e., $\theta = (\mathbf{W}^{(0)}, \mathbf{W}^{(1)})$ for the adopted GCN, and θ^* denotes the optimized parameter by minimizing $\mathcal{L}_{\text{tra}}$; Θ denotes the set of all optimizable variables. The bilevel solver first updates the model parameters θ with the perturbed graph and fixes it as θ^*. Then, we seek to find $\mathbf{A}'$ developed by $\mathbf{P}$ via Eq. (2) that increases the classification loss with θ^*. Furthermore, it has been shown that the negative training loss can be used as the attack loss, i.e., $\mathcal{L}_{\text{atk}} = -\mathcal{L}_{\text{tra}}$ [8, 13] — this notion will be used to devise our objective function in the sequel.

3.3 Threat Model

We refer to the assumption of the attacker in [43]. The attacker conducts global attacks and aims to increase the overall misclassification rate (MCR) of a model that makes the test data samples classified as any class different from the real one. However, different from works like [43], the attacker in our study also cares about the attack success rate (ASR). Unlike MCR, ASR measures the level that the classification decision of attacked model is different from the original model, which is a meaningful metric to the attack effectiveness [23].

In this work, we consider a grey-box attack setting — attackers know limited knowledge to attack the model, including all nodes' attributes and the graph structure; however, they have no knowledge about the target model's structure and weights. In this setting, the attackers employ a *querying* strategy to obtain the model's information after each routine training iteration [3], and later modify the graph structure input to the model with the perturbed one. Additionally, to make the attack more difficult to be recognized, the attackers ensure that the changes to the graph structure are not prominently noticeable.

4 SATURATION ADVERSARIAL ATTACK WITH META-GRADIENT (SAM)

Referring to the meta-learning-based method proposed in [43], one effective approach to topology attack is by *learning* a perturbed graph structure. We construct the basic architecture of our proposed approach based on [43] (See Figure 2). Specifically, we first parameterize the adjacency matrix $\mathbf{A}$. Fixing the feature matrix in Eq. (3), the meta-gradient of $\mathbf{A}$ can be differentiated by

$$\nabla_{\mathbf{A}}^{\text{meta}} = \nabla_{\mathbf{A}} \mathcal{L}_{\text{atk}} \left(f_{\theta^*}(G) \right), \tag{4}$$

where $\nabla_{\mathbf{A}}^{\text{meta}} = \{\nabla_{a_{ij}}^{\text{meta}}\} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$; θ^* can be obtained by $\theta^* = \arg\min_{\theta} \mathcal{L}_{\text{tra}} \left(f_\theta(G) \right)$. In this way, the update of parameter θ will be correlated to the graph structure. Then, we can utilize the meta-gradient of graph structure to find the fittest perturbation, i.e., which edge to be flipped, by a decision-making algorithm $\text{Dec}(\cdot)$, i.e., $\mathbf{P} = \text{Dec}(\nabla_{\mathbf{A}}^{\text{meta}})$.

This approach implies two challenges to be addressed. The first one is how to construct the objective function $\mathcal{L}_{\text{atk}}$ that we can derivate useful meta-gradients for the edge flipping. The second one is how to design an algorithm that reasonably determines the edges to be flipped. To achieve our efficient and stealthy attacking purpose, the two challenges are resolved as follows.

4.1 Meta-gradient-based Optimization

We treat meta-gradients' derivation as an optimization problem. The corresponding objective function integrates three meaningful loss terms concerning different perspectives, which are elaborated on in turn.

4.1.1 Training Loss Term. The investigated semi-supervised node classification task is a type of transductive learning. In this case, the training loss of the model on the nodes can be the one on the labeled data, i.e, $\mathcal{L}_{\text{lab}} = \mathcal{L}_{\text{ce}} \left(f_{\theta^*}(G'_{\mathcal{V}_{\text{lab}}}), \text{Y} \right)$ and the one on the unlabeled data, i.e., $\mathcal{L}_{\text{unl}} = \mathcal{L}_{\text{ce}} \left(f_{\theta^*}(G'_{\mathcal{V}_{\text{unl}}}), \text{Y}_{\text{fake}} \right)$, where $\mathcal{L}_{\text{ce}}$ denotes the cross entropy loss and $\mathcal{V}_{\text{unl}} = \mathcal{V} \backslash \mathcal{V}_{\text{lab}}$ is the unlabeled node set; since the labels of unlabled nodes are not available, we adopt Pseudo-Label algorithm [14] to generate the fake labels Y_{fake} for the unlabled nodes by

$$\text{Y}_{\text{fake}} = \arg\max \ f_{\theta^*}(G_{\mathcal{V}_{\text{unl}}}). \tag{5}$$

We recognize the significance of the training loss of both labeled and unlabeled node sets and thus consider the trade-off between the two types of loss by introducing a balanced coefficient α. The combination of $\mathcal{L}_{\text{lab}}$ and $\mathcal{L}_{\text{unl}}$, denoted as $\mathcal{L}_{\text{a-tra}}$, can be proposed as

$$\begin{aligned} \underset{\Theta}{\textbf{maximize}} \quad & \mathcal{L}_{\text{a-tra}} = \alpha \mathcal{L}_{\text{lab}} + (1-\alpha)\mathcal{L}_{\text{unl}} \\ \textbf{subject to} \quad & \underline{\varrho}^{\alpha} \leq \alpha < \overline{\varrho}^{\alpha}. \end{aligned} \tag{6}$$

Particularly, unlike previous work that preinstalls the value of α [43], we treat it as a learnable hyperparameter, and we have $\alpha \in \Theta$. Furthermore, we have tried that optimizing α without constraints will render unstable attack performance appearing in iterations (see Section 5.2.4); hence, we set bound to α where $\underline{\varrho}$ and $\overline{\varrho}$ denote the lower bound and the upper bound, respectively.

4.1.2 Fooling Loss Term. In addition to the negative training loss as introduced in Section 4.1.1, we introduce a fooling loss term to further improve the attack effect. We consider a successful attack means that the predicted class of a node with the perturbed graph structure $\mathbf{A}'$ is different from the one predicted using the original graph structure $\mathbf{A}$, i.e., $f_{\theta^*}(G'_{\mathbf{A}'}) \neq f_{\theta^*}(G_{\mathbf{A}})$. In this situation, we say that the model is fooled by the perturbed graph structure. The fooling loss term is designed to

Algorithm 1: Saturation adversarial Attack with Meta-gradient (SAM)

```
Input: adjacency matrix A, node feature matrix X, number of flipped edges $k$, threshold
       hyperparameter $\lambda$, number of queries $T$, similarity score matrix S, learning rates $\eta_1$
       and $\eta_2$, GNN $f$ with parameter $\theta$, number of training epochs for GNN $E$
Output: Perturbed adjacency matrix $\mathbf{A}'$
1  $\Theta \leftarrow$ Parameterize A, $\alpha$, $\beta$ and $\gamma$
2  Initialize $\Theta$ and $\theta$
3  $t \leftarrow 0$
4  $\mathbf{S} \leftarrow$ CosineSimilarity(X)
5  while $t \leq T$ do
6      if $\frac{|\mathbf{A}'-\mathbf{A}|}{2|\mathcal{E}|} \leq \iota$ then
7          foreach epoch $i \in E$ do
8              $\theta^{(i+1)} \leftarrow \theta^{(i)} - \eta_1 \cdot \nabla \mathcal{L}_{\text{tra}}$
9          $\nabla^{\text{meta}}_{\mathbf{A}}, \nabla^{\text{meta}}_{\alpha}, \nabla^{\text{meta}}_{\beta}, \nabla^{\text{meta}}_{\gamma} \leftarrow$ GetGradients($\mathcal{L}_{\text{atk}}$)
10         $\alpha^{(t+1)} \leftarrow \alpha^{(t)} - \eta_2 \cdot \nabla^{\text{meta}}_{\alpha}$
11         $\beta^{(t+1)} \leftarrow \beta^{(t)} - \eta_2 \cdot \nabla^{\text{meta}}_{\beta}$
12         $\gamma^{(t+1)} \leftarrow \gamma^{(t)} - \eta_2 \cdot \nabla^{\text{meta}}_{\gamma}$
13         $\mathbf{A}' \leftarrow$ TKF($\nabla^{\text{meta}}_{\mathbf{A}}$, S, $k$, $\lambda$, $t$)  // Refer TKF to Algorithm 2
```

emphasize the degree of fooling by the perturbed graph structure, which is formulated as

$$\textbf{maximize} \quad \mathcal{L}_{\text{foo}} = \mathcal{L}_{\text{Huber}-\delta}(f_{\theta^*}(G'_{\mathbf{A}'}), f_{\theta^*}(G_{\mathbf{A}})) \tag{7}$$

where $\mathcal{L}_{\text{Huber}-\delta}$ denotes the Huber loss with error δ, which is adopted here to calculate the regression loss. We set $\delta = 1$ as a matter of experience. By maximizing this loss term, we can obtain meta-gradients reflecting the fooling capacity of graph structures.

4.1.3 Similarity Loss Term. Furthermore, we seek to ensure the natural connectivity of the perturbed graph since edge flipping may cause fragmentation and unnatural connectivity pattern. Following the common pattern in real-world graphs that adjacent nodes are more likely to have similar attributes, we introduce a similarity loss term, which is designed as

$$\begin{aligned} &\textbf{minimize} \quad \mathcal{L}_{\text{sim}} := \frac{1}{2}\sum_{i=1}^{|\mathcal{V}|}\sum_{j=1}^{|\mathcal{V}|} a_{ij}\left(x_i - x_j\right)^2 := \text{Tr}(\mathbf{X}^T\hat{\mathbf{L}}\mathbf{X}) \\ &\textbf{subject to} \quad \mathbf{A} = \mathbf{A}^T, \end{aligned} \tag{8}$$

where $\text{Tr}(\cdot)$ denotes the trace of a matrix; $\mathbf{X}_i$ denotes the feature vector of node i; $\hat{\mathbf{L}}$ is the normalized Laplacian of A, which is adopted here for making feature smoothness independent of node degrees [41]. Given a node pair (i, j), their feature difference, i.e., $x_i - x_j$, is fixed since the node feature is assumed to be unchanged in this work; however, A varies from each round of the proposed method. We can easily infer that this term is minimized when there exist massive pairs such that a_{ij} is 0 and $x_i - x_j$ is large, which accords with our purpose. The connectivity consideration will also be embodied in the similarity constraint in our edge flipping algorithm (see Section 4.2 later).

```
Algorithm 2: Top-$k$ Flipping (TKF)
   Input: adjacency matrix $\mathbf{A}$, similarity score matrix $\mathbf{S}$, threshold hyperparameter $\lambda$, $t$
   Output: Perturbed adjacency matrix $\mathbf{A}'$
1  $\mathbf{P} \leftarrow 0$
2  $\mathbf{M} \leftarrow$ Map $\nabla_{\mathbf{A}}^{\text{meta}}$ by Eq. (10)
3  $k^* \leftarrow k/\sqrt{t}$
4  $\mathcal{P}_{\text{cand}} \leftarrow$ Find $k^*$ largest entries of $\mathbf{M}$
5  $\mathbf{P} \leftarrow$ Obtain thresholded perturbation by Eq. (12)
6  $\mathbf{A}' \leftarrow$ Obtain perturbed adjacency matrix by Eq. (2)
```

The objective of the overall optimization problem of SAM is to minimize the attack loss $\mathcal{L}_{\text{atk}}$, considering the three loss functions in Eq. (6), (7), and (8). Finally, the problem can be formulated as

$$\begin{aligned}
\underset{\Theta}{\textbf{minimize}} \quad & \mathcal{L}_{\text{atk}} = -\mathcal{L}_{\text{a-tra}} - \beta\mathcal{L}_{\text{foo}} + \gamma\mathcal{L}_{\text{sim}} \\
\textbf{subject to} \quad & \theta^* = \arg\min_{\theta} \ \mathcal{L}_{\text{tra}}\left(f_\theta(G')\right) \\
& \{\alpha, \beta, \gamma\} \in \Theta \\
& \underline{\varrho}^{\alpha} \le \alpha < \overline{\varrho}^{\alpha} \\
& \underline{\varrho}^{\beta} \le \beta < \overline{\varrho}^{\beta} \\
& \underline{\varrho}^{\gamma} \le \gamma < \overline{\varrho}^{\gamma} \\
& \mathbf{A} = \mathbf{A}^T,
\end{aligned} \tag{9}$$

where β and γ are variables that control the contribution of feature similarity and fooling level; likewise as α, we treat β and γ as bounded learnable control variables.

Particularly, the optimal solutions for α, β, and γ are obtained from Eq. (9) by gradient descent method. For $\mathbf{A}$, we only compute its gradient (i.e., $\nabla_{\mathbf{A}}^{\text{meta}}$) at each iteration and further feed it into the proposed Top-k Flipping algorithm to decide which edges to be flipped. We additionally adopt the circuit-breaker mechanism that terminates the attack when the number of perturbed edges reaches a maximum threshold, which is defined as $\frac{|\mathbf{A}'-\mathbf{A}|}{2|\mathcal{E}|} \le \iota$.

Convergence. As shown in Eq. (9), the objective function consists three parts, namely, $\mathcal{L}_{\text{a-tra}}$, $\mathcal{L}_{\text{foo}}$, and $\mathcal{L}_{\text{sim}}$. Considering function $\mathcal{L}_{\text{a-tra}}$ and $\mathcal{L}_{\text{foo}}$ as a whole, it follows a multi-task learning setting; by taking the second derivative of this function with respect to Θ, the function is continuously differentiable. In addition, it can be proved that the local optimal solution is obtained with a fast convergence by following [20]. On the other hand, for the function $\mathcal{L}_{\text{sim}}$, the existence of the optimal solution can be easily verified since this function is affine and convex when $x_i - x_j$ is fixed. Besides, it can be validated that the feasible set of the formulated problem in Eq. (9) is closed and convex. Then, according to [2], since the objective function in Eq. (9) is continuously differentiable, the existence of the local-optimal solution can also be guaranteed in Eq. (9) in a similar way. Therefore, we can conclude that the problem formulated in Eq. (9) can reach the local-optimal solution Θ^*. We summarize the procedure presented thus far in Algorithm 1.

4.2 Top-k Flipping with Similarity Constraints

We devise a decision-making algorithm named Top-k Flipping to determine the edges to be flipped given the calculated meta-gradients. We first map the meta-gradients into a meta score matrix

$\mathbf{M} = \{m_{ij}\} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ by

$$\mathbf{m}_{ij} = \begin{cases} \nabla^{\text{meta}}_{a_{ij}}, & a_{ij} = 0 \\ -\nabla^{\text{meta}}_{a_{ij}}, & a_{ij} = 1 \end{cases}. \tag{10}$$

Consider a pair of connected nodes (i, j), denoted by $a_{ij} = 1$. If the meta-gradient $\nabla^{\text{meta}}_{a_{ij}}$ is notably negative, it implies that the corresponding edge has a negative effect on our objective. Thus, we want to remove the edge by changing the value of a_{ij} to 0. In accordance with Equation (10), we reverse the sign of the meta-gradient to obtain a positive meta score. As a result, the corresponding edge is more likely to be flipped during our Top-k flipping algorithm. Conversely, if (i, j) has a considerably large positive $\nabla^{\text{meta}}_{a_{ij}}$, it indicates that the edge contributes positively to our objective. By flipping $\nabla^{\text{meta}}_{a_{ij}}$, a negative meta score is produced, making it more probable for the edge to be retained in our Top-k flipping algorithm. Based on the notion of saturation attack, we assume that simultaneously flipping multiple edges in a query can significantly improve the attack efficiency. Thereupon, $\mathbf{M}$ is sorted and the corresponding edges of the largest k entries are considered as the flipping candidates, which is denoted by $\mathcal{P}_{\text{cand}} = \{p_1, p_2, \ldots, p_k\}$ where p represents an edge (i, j). In particular, k is varied with regards to the query t where we have $k^* = \frac{k}{\sqrt{t}}$ — this is to warrant the attack effect in the beginning rounds of the query.

Meanwhile, we involve a similarity constraint in the process of edge flipping. Particularly, we adopt cosine similarity to measure the feature similarity between any two nodes in the graph, which is formulated as

$$s_{ij} = \frac{\sum_{u=1}^{l} x_{i,u} x_{j,u}}{\sqrt{\sum_{u=1}^{l} x_{i,u}^2}\sqrt{\sum_{u=1}^{l} x_{j,u}^2}}, \tag{11}$$

where $x_{i,u}$ and $x_{j,u}$ denote the scalars of x_i and x_j respectively, D is x_i's dimension, and u is the index; $\mathrm{S} = \{s_{ij}\} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is a similarity matrix that stores the calculated similarity score. Note the nature of the cosine similarity metric — the scoring between any two total factor vectors is symmetric, i.e., $s_{ij} = s_{ji}$. We introduce a similarity threshold ζ, which is correlated to the mean μ and standard deviation σ of the similarity scores of node pairs, defined by $\zeta = \mu + \lambda\sigma$; where λ is a predefined hyperparameter to control the level of the threshold ζ. For all $p \in \mathcal{P}_{\text{cand}}$, we have

$$p_{ij} = \begin{cases} 0, & s_{ij} > \zeta \ \wedge \ a_{ij} = 1 \\ 1, & \text{otherwise,} \end{cases} \tag{12}$$

In this way, we attempt to ensure that any connected two nodes with similar attributes are not disconnected during our attack. Finally, we can compute the perturbed adjacency matrix by Eq. (2). It is worth noting that since Top-k Flipping with similarity constraint tries to preserve the connectivity of nodes with similar characteristics, our approach is more applicable to graph data that are consistent with this characteristic, such as citation networks and social networks. The procedure of Top-k Flipping is summarized in Algorithm 2.

4.3 Discussion on the Attack Capability of SAM

In this subsection, we discuss the attack capability of SAM. SAM incorporates two main components: (1) meta-gradient-based optimization and (2) Top-k flipping with similarity constraints. In the meta-gradient-based optimization, we treat the adjacency matrix (i.e., graph structure) as a learnable hyperparameter of GNNs and compute the corresponding meta-gradients in the backpropagation procedure. The interpretation and utilization of the meta-gradients of to-be-perturbed data have been evaluated for adversarial attacks [37, 43]. In the context of adversarial attacks on GNNs, based on [43], we observe the direction of the meta-gradients during the attack optimization procedure

Table 2. Statistical summary of Cora, Citeseer, and ACM datasets.

	# Node	# Edge	Density	# Class	# Feature	Train/Val/Test
Cora	2708	5278	$14.3e-4$	7	1433	
Citeseer	3312	4536	$8.2e-4$	6	3703	140/500/1000
ACM	3025	13128	$28.0e-4$	3	1870	

corresponding to the adjacency matrix entries (i.e., edges). Based on this information, we identify the steepest entries as candidates to be flipped as they are the most deterministic ones to our designed attack objective and then we apply the Top-k flipping algorithm. Top-k flipping algorithm has similarity constraints, and the number of candidates to be flipped is decreasing. This enables the attack effect to be most prominent in the initial rounds of queries. Furthermore, it is important to note that we do not employ completely optimization-based methods such as projected gradient descent (PGD) to develop the perturbed adjacency matrix. Although these methods can transform the optimization problem into a continuous one and offer closed-form solutions, they usually require a large number of iterations (queries) to achieve a satisfactory attack effect [36]. This is inconsistent with our goal of achieving query efficiency. While the Top-k flipping algorithm is heuristic, SAM can also provide a sub-optimal solution that can yield effective perturbations within a reasonable number of queries. Further justification can be seen in our experiments as shown in Section 5.2.

5 PERFORMANCE EVALUATION

5.1 Experimental Setup

5.1.1 Datasets. To demonstrate our approach fairly, three widely-adopted datasets, **Cora**, **Citeseer**, and **ACM** [22, 31], are employed in our experiments. We summarize their statistical information in Table 2. Both Cora and Citeseer have a comparatively sparse graph structure (i.e., a smaller number of edges) with diversified node attributes (i.e., a larger number of node classes), while the contrary is the case of ACM. Note that the data split setting follows the one adopted in GCN's paper [12], and only the largest connected component of a graph is considered as did in [38, 43].

5.1.2 Parameter Settings. Several hyperparameters (i.e., predefined parameters) are incorporated in this work. As the default setting of subsequent experiments, we set $k = 40$ and $\lambda = 0$. We also investigate the optimal values for these hyperparameters, which will be shown later. Experiments for each setting are run repeatedly five times to eliminate randomness. The inner training epoch for GCN is set as $E = 100$ per query. Additionally, the GCN model is pretrained for 200 epochs to obtain a good initial classification capacity before our attack tests, which is regarded as the natural model. The learning rates η_1 and η_2 are both set to 0.01. For α, we adopt a conservative strategy to optimize by bounding it between $[0.5, 1]$. For β and γ, their bounds are empirically constructed according to [41] in which we let $\beta \in [0, 1]$ and $\gamma \in [0.00005, 0.00015]$.

5.1.3 Metrics. We adopt Misclassification Rate (MCR) and Attack Success Rate (ASR) as the metrics, which are respectively defined as

$$\text{ASR} = \frac{\text{\# Fooled instances}}{\text{\# All instances}} \times 100\% \tag{13}$$

$$\text{MCR} = \frac{\text{\# Misclassified instances}}{\text{\# All instances}} \times 100\%. \tag{14}$$

Table 3. ASR and MCR performance across different baselines. The results are the average of five repeated experiments. The performance on poisoning attack and evasion attack are distinguished by suffixes "PA" and "EA". The best results are highlighted in **bold**. The description of the baseline methods can be seen in Section 5.2.1.

Method	Cora		Citeseer		ACM	
	ASR	MCR	ASR	MCR	ASR	MCR
Original	–	20.14%	–	34.07%	–	13.74%
Random	1.08%	22.94%	1.40%	36.59%	0.87%	15.47%
DICE [32]	0.80%	23.01%	1.08%	36.58%	0.61%	15.55%
PGD-EA [36]	0.36%	22.67%	0.21%	36.52%	0.44%	15.38%
Meta-Both-EA [43]	1.90%	23.07%	1.85%	36.32%	2.16%	16.18%
SAM-EA	**3.10%**	**23.26%**	**4.18%**	**36.36%**	**9.46%**	**20.60%**
PGD-PA [36]	0.35%	21.06%	0.07%	32.71%	0.37%	14.46%
Meta-Both-PA [43]	1.25%	35.74%	1.19%	40.60%	0.77%	25.03%
SAM-PA	**2.47%**	**38.42%**	**1.85%**	**42.37%**	**8.01%**	**36.74%**

Table 4. Statistical summary before and after attacks by SAM.

	Cora	Citeseer	ACM
ASPL	6.31	9.31	5.77
Perturbed	6.11	9.28	6.29
Density	$14.3e^{-4}$	$8.2e^{-4}$	$28.0e^{-4}$
Perturbed	$15.0e^{-4}$	$8.6e^{-4}$	$31.1e^{-4}$
Modularity	0.646	0.745	0.771
Perturbed	0.635	0.732	0.758
# Clique	5	6	46
Perturbed	5	6	46

As aforementioned in Section 4.1, for ASR, we adopt the definition of a *successful attack* that the classification result on a node with the perturbed graph structure is different from the one with the original graph structure, no matter which result is correct[2]. Thus, ASR and MCR measure the method's effectiveness from different angles: the former emphasizes the method's fooling capacity, and the latter focuses on the method's practical attack performance [23].

All experiments are conducted on computing servers with two Intel Xeon E5 CPUs and 128 GB RAM. The approach and simulation are implemented with PyTorch. Eight nVidia GTX 2080 Ti GPUs are employed on each server for neural network computing acceleration.

5.2 Results

In this section, we carry out a series of comprehensive case studies on three real-world datasets to evaluate the efficacy of the proposed graph structure-based poisoning attacking method. In

[2]Note that the definition of ASR can vary from work; another definition assumes the target nodes are classified correctly before the attack [17].

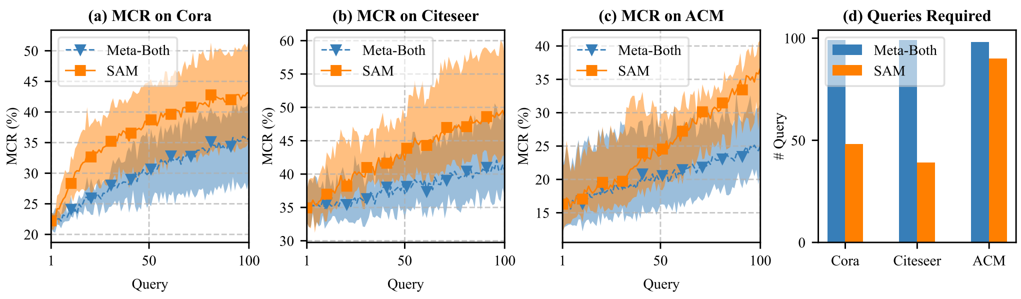


Fig. 3. MCR versus queries ((a), (b), and (c)) and the actual needed number of queries to achieve the attack goal ((d)) on different datasets.

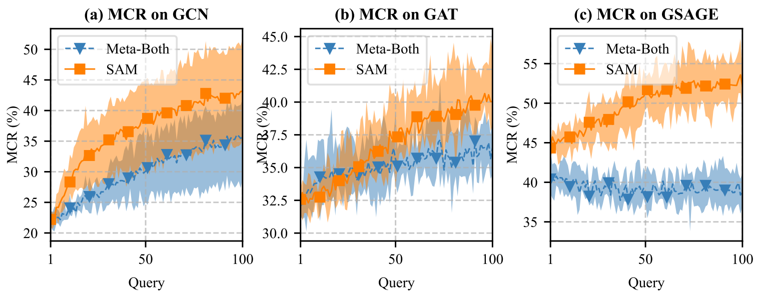


Fig. 4. MCR versus queries on different GNNs.

particular, we first compare the classification accuracy decreased by our method with other baselines. Subsequently, the attack efficiency of the proposed method is demonstrated. Then we analyze the stealthiness of the proposed method. Finally, an ablation test is conducted to evaluate the impact of hyperparameters and constraints on the proposed method.

5.2.1 Comparison with Baselines. We compare our method with four baseline methods: 1). Random: Randomly flip edges in each attack. 2). DICE [32]: Delete edges internally, connect externally; 3). PGD [36]: Projected gradient descent topology attack. 4). Meta-Both [43]: Meta-gradient topology attack[3]. For a fair comparison, we set the same performance evaluation criterion and train-test splitting (as presented in Table 2) for all methods. The maximum number of queries is set to $T = 100$. Additionally, the maximum number of perturbed edges is set to be 10% of the number of edges in the original graph, i.e., $\iota = 10\%$. Furthermore, since Random and DICE are designed as evasion attack (a.k.a. test-time attack) methods, we also show the performance of SAM on evasion attacks for comparison. Specifically, incorporating the perturbed graph structure yielded by the SAM in the last epoch as the adversarial example in the testing set, we evaluate the performance of a GNN trained on the normal example against the adversarial example. The performance of the natural model with the unperturbed topology is also presented, labeled as "Original".

[3]Meta-Both is the best-performing version of the proposed approaches in the referenced work.

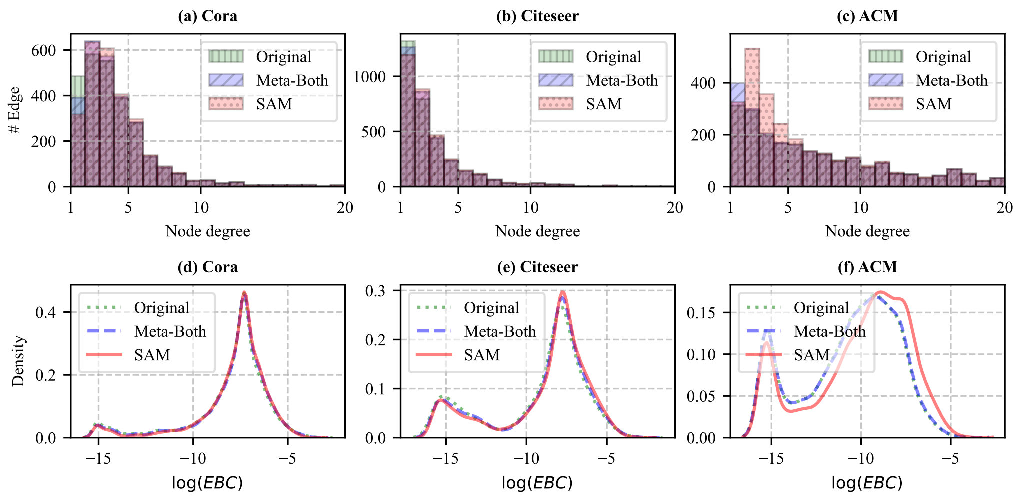
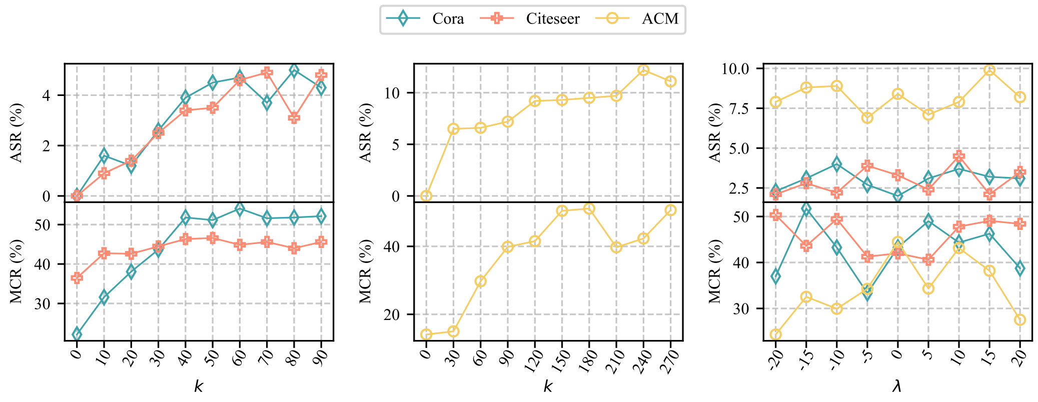


Fig. 5. Statistical distribution of the original graph and the perturbed graphs. Node degree ((a), (b), and (c)). Edge betweenness centrality ((d), (e), and (f)).

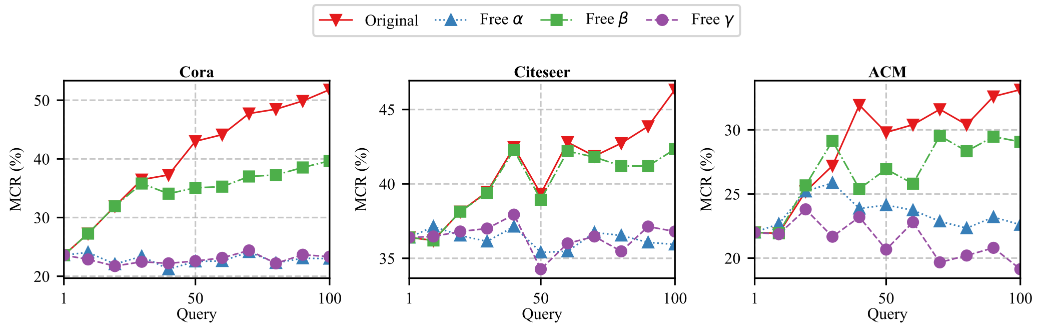
Table 3 summarizes the MSR and ACR results. Generally, we observe that SAM achieves the best performance over nearly all the criteria. Particularly, in terms of PA, SAM surpasses the state-of-the-art method Meta-Both by 1.22% (Cora) and 0.66% (Citeseer) in terms of ASR, and 2.68% (Cora) and 1.77% (Citeseer) respectively in terms of MCR. While the advantage seems to be superior on ACM that SAM achieves 5.34% ASR and 11.71% MCR improvement over Meta-Both — the performance of SAM on ACM is actually worse than on the other two datasets, which will be discussed later in Section 5.2.2.

It is also noted that the performance of PGD is far behind those of SAM and Meta-Both. This is due to the design principle of PGD, which needs sufficient queries to ensure the attack effect ($T \gtrapprox 200$ in offline tests). In terms of EA, while SAM and Meta-Both are not specifically designed for this task, they still outperform Random and DICE; especially, the ASR of SAM reaches up to 3.10% and 4.18%. This can be credited to the advantage of both approaches that formulate bilevel optimization problems to develop the perturbations. Bi-level optimization-based methods consider the complex interplay between perturbations and the model's training process. By simultaneously optimizing two interdependent objectives, these methods can generate more effective perturbations. In contrast, traditional methods simply add perturbations directly to the training and testing data, which may not be effective against well-trained models.

5.2.2 Evaluation on Attack Efficiency. SAM is designed to be query-efficient. We further evaluate its attack efficiency by comparing its MCR performance per query with Meta-Both. It is worth noting that, in terms of attack efficiency, target attacks attempt to misclassify a node within as few rounds/queries as possible. However, global attacks aim to make GNN misclassify as many nodes as possible at a given number of rounds/queries' attacks. From the results shown in Figure 3, we can easily uncover the superiority of SAM. On Cora, SAM achieves 30% MCR in just a few queries, while Meta-Both takes nearly 50 queries to achieve the same goal. In particular, the attack effect in the beginning rounds of queries is significantly obvious. Similar patterns can also be found in the results on Citeseer. These results demonstrate the efficacy of our design. First, useful meta-gradients



(a) ASR and MCR performance versus different k and λ.



(b) Attack performance when relaxing α, β, and γ respectively.

Fig. 6. Hyperparameter sensitivity.

can be obtained from the devised objective function for edge flipping. Further, Top-k Flipping algorithm incorporating the strategy of saturation attack significantly improves the attack effect in a single query.

Comparing the results among the three datasets, we also notice that SAM's attack efficiency on ACM is inferior to that on Cora and Citeseer. There is no definite advantage of SAM until approximately 30 epochs. Recalling the difference between the three datasets as presented in Table 2, ACM describes a more dense graph structure where there are more edges connecting different nodes, which makes itself more robust to adversarial attacks. This is consistent with similar discovery in [40] and [15]. In addition, Table 3 indicates that when evaluated on clean graph data, the GCN classification accuracy on ACM is higher than the other two datasets. ACM only involves three different node classes, which means that it will take more effort to deflect the classifier of GNN to reach the other side of the classification boundary of a determined class.

Furthermore, we compare the results on different GNNs, involving two additional GNNs, graph attention networks (GAT) [27] and GraphSAGE (GSAGE) [9]. From the results shown in Figure 4, we can observe that SAM generally exhibits superior attack efficiency across three GNNs than Meta-Both. However, due to the varying operating principles of the GNNs, the performance patterns are not entirely consistent. In the case of GAT, SAM does not show an advantage in the first few rounds. This could be attributed to the attention mechanism, which fully considers local dependencies and enables the model to maintain robustness against local structural changes for a certain period. Regarding GSAGE, Meta-Both does not appear to be effective in improving MCR, while SAM continues to exhibit effectiveness. This can be explained by the sampling mechanism employed in GraphSAGE, which makes the training of GNN disregard the small and local structural change; this suggests that this approach can be used as a potential defensive countermeasure.

5.2.3 Evaluation on Stealthiness. While our method can lead to a significant decrease in the model performance, a natural concern may arise if it sacrifices the stealthiness. Thereupon, we investigate the stealthiness of our method from two perspectives of proactive defense.

First, stealthiness is closely associated with query numbers. Since many detection systems consider excessive querying suspicious behaviors, users who make many queries are often flagged as potential threats. Query-efficient methods such as SAM have a higher chance of achieving the attack goal before being detected by the system. Recall the results in the last plot of Figure 3, where SAM can achieve the attack goal using up to 60 fewer queries than Meta-Both; this shows that our method can be more quickly effective in attacking and thus achieve stealth.

Secondly, we conduct a comparison of graph property statistics before and after the attacks. Unlike images, real-world graphs can be very large and intricate, making it difficult for the human eye to perceive subtle differences; hence, a common inspection method is through statistical analysis. Here, we employ two important graph property statistical metrics: node degree and edge betweenness centrality (EBC). It has been investigated that the less difference the metrics, the more similar the graph structure [5]. We compare the distributions of the two metrics among the original graph structure and the graph structures perturbed by SAM and Meta-Both. From the results on Cora and Citeseer shown in Figure 5, no clear distinction can be observed. This can be explained: while such a strategy of the saturation attack is adopted, we also set constraints to prevent the changes from overflowing. Comparing the results between SAM and Meta-Both, SAM performs slightly worse than Meta-Both; however, the difference is subtle. This is due to while SAM flips more edges than Meta-Both does, the perturbed graph structure does not betray the power-law distribution of the original one. Furthermore, for graph data such as ACM whose power-law distribution of the original graph is not salient, SAM's attack may lead to a pronounced change in the statistical information. As shown in Table 4, we measure four additional statistics of the three graphs before and after SAM's attack, namely, average shortest path length (ASPL), density, modularity [19], and the number of cliques, in which the latter two can reveal the structural information of graphs. We can draw the consistent conclusion that no significant difference exists.

Furthermore, we notice that our method tends to connect the nodes that have low degrees. These low-degree nodes are highly associated with the robustness and vulnerability of graph data. While these observations reveal the deficiency and limitation of SAM — from the other side, they provide useful knowledge for developing defense strategies in future work.

5.2.4 Ablation Tests.

Impact of hyperparameters. There exist two major hyperparameters in our method: k and λ. They collaboratively control the attack effect and number of flipped edges; here, we study the sensitivity of SAM performance to them. From the results presented in Figure 6(a), we find that while the larger the k, the better the method performance, there is a bottleneck — the performance plateaus when k is increased to approximately 50. However, while larger k can develop better results, the corresponding number of flipped edges overflows (i.e., trigger the circuit-breaker);

thus, the results are invalid. For λ, the results imply that SAM is less related to it; nevertheless, fine-tuning can slightly improve the method performance.

Impact of relaxing loss functions' controlling variables. For three optimizable variables, $\{\alpha, \beta, \gamma\} \in \Theta$ controlling the contribution of constituent loss terms, we specifically construct bounds to constraint them in the optimization, as introduced in Section 4.1. Here, we attempt to evaluate the effect of the bounds by freeing each variable. As the results shown in the right column of Figure 6(b), it can be observed that freeing different variables can lead to performance loss at different levels. For β, relaxing does not degrade the attack performance at the first few rounds of queries but then does. For α and γ, more significantly, freeing them will render the attack abortive; MCR cannot increase by the number of queries. We can conclude that constraining the optimization of the three variables is of necessity to guarantee the iterative attack performance. Nonetheless, as stated in Section 5.1.2, we set the bounds of these variables from experience. Further exploration for reasoning proper values of the bounds can be a future extension.

6 COUNTERMEASURE DISCUSSION

For defending against query-based poisoning attacks, possible methods can be generally classified into two types: 1) identify the malicious query proactively [30]; 2) clean or reduce the negative training effects of the perturbed graph [11]. The first type of method falls into the realm of cybersecurity: if the attacker can bypass the detection, the defensive line is broken through. For the second type of method, one mainstream solution is leveraging adversarial training. However, one concern is that the lack of supervised information about real perturbations in a poisoned graph impedes GNNs from modeling the distribution of adversarial edges for achieving robust training. According to a recent research [26], we find that transfer learning could be a potential technique. This method is designed to inject adversarial perturbations as supervisions for training robust GNNs and transfer the learned knowledge to improve the robustness of GNNs to the poisoned graph data. While the involved clean graphs are assumed to have similar topology distributions and attributes to the actual graph data, it is not difficult to obtain such similar graphs in the real world.

7 SUMMARY AND FUTURE WORK

In this paper, we consider a global topology attack on graph data and present an enhanced method, named SAM, to conduct query-efficient and stealthy attacks. Specifically, we propose an enhanced meta-gradient-based optimization method to obtain valuable gradient information for the attack. Then, borrowing the notion of saturation attack, we devise a constrained decision-making algorithm for edge flipping. Experiments on three datasets demonstrate that our method can reduce the accuracy of GNNs within a few queries; meanwhile, the attack is in stealth that the change to the graph structure is unnoticeable. Additionally, ablation tests further justify the proposed method's efficacy of different technical components.

This paper provides potential tools for studying the vulnerability of GNNs to adversarial attacks, especially from a perspective of practicability. In the future, on the one hand, we suppose to further dig into the stealthiness of adversarial attacks on graph neural networks: how to define it more properly, how to better quantize it, and how to make it more practical. On the other hand, we plan to rethink current work in a black-box setting and develop corresponding countermeasures.

ACKNOWLEDGMENTS

This research is supported by the Australian Research Council (ARC) DP200101374.

REFERENCES

[1] Aleksandar Bojchevski and Stephan Günnemann. 2019. Adversarial attacks on node embeddings via graph poisoning. In *Proc. International Conference on Machine Learning*. 695–704.

[2] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization.* Cambridge University Press, New York, NY, USA.

[3] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. 2020. Hopskipjumpattack: A query-efficient decision-based attack. In *Proc. IEEE symposium on security and privacy (sp)*. 1277–1294.

[4] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *Proc. International conference on machine learning*. 1115–1124.

[5] Claire Donnat and Susan Holmes. 2018. Tracking network dynamics: A survey using graph distances. *The Annals of Applied Statistics* 12, 2 (2018), 971–1012.

[6] Jiawei Du, Hu Zhang, Joey Tianyi Zhou, Yi Yang, and Jiashi Feng. 2019. Query-efficient Meta Attack to Deep Neural Networks. In *Proc. International Conference on Learning Representations.*

[7] 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).

[8] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *Proc. International Conference on Learning Representations.*

[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Proc. Advances in neural information processing systems* 30 (2017).

[10] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. 2021. Stealing links from graph neural networks. In *30th USENIX Security Symposium (USENIX Security 21)*. 2669–2686.

[11] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *Proc. ACM SIGKDD international conference on knowledge discovery & data mining*. 66–74.

[12] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proc. International Conference on Learning Representations.*

[13] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2017. Adversarial Machine Learning at Scale. In *Proc. International Conference on Learning Representations.*

[14] Dong-Hyun Lee. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, Vol. 3. 896.

[15] Jintang Li, Tao Xie, Chen Liang, Fenfang Xie, Xiangnan He, and Zibin Zheng. 2021. Adversarial attack on large scale graph. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proc. IEEE conference on computer vision and pattern recognition*. 2574–2582.

[17] Jiaming Mu, Binghui Wang, Qi Li, Kun Sun, Mingwei Xu, and Zhuotao Liu. 2021. A hard label black-box adversarial attack against graph neural networks. In *Proc. ACM SIGSAC Conference on Computer and Communications Security*. 108–125.

[18] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. 2017. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proc. ACM Workshop on Artificial Intelligence and Security*. 27–38.

[19] Mark EJ Newman. 2016. Equivalence between modularity optimization and maximum likelihood methods for community detection. *Physical Review E* 94, 5 (2016), 052315.

[20] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999* (2018).

[21] Adnan Qayyum, Muhammad Usama, Junaid Qadir, and Ala Al-Fuqaha. 2020. Securing connected & autonomous vehicles: Challenges posed by adversarial machine learning and the way forward. *IEEE Communications Surveys & Tutorials* 22, 2 (2020), 998–1026.

[22] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[23] Lichao Sun, Yingtong Dou, Carl Yang, Kai Zhang, Ji Wang, Philip S. Yu, Lifang He, and Bo Li. 2022. Adversarial Attack and Defense on Graph Data: A Survey. *IEEE Transactions on Knowledge and Data Engineering* (2022), 1–20. https://doi.org/10.1109/TKDE.2022.3201243

[24] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014.*

[25] Tsubasa Takahashi. 2019. Indirect adversarial attacks via poisoning neighbors for graph convolutional networks. In *Proc. IEEE International Conference on Big Data (Big Data)*. 1395–1400.

[26] Xianfeng Tang, Yandong Li, Yiwei Sun, Huaxiu Yao, Prasenjit Mitra, and Suhang Wang. 2020. Transferring robustness for graph neural network against poisoning attacks. In *Proc. International Conference on Web Search and Data Mining*. 600–608.

[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proc. International Conference on Learning Representations.*

[28] Binghui Wang and Neil Zhenqiang Gong. 2019. Attacking graph-based classification via manipulating the graph structure. In *Proc. ACM SIGSAC Conference on Computer and Communications Security*. 2023–2040.

[29] Binghui Wang, Tianxiang Zhou, Minhua Lin, Pan Zhou, Ang Li, Meng Pang, Cai Fu, Hai Li, and Yiran Chen. 2020. Evasion attacks to graph neural networks via influence function. *arXiv preprint arXiv:2009.00203* (2020).

[30] Haopei Wang, Lei Xu, and Guofei Gu. 2015. Floodguard: A dos attack prevention extension in software-defined networks. In *Proc. Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. 239–250.

[31] Shen Wang and S Yu Philip. 2019. Heterogeneous Graph Matching Networks: Application to Unknown Malware Detection. In *Proc. IEEE International Conference on Big Data (Big Data)*. 5401–5408.

[32] Marcin Waniek, Tomasz P Michalak, Michael J Wooldridge, and Talal Rahwan. 2018. Hiding individuals and communities in a social network. *Nature Human Behaviour* 2, 2 (2018), 139–147.

[33] Bingzhe Wu, Yatao Bian, Hengtong Zhang, Jintang Li, Junchi Yu, Liang Chen, Chaochao Chen, and Junzhou Huang. 2022. Trustworthy Graph Learning: Reliability, Explainability, and Privacy Protection. In *Proc. ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4838–4839.

[34] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[35] Feng Xia, Ke Sun, Shuo Yu, Abdul Aziz, Liangtian Wan, Shirui Pan, and Huan Liu. 2021. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence* 2, 2 (2021), 109–127.

[36] 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. In *Proc. International Joint Conference on Artificial Intelligence*. 3961–3967.

[37] Zheng Yuan, Jie Zhang, Yunpei Jia, Chuanqi Tan, Tao Xue, and Shiguang Shan. 2021. Meta gradient adversarial attack. In *Proc. IEEE/CVF International Conference on Computer Vision*. 7748–7757.

[38] Xiao Zang, Yi Xie, Jie Chen, and Bo Yuan. 2021. Graph Universal Adversarial Attacks: A Few Bad Actors Ruin Graph Learning Models. In *Proc. International Joint Conference on Artificial Intelligence*. 3328–3334.

[39] He Zhang, Bang Wu, Xingliang Yuan, Shirui Pan, Hanghang Tong, and Jian Pei. 2022. Trustworthy Graph Neural Networks: Aspects, Methods and Trends. *arXiv preprint arXiv:2205.07424* (2022).

[40] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. 2021. Backdoor attacks to graph neural networks. In *Proc. ACM Symposium on Access Control Models and Technologies*. 15–26.

[41] Zaixi Zhang, Qi Liu, Zhenya Huang, Hao Wang, Chengqiang Lu, Chuanren Liu, and Enhong Chen. 2021. GraphMI: Extracting Private Graph Data from Graph Neural Networks. In *Proc. International Joint Conference on Artificial Intelligence*. 3749–3755.

[42] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2847–2856.

[43] Daniel Zügner and Stephan Günnemann. 2018. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *Proc. International Conference on Learning Representations.*