A Black-Box Adversarial Attack Method via Nesterov Accelerated Gradient and Rewiring Towards Attacking Graph Neural Networks

Shu Zhao, Wenyu Wang, Ziwei Du, Jie Chen, and Zhen Duan

Abstract—**Recent studies have shown that Graph Neural Networks (GNNs) are vulnerable to well-designed and imperceptible adversarial attack. Attacks utilizing gradient information are widely used in the field of attack due to their simplicity and efficiency. However, several challenges are faced by gradient-based attacks: 1) Generate perturbations use white-box attacks (i.e., requiring access to the full knowledge of the model), which is not practical in the real world; 2) It is easy to drop into local optima; and 3) The perturbation budget is not limited and might be detected even if the number of modified edges is small. Faced with the above challenges, this article proposes a black-box adversarial attack method, named NAG-R, which consists of two modules known as Nesterov Accelerated Gradient attack module and Rewiring optimization module. Specifically, inspired by adversarial attacks on images, the first module generates perturbations by introducing Nesterov Accelerated Gradient (NAG) to avoid falling into local optima. The second module keeps the fundamental properties of the graph (e.g., the total degree of the graph) unchanged through a rewiring operation, thus ensuring that perturbations are imperceptible. Intensive experiments show that our method has significant attack success and transferability over existing state-of-the-art gradient-based attack methods.**

Index Terms—**Adversarial attack, black-box attack, gradient-based attack, graph neural networks, node classification.**

I. Introduction

GRAPH Neural Networks (GNNs) [1], [2] have attracted widespread attention by applying deep learning-based methods to graph data. Typically, GNNs learn the structure and feature information of graph data by aggregating the neighborhood information of nodes in a graph, and have been rapidly developed for node classification [3], [4], link prediction [5], and graph classification [6], [7]. Although graph neural networks have excellent performance, recent studies have shown that attackers can mislead GNNs to give wrong predictions by slightly perturbing their inputs (e.g., manipulating certain edges) [8], [9], [10], [11], [12]. Such erroneous predictions will have serious economic and social consequences. For example, attackers can establish connections with multiple high-credit customers to evade the fraud detection patterns of the financial system [13], or in drug discovery applications, trigger incalculable consequences by perturbing specific compounds for misclassification [14].

Graph adversarial attack has attracted more attention recently. So far, most of the existing attacks, whether attacking the adjacency matrix or the feature matrix, are mainly gradient-based, since the method of flipping the edges between nodes by gradient information [8], [15], [16] [17] is simple and effective. In addition to gradient-based algorithms, several heuristics have been proposed to achieve the goal of the attack, such as those based on genetic algorithms [18] and reinforcement learning [19], where an attacker can corrupt the model without any gradient information. Yu et al. [20] propose a network embedding attack strategy based on feature decomposition and genetic algorithm, respectively. For the node classification and graph classification tasks, an effective attack method based on reinforcement learning and genetic algorithms is proposed in [9]. Wang et al. [21] propose a strategy for learning rewiring attacks on graph classification tasks using reinforcement learning. However, the process of generating adversarial networks by these methods is complex and expensive compared to gradient-based attacks.

Despite the great success of gradient-based attacks, there are still some challenges for the attackers: i) Due to the discrete structure of the graph, the adversarial networks generated by existing gradient attack methods can easily fall into poor local maxima, resulting in poor transferability of these attacks; ii) Most gradient attack efforts are white-box attacks, i.e., they are performed with knowledge of the target model, which is not practical in the real world; and iii) Current researchers have proposed defense strategies to deal with attacks [22], [23], [24]. For example, GCN-Jaccard [25] uses Jaccard coefficients to effectively measure the similarity between nodes and eliminating edges with low similarity to effectively defend against targeted adversarial attacks. From an attack budget perspective, adding/removing links with too much variance affects

Manuscript received 29 November 2022; revised 10 July 2023; accepted 13 July 2023. Date of publication 19 July 2023; date of current version 13 November 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 61876001, in part by the Natural Science Foundation for the Higher Education Institutions of Anhui Province of China under Grant KJ2021A0039, in part by Provincial Graduate Academic Innovation Program, and in part by the Beijing Academy of Artificial Intelligence (BAAI). Recommended for acceptance by Z. Liu. *(Corresponding author: Shu Zhao.)*

The authors are with the Key Laboratory of Intelligent Computing and Signal Processing, Ministry of Education, Anhui University, Hefei, Anhui 230601, China, also with the School of Computer Science and Technology, Anhui University, Hefei, Anhui 230601, China, and also with the Information Materials and Intelligent Sensing Laboratory of Anhui Province, Anhui University, Hefei, Anhui 230601, China (e-mail: zhaoshuzs@ahu.edu.cn; e21201097@ahu.edu.cn; duziwei1225@163.com; chenjie200398@163.com; 2844926@qq.com).

Digital Object Identifier 10.1109/TBDATA.2023.3296936

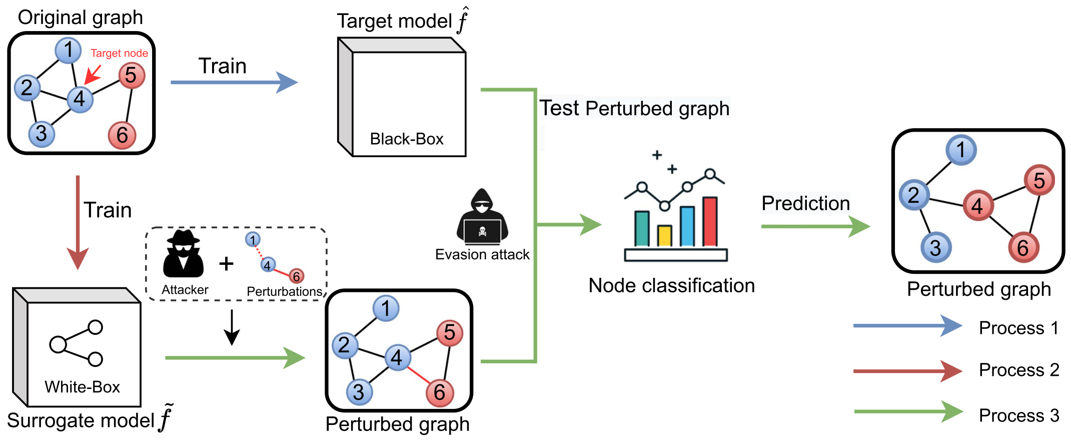

Fig. 1. Example of a black-box attack. The attacker makes the target node misclassified by flipping its edges. Process 1 and process 2 train the local surrogate model $\hat{f}$ and the target model $\tilde{f}$ with the original graph, respectively, and in process 3 the attacker generates a perturbed graph through the surrogate model $\hat{f}$ and attacks the target model in the test phase to misclassify the target nodes.

the basic properties of the graph (e.g., the total degree of the graph) and is easily detected by the defense model.

In this work, we focus on gradient-based black-box attacks on graph neural networks. In keeping with the mainstream black-box attack methods [26], [21], we generate a perturbation graph on the surrogate model and then attack the target model. Fig. 1 illustrates the basic process of a black box attack. To address the above challenges, we propose a black-box optimization framework, NAG-R. NAG-R consists of two modules, nesterov accelerated gradient attack module and rewiring optimization module. Specifically, to tackle the first two challenges, we can view the process of generating adversarial networks as an optimization process. Inspired by adversarial attacks on images [27], nesterov accelerated gradient [28] is more efficient than ordinary gradient optimization, we introduce the idea of NAG to gradient-based adversarial attacks on graphs. Compared with the existing gradient attack methods [15], [29], [30], introducing the anticipatory update of NAG to correct the previously accumulated gradients can stabilize the update direction and help us to jump out of the local optimal solution. For the third challenge, the second module of the attack framework operates rewiring optimization to reduce the attack budget. We use cosine distance to describe the similarity between nodes after gradient attack, and reconfigure the number of edges added/removed according to the importance of edge scores to keep the node degree stable, so that the perturbation is naturally insignificant. Specifically, the Nesterov accelerated gradient method utilized in the optimization process helps to avoid local minima and accelerates the iteration speed. On the other hand, the rewiring operation aids in preserving the graph structure and maintaining the continuity of the input graph. By combining these two modules, the NAG-R method achieves a higher success rate and better scalability compared to traditional gradient-based methods.

We summarize the main contributions as follows:

- From the perspective of optimizing the attack budget, we stabilize the change in node degree by reconstructing the number of edges added/removed after an nesterov accelerated gradient attack, thus keeping the perturbations undetected.
- We adapt nesterov accelerated gradient into the nesterov accelerated gradient attack on graphs, which overcomes the defect that the conventional gradient attack is easy to fall into local optimum.
- We attack several robust graph neural networks (GCN-Jaccard, SimP-GCN) and conducted comparative experiments on several datasets. Experimental results show that our attack method achieves significant improvement in attack success rates compared with other state-of-the-art methods.

II. Related Work

Several recent studies have focused on the vulnerability of graph data to adversarial attacks. Nettack [8], RL-S2V [9] first studied adversarial attacks against graph neural networks. The results show that slight perturbations to the graph structure and node features can affect the node classification results of GNNs. Based on the attacker's knowledge of the target model, the current adversarial attack methods mainly include white-box attack, gray-box attack, and black-box attack. According to the different attack stages, attacks can also be divided into poisoning attacks and evasion attacks. Here, we review these attacks.

A. Attacker's Knowledge

1) White-Box: In white-box attacks [15], [31], [25], [32], [33], the attacker has all the information of the target model, including model structure, parameters, and gradient information, that is, the target model is completely exposed to the attacker. Typically, white-box attacks use gradient information to generate an adversarial network. FGA [15] obtains the gradient information of the links through training losses and then greedily selects the node pair with the largest absolute gradient to iteratively modify the graph. Sun et al. [34] mainly attack the link prediction task with deep graph convolutional embedding model. Q-Attack [35] is a method based on genetic algorithm to attack community detection model. Wu et al. [25] used integrated gradients to search for adversarial edges and feature perturbations. Xu et al. [31] propose a Projected Gradient Descent (PGD) structural attack, a gradient attack from the perspective of first-order optimization. TGA [36] is the first attack algorithm for dynamic graph link prediction, which exploits the gradient information of deep dynamic graph embedding at different moments to reconnect edges, making target links unpredictable. MGA [30] introduces the idea of momentum into adversarial attacks, effectively jumping out of local optima, but with poor transferability. However, the white-box attack strategy is impracticable in the real world since it is expensive to possess the full knowledge of the model.

2) Gray-Box: Gray-box attacks require only partial information (e.g., only familiar and target model structure without access to model parameters), which better reflects the real-world situation [8], [16], [17], [37]. Nettack [8] generates the perturbations which mainly change the small singular values of the graph adjacency matrix. Wang et al. [38] propose AFGSM to derive an approximate closed-form solution with a lower time cost for the attack model and sequentially inject fake nodes into the graph. Metattack [16] exploits meta-gradient to attack, however, their attack setting relies on label information. In [39], a novel Node Injection Poisoning Attack (NIPA) misleads the model by exploiting fake nodes to generate wrong predictions. Liu et al. [17] analyze the errors in structural gradients caused by model instability and the discreteness of graph structure in untargeted gray-box attacks. Liu et al. [37] propose Surrogate Representation Learning with Isometric Mapping (SRLIM), enabling the model to learn topology. It combines geodesic distances to estimate the similarity between nodes on the graph.

3) Black-Box: In the black-box attack setting [9], [21], [40], [41], [42], the attacker does not have access to the parameters or training labels of the model, which is a challenging problem from the attacker's perspective. RL-S2V [9] is the first work that uses reinforcement learning techniques to generate adversarial attacks on graphs, however, they did not evaluate transferability. Bojchevski et al. [40] show a way to attack the family of node embedding models in the black-box setting. GF-Attack [26] constructs an attacker based on graph filters and attribute matrices. ReWatt [21] proposes a novel framework for learning rewiring attacks on graph classification tasks using reinforcement learning. The work [43] proposes a more restricted black-box attack strategy where the attacker is allowed to access only a subset of nodes and can only attack a small fraction of them. Fan et al. [44] leverage a strategy of reinforcement learning for dynamic graph link prediction, which is an escape attack in a black-box setting. Xu et al. [41] propose a new attack strategy, strict black-box graph attack (STACK), which has no knowledge of the victim model and no access to queries or labels at all.

In summary, black-box attacks allow the most limited knowledge compared to white-box and gray-box attacks. In this case, the attacker can only perform black-box queries on a limited number of samples at most. However, once successful, it will be the most dangerous attack [45]. From the above review, it can be observed that most of the existing black-box attacks [9], [21], [39], [44] are generated through reinforcement learning strategies or genetic algorithms, which are computationally complex and expensive. Compared with reinforcement learning, using gradient information to generate adversarial networks is simple and efficient. In this paper, we propose a gradient-based black-box attack framework strictly according to the setup of black-box attacks, and experimentally verify that the attack has strong transferability.

B. Attack Stage

According to the different stages of the attack, the attack strategy can be divided into poisoning attack [39], [40], [46], [47], [48], [49], [50] and evasion attack [21], [32], [51], [52]. Poisoning attacks aim to perturb the training dataset, thereby affecting the performance of the target model in the training phase, while evasion attacks add perturbations in the testing phase. Most of the existing attack algorithms use poisoning attacks, every time the attacker modifies the data, the model will be retrained. In contrast, evasion attacks assume that the parameters of the trained model remain unchanged, only modifying the test data without retraining the model. However, poisoning attacks are often ineffective because models are retrained to mitigate the adverse effects of such attacks. By contrast, evasion attacks are simple and effective, making them an attractive option. Therefore, the research focus of this paper is on the method to realize the evasion attack.

III. Preliminaries

In this section, we will first give the common notation for graph data, then introduce the surrogate model GCN and gradient-based attack. See Table I for frequently used notations.

A. Notations

We consider the task of semi-supervised node classification in a single, undirected, attributed graph. Formally, we denote an attributed graph with n nodes as $\boldsymbol{G} = (\boldsymbol{A}, \boldsymbol{X})$, where $\boldsymbol{A} \in \{0,1\}^{n\times n}$ is the adjacency matrix and $\boldsymbol{X} \in \{0,1\}^{n\times d}$ is the node features matrix with d-dimension.

We use V to denote the set of nodes and $E \subseteq V \times V$ is the set of edges. In semi-supervised node classification task, given the set of labeled nodes $V_M \subseteq V$ and the unlabeled node set $V_U \subseteq V$, the goal is to learn a function $f : V \mapsto C$ which maps each node $v \in V$ to one class in C. The function f is parameterized by θ and we denote the function as f_θ. In general, the function f_θ is learned by minimizing the loss function $\mathcal{L}_{tra}$ on the labeled training nodes

$$\theta_L = \arg\min_\theta \sum_{v_i \in V_M} \mathcal{L}_{tra}(f_\theta(\boldsymbol{G})). \tag{1}$$

B. Graph Convolution Network

In this work, we generate adversarial networks based on the gradient information of graph convolutional networks (GCN), which is a well-established method for semi-supervised node classification. Therefore, in the black-box attack, we choose GCN as the local surrogate model, referring to this work [53],

TABLE I
SUMMARY OF NOTATIONS

Notations	Descriptions
$\boldsymbol{G}$	Original graph
$\boldsymbol{G}'$	Perturbed graph
V	Set of nodes $V = \{v_1, v_2, \ldots, v_n\}$
E	Set of edges $E = \{e_{1,1}, e_{1,2}, \ldots, e_{n,j}\}$
C	Set of edges class labels of nodes
n, d	Number of nodes, number of features
$\boldsymbol{A}$	Adjacency matrix, $\boldsymbol{A} \in \{0,1\}^{n\times n}$
$\boldsymbol{X}$	Feature matrix, $\boldsymbol{X} \in \{0,1\}^{n\times d}$
$\boldsymbol{A}'$	Modified adjacency matrix
$\boldsymbol{X}'$	Modified feature matrix
$\hat{\boldsymbol{A}}$	Normalized adjacency matrix
$\boldsymbol{Z}$	Output of the model $\boldsymbol{Z} = f_\theta(\boldsymbol{X}, \boldsymbol{A})$
$\boldsymbol{B}$	Gradient matrix
$\hat{\boldsymbol{B}}$	Accumulated gradient matrix
E_1, E_2	Set of deleted edges, set of added edges
V_M, V_U	Set of labeled nodes ,set of unlabeled node
$\mathcal{S}(\cdot)$	Number of elements in the set
Δ	Attack budgets
$K_{i,j}$	Similarity score for node pair (v_i, v_j)
f_θ	Graph neural networks model
$\hat{f}_\theta, \tilde{f}_\theta$	Surrogate model, Target model for defense
$\mathcal{L}$	Loss function
$\|\cdot\|_k$	l_k norm

for GCN, initially, $H^0 = X$. Then the GCN model follows the following rules to aggregate the adjacent features

$$\boldsymbol{H}^{(l+1)} = \sigma(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}), \tag{2}$$

where $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_n$ is the adjacency matrix of the graph $\boldsymbol{G}$ with self connections added, $\boldsymbol{I}_n$ is a n by n identity matrix. $\tilde{\boldsymbol{D}}$ is a diagonal matrix with $\tilde{\boldsymbol{D}}_{ii} = \sum_j \tilde{\boldsymbol{A}}_{ij}$, and $\sigma(\cdot)$ is a non-linear activation function. A two-layer GCN is commonly used for semi-supervised node classification tasks [53]. Therefore the whole network is usually described as

$$\boldsymbol{Z} = f_\theta(\boldsymbol{X}, \boldsymbol{A}) = \text{softmax}(\hat{\boldsymbol{A}}\sigma(\hat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}^{(0)})\boldsymbol{W}^{(1)}), \tag{3}$$

where $\hat{\boldsymbol{A}} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}$, $\hat{\boldsymbol{A}}$ is essentially a symmetric normalized adjacency matrix. For node classification, the GCN network is usually trained with a cross entropy loss. The training loss $\mathcal{L}_{tra}$ is defined as

$$\mathcal{L}_{tra}(\theta; \boldsymbol{X}, \boldsymbol{A}) = -\sum_{i\in V_M} \ln \boldsymbol{Z}_{i,c_i}, \ \boldsymbol{Z} = f_\theta(\boldsymbol{X}, \boldsymbol{A}). \tag{4}$$

C. Gradient-Based Attack

Overall goal: The goal of gradient-based attacks is to minimize the number of correct predictions of f_θ on a set of target nodes V_t by modifying the original graph $\boldsymbol{G}$

$$\min_{\boldsymbol{G}'}\{f_\theta(\boldsymbol{G}')_i = y_i\}, i \in V_M$$
$$s.t. \ \| \boldsymbol{A}' - \boldsymbol{A} \|_0 \leq \ 2\Delta, \tag{5}$$

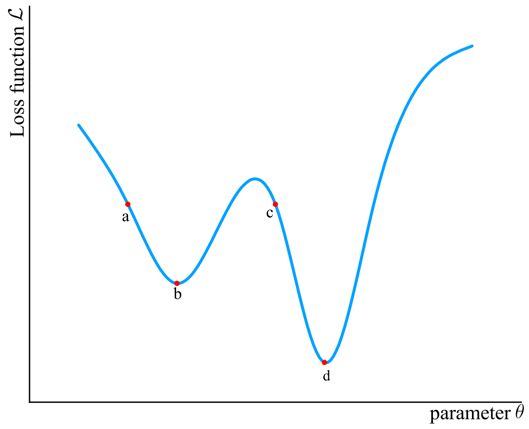

Fig. 2. Gradient descent method description. a, b, c, and d represent different positions in the gradient descent process, respectively, and the value of the corresponding point represents the value of the loss function $\mathcal{L}$.

where $\boldsymbol{G}'$ is the perturbed graph, y_i is the true label of node i and Δ is the attack budget. Since the graph $\boldsymbol{G}'$ is undirected, Δ is multiplied by 2 in the constraints.

Attack process: We will describe the general procedure of the gradient-based attack method in this section. We use the training loss function $\mathcal{L}_{tra}$ to calculate the gradient information of each edge, and then modify the elements one by one according to the principle of the greedy algorithm. Specifically, we select the edge with the largest absolute gradient value for modification, because the edge with higher gradient value has a greater influence on the decision of the classifier. Since only attacks on the adjacency matrix $\boldsymbol{A}$ are considered, we need to first calculate the gradient of the loss function $\mathcal{L}_{tra}$ with respect to $\boldsymbol{A}$, we introduce a gradient matrix $\boldsymbol{B}^{(k)} \in \mathbb{R}^{|V|\times|V|}$ which is defined as

$$\boldsymbol{B}^{(k)} = \frac{\partial \mathcal{L}_{tra}(\theta; \boldsymbol{X}, \boldsymbol{A}'^{(k)})}{\partial \boldsymbol{A}'^{(k)}} = \nabla_{\boldsymbol{A}}\mathcal{L}_{\text{tra}}(f_\theta(\theta; \boldsymbol{X}, \boldsymbol{A}'^{(k)})), \tag{6}$$

where $\boldsymbol{B}^{(k)}$ represents the gradient matrix obtained at the k-th iteration, and $\boldsymbol{A}'^{(k)}$ is the adjacency matrix after the $(k-1)$th perturbation. Specially, when $k = 0$, $\boldsymbol{A}'^{(0)} = \boldsymbol{A}$.

In each iteration, the edge $e_{u,v}$ with the largest absolute gradient and satisfying the following constraints is selected for flipping

$$\begin{cases} \text{Connect } (u,v), \text{ if } \boldsymbol{B}^{(k)}_{i,j} > 0 \text{ and } \boldsymbol{A}'^{(k)}_{i,j} = 0, \\ \text{Remove } (u,v), \text{ if } \boldsymbol{B}^{(k)}_{i,j} < 0 \text{ and } \boldsymbol{A}'^{(k)}_{i,j} = 1. \end{cases} \tag{7}$$

To conclude, the goal of gradient attack is to generate the final adversarial network $\boldsymbol{G}'$ through k iterations.

D. Nesterov Accelerated Gradient

In deep learning, the loss function is optimized to converge to a certain value so that better results can be obtained [54]. Gradient descent is a common optimization method whose main purpose is to gradually converge the objective function to a minimum value through iteration. As can be seen from Fig. 2, the gradient descent method sometimes obtains a local optimal solution [55] (if the initial value is set at point a, the direction of negative gradient at the current position is used as the search direction, and finally a local optimal solution is obtained at point b), and if the loss function is a convex function, the solution obtained by the gradient descent method is the global optimal solution. Nesterov Accelerated Gradient (NAG) [28] is an improvement of momentum optimization [56]. Momentum optimization mainly utilizes the accumulated historical momentum instead of the current gradient to accelerate convergence. The difference between NAG and Momentum optimization is that when calculating the current gradient, it moves forward with the historically accumulated momentum once, providing corrections for the previously accumulated gradients, helping the algorithm iterative process to get rid of local extremes and reach the required global minimum value.

The NAG is viewed as an improved momentum method and can be expressed as

$$g_t = \mu\, g_{t-1} + \eta \bigtriangledown_\theta \mathcal{L}(\theta - g_{t-1}), \tag{8}$$

$$\theta = \theta - g_t, \tag{9}$$

where g_t denotes the accumulated gradients at iteration t, η is the learning rate, μ denotes the decay factor, where the larger a is, the greater the influence of the previous gradient on the present update direction. $\bigtriangledown_\theta \mathcal{L}$ denotes the partial derivative of $\mathcal{L}$ concerning θ, and θ is the model parameters.

IV. The Proposed NAG-R Method

Algorithm 1: The Framework of NAG-R Against the Target Nodes.

```
Input: Original graph $\boldsymbol{G} = (\boldsymbol{A}, \boldsymbol{X})$, set of target nodes
       $V_t$, attack budget $\Delta$, surrogate model $\hat{f}_\theta$
Output: Attacked graph $\boldsymbol{G}' = (\boldsymbol{A}', \boldsymbol{X})$
1  for $v \in V_t$ do
2      /* Initialization */
3      $\hat{\boldsymbol{B}}^{(0)} \leftarrow \boldsymbol{0}^{N\times N}; \hat{\boldsymbol{A}}'^{(0)} \leftarrow norm(\boldsymbol{A}')$ ;
4      /* Nesterov accelerated gradient attack */
5      for $k = 0$ to $k = \Delta - 1$ do
6          Get $\hat{A}'^{(k)}_{nes}$ by Eq.(10) ;
7          Calculate $\boldsymbol{B}^{(k)}$ using $\nabla_A \mathcal{L}_{\text{tra}}(f_\theta(\theta; \boldsymbol{X}, \hat{A}^{(k)}_{nes}))$ ;
8          Update $\hat{\boldsymbol{B}}^{(k+1)}$ by Eq.(11) ;
9          Update $\boldsymbol{A}'^{(k+1)}$ by Eq.(7), Eq.(12) and Eq.(13) ;
10         $\hat{\boldsymbol{A}}'^{(k+1)} \leftarrow norm(\boldsymbol{A}'^{(k+1)})$
11     end
12     /* Rewiring optimization */
13     while $|deg(v_t') - deg(v_t)| > 1$ do
14         Restore the modified edges in $E_1/E_2$ by
             Eq.(15) and Eq.(16);
15         Modify the edges of the original graph by
             Eq.(15) and Eq.(17)
16     end
17     $\boldsymbol{G}' = (\boldsymbol{A}', \boldsymbol{X})$
18 end
```

In this section, we describe our proposed attack framework NAG-R, which can generate more effective attacks with smaller perturbations. It consists of two modules: nesterov accelerated gradient attack module and rewiring optimization module. The details of each module are described below. Fig. 3 shows an overview of the entire attack process of NAG-R.

A. Nesterov Accelerated Gradient Attack

In this module, NAG is introduced as a gradient-based attack, leveraging the look-ahead of NAG to escape from poor local maxima more easily, thus improving transferability. In the gradient-based attack, k iterations are needed to generate the perturbed graph $\boldsymbol{G}'_k$. In each iteration, a perturbed edge $e_{u,v}$ is generated by computing the gradient matrix $\boldsymbol{B}$ in (6). The update process of the k-th iteration can be formalized as follows:

$$\hat{\boldsymbol{A}}^{(k)}_{nes} = \hat{\boldsymbol{A}}^{(k)} + \alpha \cdot \mu \cdot \hat{\boldsymbol{B}}^{(k)}, \tag{10}$$

where $\hat{\boldsymbol{A}}^{(k)}$ and $\hat{\boldsymbol{B}}^{(k)}$ denote the normalized adjacency matrix and accumulated gradients at the iteration k. Specially, when the number of iterations $k = 0$, $\hat{\boldsymbol{B}}^{(0)} = \boldsymbol{0}^{n\times n}$. α is the step size and μ is the decay factor in the range [0,1]. The accumulated gradient is derived as

$$\hat{\boldsymbol{B}}^{(k+1)} = \mu \cdot \hat{\boldsymbol{B}}^{(k)} + \frac{\boldsymbol{B}^{(k)}}{\|\boldsymbol{B}^k\|_1}, \tag{11}$$

where $\boldsymbol{B}^{(k)}$ is the gradient matrix of the k-th iteration calculated by $\nabla_{\hat{\boldsymbol{A}}} \mathcal{L}_{\text{tra}}(f_\theta(\theta; \boldsymbol{X}, \hat{\boldsymbol{A}}^{(k)}_{nes}))$, and $\|\cdot\|_1$ represents the l_1 norm.

We update the kth adversarial network by modifying the link $e_{i,j}$ that satisfies the (7) and has the largest absolute accumulated gradient

$$\hat{\boldsymbol{B}}^{(k+1)}_{i,j} = \max_{v_i, v_j \in V} \{|\hat{\boldsymbol{B}}^{(k+1)}|\}, \tag{12}$$

$$\boldsymbol{A}'^{(k+1)}_{i,j} = \boldsymbol{A}'^{(k)}_{i,j} + \mathbb{I}(\hat{\boldsymbol{B}}^{(k+1)}_{i,j}), \tag{13}$$

where $\mathbb{I}(\hat{\boldsymbol{B}}^{(k+1)}_{i,j})$ denotes the accumulated gradient sign of the selected node pair (v_i, v_j). $\mathbb{I}(\boldsymbol{x})$ is defined as follows:

$$\mathbb{I}(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{x} > 0, \\ -1, & \text{otherwise}, \end{cases} \tag{14}$$

when $k = \Delta$, where Δ is the attack budget, the final adversarial network generated by the nesterov accelerated gradient attack is output.

B. Rewiring Optimization

Consider a situation where the difference between the number of edges added and removed is too large when the amount of perturbations is relatively large, which can destroy the basic properties of the graph (e.g. number of edges and total degrees of the graph). Therefore, from the perspective of optimizing the graph structure, the main goal of the second module of the framework is to minimize the variation of the total degree of the graph in the case of the attack budget Δ.

The rewiring operation is not executed for every perturbation, the trigger condition for execution is $\mathcal{S}(E_1) \neq \mathcal{S}(E_2)$ after the gradient attack, where E_1 and E_2 refer to the set of deleted edges

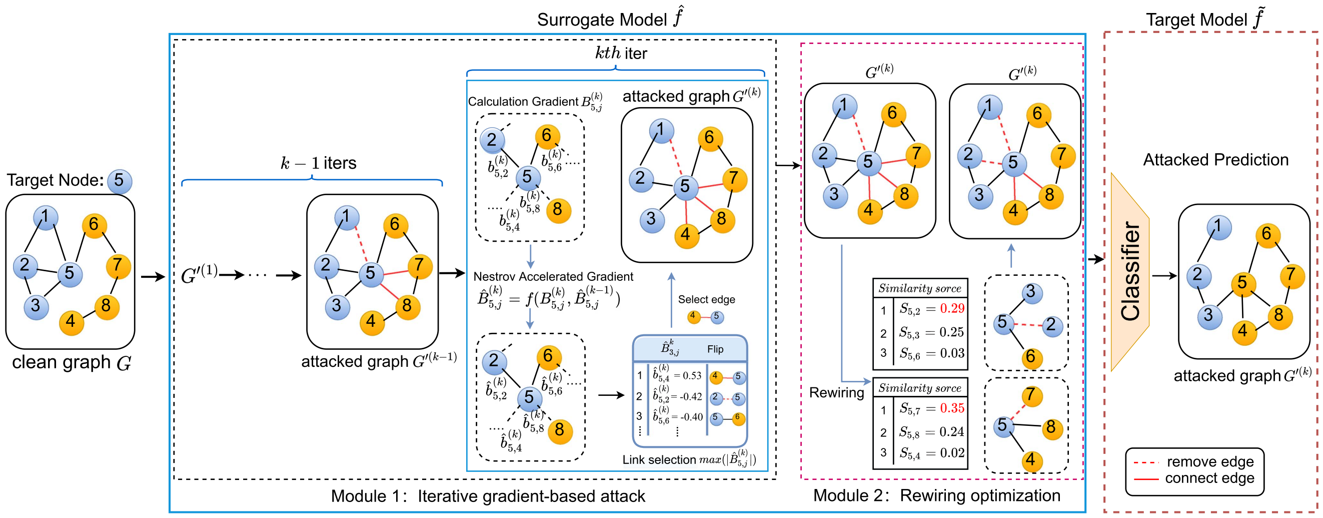
Surrogate Model $\hat{f}$
Target Model $\tilde{f}$
kth iter
$k-1$ iters
Target Node: 5
clean graph G
attacked graph $G'^{(k-1)}$
Calculation Gradient $B^{(k)}_{5,j}$
attacked graph $G'^{(k)}$
Nestrov Accelerated Gradient
$\hat{B}^{(k)}_{5,j} = f(B^{(k)}_{5,j}, \hat{B}^{(k-1)}_{5,j})$
Select edge
Flip
Link selection $max(|\hat{B}^{(k)}_{5,j}|)$
Rewiring
Similarity sorce
$S_{5,2} = 0.29$
$S_{5,3} = 0.25$
$S_{5,6} = 0.03$
$S_{5,7} = 0.35$
$S_{5,8} = 0.24$
$S_{5,4} = 0.02$
Classifier
Attacked Prediction
attacked graph $G'^{(k)}$
remove edge
connect edge
Module 1: Iterative gradient-based attack
Module 2: Rewiring optimization

Fig. 3. Overview of the entire attack procedure of the NAG-R framework. We perform a black-box evasion attack, so we need a surrogate model to perform the attack. We use a linearized two-layer graph convolutional network in (3) as a surrogate model $\hat{f}$. $\boldsymbol{G}'^{(k)}$ denotes the adversarial network generated after k iterations. $\boldsymbol{B}^{(k)}_{5,j}$ is the gradient vector of node 3 at the k-th iteration calculated by (6). $\hat{\boldsymbol{B}}^{(k)}_{5,j}$ is the accumulated gradient vector of node 5 of iteration k calculated by (11).

and the set of added edges, respectively, and $\mathcal{S}(\cdot)$ represents the number of elements in the set. Therefore, two cases trigger the rewiring operation, $\mathcal{S}(E_1) > \mathcal{S}(E_2)$ and $\mathcal{S}(E_1) < \mathcal{S}(E_2)$. We use the cosine similarity $K_{i,j}$ to describe the importance of edge $e_{i,j}$ to the target node, and a lower $K_{i,j}$ indicates less influence on the classification result of the target node.

Rewiring the links is divided into two parts, one is to restore the links modified by the gradient attack in E_1 or E_2, and the other is to modify the links in the original graph. For restoring edges in E_1 or E_2, when $\mathcal{S}(E_1) > \mathcal{S}(E_2)$, the link with the smallest $K_{i,j}$ in E_1 is selected for restoration each time; when $\mathcal{S}(E_1) < \mathcal{S}(E_2)$, the link with the largest $K_{i,j}$ in E_2 is selected for restoration. We update the adjacency matrix $\boldsymbol{A}'$ by

$$\boldsymbol{A}'^{(t)}_{i,j} = \boldsymbol{A}'^{(t)}_{i,j} - \mathbb{I}(\boldsymbol{A}'^{(t)}_{i,j}), \tag{15}$$

$$\begin{cases} E_1 = E_1 \setminus \{e_{i,j}\}, & \mathcal{S}(E_1) > \mathcal{S}(E_2), \\ E_2 = E_2 \setminus \{e_{i,j}\}, & \mathcal{S}(E_1) < \mathcal{S}(E_2), \end{cases} \tag{16}$$

where $\boldsymbol{A}'^{(t)}_{i,j}$ denotes the tth edge of the recovery.

Reconstructing the edges of the original graph is also divided into two cases. When $\mathcal{S}(E_1) > \mathcal{S}(E_2)$, we choose the link with smaller $K_{i,j}$ in the range $[\xi, 1]$ among the 2-hop neighbors of the target node to modify, where ξ is a hyperparameter, if ξ is set too small, it means that two nodes are not correlated, and the connection does not produce an attack effect. More discussion will be shown in the section on ablation experiments in Section V. Another case is $\mathcal{S}(E_1) < \mathcal{S}(E_2)$. Unlike the above operation, the link with larger $K_{i,j}$ among the 1-hop neighbors of the target node is selected for modification. The modified edge set E of the original graph is described as

$$E = \begin{cases} E \cup \{e_{i,j}\}, & \mathcal{S}(E_1) > \mathcal{S}(E_2), \\ E \setminus \{e_{i,j}\}, & \mathcal{S}(E_1) < \mathcal{S}(E_2). \end{cases} \tag{17}$$

After the rewiring operation, the total degree change of the graph is kept minimal and the generated $\boldsymbol{G}'$ is used as the final adversarial network, provided that the budget Δ is satisfied. The overall attack process of NAG-R is illustrated in Algorithm 1. We first generate the adversarial network by perturbation using NAG in lines 2-11, and then perform the rewiring operation in lines 12-16.

V. Experiments

In this section, we evaluate the effectiveness of NAG-R for node classification applications, a common task in the evaluation of many existing graph adversarial attack methods. For targeted attacks (i.e., attacks against the target node), the effectiveness of introducing NAG-NR without rewiring as well as NAG-R to attack is verified. The main attack task is consistent with Nettack [8], RL-S2V [9], by modifying the graph structure to minimize the correct prediction of the target node set V_t in (5). For global attacks (i.e. select a specified number of perturbed edges from the entire graph structure), the experimental part only verifies the attack performance of NAG-NR against global attacks since there are no target nodes and the global attack is not convenient for rewiring operations. Finally, we discuss the important tests of NAG-R by conducting ablation experiments from different perspectives.

A. Datasets

In this section, we evaluate NAG-R on four datasets, namely Cora, Citeseer, Cora-ML [8], Pubmed [57], and compare it with state-of-the-art gradient attack algorithms to illustrate its effectiveness. The statistics of the datasets are shown in Table II.

- *Cora:* This dataset is a citation network containing 2708 scientific publications which are classified into seven research classes. There are 5429 edges between all these papers which indicating their citation relations. Each paper is associated with the title and abstract content. The

TABLE II
DATASET STATISTICS

Datasets	Nodes	Edges	Features	Classes
Cora	2708	5429	1433	7
Citeseer	3312	4715	3703	6
Cora-ML	2995	8416	2879	7
Pubmed	19717	44,338	500	3

attribute of each node is the bag of words representation of the corresponding paper.

- *Citeseer:* This dataset consists of 3312 scientific publications grouped into six categories. The citation network consists of 4715 links. Each paper is associated with the title and abstract content. The dictionary consists of 3703 unique words.
- *Cora-ML:* The dataset is a network extracted from the original citation network by Graph2Gauss [58].
- *Pubmed:* This dataset consists of 19717 diabetes-related articles in the Pubmed database, divided into three categories. The citation network consists of 44338 links, where the edges indicate citation relationships. The node attributes are TF-IDF weighted word frequencies, and node labels are the types of diabetes addressed in the articles.

B. Baselines Methods

Here, we evaluate our method on different datasets with multiple baseline methods. For a fair comparison, all methods will use the same surrogate model GCN. The work in this paper focuses on evasive attack on target nodes, but the experimental results show that our method is also applicable to global and poisoning attacks, which will be verified separately below.

1) Targeted Attacks: In this section, we follow the setting of Nettack [8] to attack the target nodes and only consider the largest connected component of the graph for each dataset, the attack budget Δ is set as the degree of the target node, the baseline methods are Random Attack (RA) [53], GradArgmax [9], Nettack.

- *Random Attack-T (RA-T):* RA-T is the simplest method of attack on the target node, which randomly adds or removes edges between the target node and other nodes with probability p.
- *GradArgmax:* GradArgmax is a traditional gradient-based algorithm where the attack budget Δ is defined as the degree of the target node t. By calculating the gradient of all edges of the target node and other nodes in the adjacency matrix, the edge with the largest absolute gradient is selected for deletion or addition.
- *Nettack:* Nettack is the strongest baseline for graph structure attacks and can be perturbed by modifying the adjacency matrix and the feature matrix of the graph. To be fair, we only use Nettack to modify the adjacency matrix without feature attacks when performing structure attacks.

2) Global Attack: The purpose of the global attack is to reduce the classification accuracy of the model with perturbations within the budget. The baseline methods for comparison are Random Attack-G (RA-G), DICE [59], GradArgmax-G [60], EpoAtk [29].

- *Random Attack-G (RA-G):* RA-G is a global attack under which the budget Δ node pairs are randomly chosen to add or remove edges. It is the simplest global attack and the least effective one.
- *DICE:* DICE is originally a heuristic algorithm for disguising communities. In this method, the class information is obtained from a model instead of the real label. Edges are inserted between nodes of different classes and deleted between nodes of the same class.
- *GradArgmax-G:* GradArgmax-G adds or removes edges by calculating the gradient over the entire adjacency matrix and selecting the Δ node pairs with the largest absolute gradient.
- *EpoAtk:* EpoAtk overcomes the shortcomings of the GradArgmax-G approach while inheriting the advantages of efficient gradient-based attacks. EpoAtk does not just select the perturbation of the maximum gradient, but generates a perturbation candidate set, and then evaluates each element in the candidate set with an efficient evaluation function in the evaluation phase.

C. Target Models

To demonstrate the transferability of our proposed method, several defense algorithms are used in our experiments.

- *GCN [53]:* GCN is one of the most representative graph neural networks that learn hidden representations by encoding both local graph structure and node features. In this work, we use a two-layer graph convolutional network as a surrogate model.
- *GCN-Jaccard [25]:* Existing attack methods usually add edges to two nodes to perturb. By removing the edges between two dissimilar nodes, GCN-Jaccard can effectively defend against targeted adversarial attacks, and can guarantee accuracy does not drop on GCN models.
- *SimP-GCN [60]:* SimP-GCN achieves adaptively balancing graph structure information and node feature information by aggregating on original and feature maps simultaneously.

D. Performance Metrics

To evaluate the performance of adversarial attack methods on the node classification task, we use the misclassification rate and accuracy as evaluation metrics.

- *Misclassification Rate (MR) [16]:* In a semi-supervised classification task, this metric can be described as the percentage of unlabeled nodes that are misclassified by an attacker by perturbing a graph. The higher misclassification rate indicates better attack performance

$$MR = \frac{FP + FN}{TP + TN + FP + FN}, \tag{18}$$

where TP, FP, FN, TN denote true positive, false positive, false negative and true negative, respectively.

- *Accuracy (ACC) [38]:* This metric is generally used in global attacks to describe the effectiveness of an attack by comparing the classification accuracy of the model before and after the attack, with lower values indicating a more effective attack

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}, \tag{19}$$

E. Parameters and Experimental Settings

1) Running Environment: The experiments in this paper are conducted on the Inspur NF8460M4 server with 4 Intel E7-4809 CPU @ 2.1 GHz and 1 Tesla P100 16 GB GPU, 1 T of RAM and 12 T Hard Disk. The codes of our proposed method are implemented with Linux in python 3.6.

2) Parameters Configuration: In order to conduct the comparison experiments fairly, we conducted different experimental setups for the attacks against the target nodes and the global attacks, respectively.

- *Targeted Attacks [8]:* For the attack against target nodes, following the experimental setup of Nettack, we take the dataset in Table II and only consider the largest connected component of the graph for each dataset. We randomly select 20% of the nodes to form the training set (half of them as the validation set), and the rest as the test set. For the other hyperparameters, the step size α and decay factor μ in (12) we set to $\alpha = 1.0$ and $\mu = 0.6$, respectively.
- *Global Attack [29]:* For global attacks, we conduct comparative experiments using the setup described in EpoAtk [29]. Closely following the experimental setting in EpoAtk, only the Citeseer dataset was validated using the largest connected component (LCC). We randomly split each of dataset into 10% labeled nodes and 90% unlabeled nodes respectively. According to the perturbation budget setting of EpoAtk, the attack budget Δ is the percentage of the edge that can be perturbed by the attacker, which is controlled at 1% and 3% respectively.

3) Code Details: It was experimentally verified that the public datasets mentioned in our proposed method are available and the code is able to run server in Section V-E1. In our experiments, version 3.6 of python was used, version 1.9 of PyTorch and version 1.17.0 of numpy were used. We will release the code and datasets on Github. Code will be available at https://github.com/AHU-ICKE/Network-Representation-Learning.

F. Experimental Results

In this section, we conduct comprehensive experiments to verify the attack performance of our method. We analyze and compare the experimental results under different attack scenarios. The results in this section are the average of ten experimental results, and the results are true and valid.

1) Targeted Attacks: In this task, we use the same setting as in Nettack [8]. Specifically, for each dataset, we choose 100 nodes with highest margin of classification (i.e., they are clearly correctly classified) as the target nodes. Table III shows the attack performance of NAG-R compared with the baseline approach on the surrogate model. The higher misclassification rate indicates better performance. As can be observed in Table III, NAG-R achieves the best performance across nearly all the datasets when attacking the surrogate model, which demonstrates the effectiveness of NAG-R. For instance, our method achieves performance gains of up-to 7% on the Cora dataset and the Pubmed dataset, and 4% on the Cora-ML dataset.

TABLE III
MISCLASSIFICATION RATE (%) ON THE SURROGATE MODEL GCN

Datasets	**Citeseer**	**Cora**	**Cora-ML**	**Pubmed**
Random	38.00	45.00	50.00	61.00
GradArgmax	61.00	58.00	66.00	72.00
Nettack	**74.00**	66.00	73.00	74.00
NAG-R	72.00	**73.00**	**77.00**	**81.00**

Here the best results are boldfaced.

Table IV depicts the performance of the adversarial samples generated with the surrogate GCN model against various defense models for black-box evasion attacks. As described in Table IV, our proposed attack framework achieves the best misclassification rate in most cases. Experimental results show that our attack method NAG-R still achieves a high misclassification rate even if the attacker has limited knowledge of the target model.

Combining Tables III and IV, it can be seen that one exception is Citesser where the attack performance of NAG-R is relatively lower than Nettack on surrogate GCN model. It outperforms Nettack on the target model, indicating that our method is highly transferable.

2) Global Attack: Considering comprehensively, since the global attack utilizes the gradient information to select the corresponding perturbation edges to attack and has no target node (one perturbation edge corresponds to two related nodes), the calculation of similarity and rewiring increases the perturbation budget. Therefore, for the global attack, we only consider the attack NAG-NR that introduces nesterov accelerated gradient to optimize the gradient. Table V describes the experimental results of NAG-NR for global poisoning attack. The lower test accuracy indicates better performance. As can be seen from the table, our method is still effective in the global poisoning attack. Furthermore, the attack that only introduces NAG without rewiring optimization still outperforms the baseline method. The experimental results were analyzed as follows.

- Compared with the state-of-the-art global attack, our method shows a significant improvement in attack performance. With only 1% attack budget, our method achieves a 5.38% performance improvement over the best performing EpoAtk on the Cora-ML dataset, and by 3.2% and 3.1% on Cora and Citesser, respectively. When the attack budget is 3%, the average performance shown on the three benchmark datasets is 5.77% higher than EpoAtk.
- NAG-NR in Table V means that only NAG iterations are used to generate the adversarial network without rewiring. We can observe that the introduction of NAG outperforms other baseline methods even without rewiring operations. Especially on the Citeseer dataset, when the budget is only 1%, the performance is improved by 4.05% compared to EpoAtk.

TABLE IV
COMPARING THE TRANSFERABILITY OF ATTACKS ON THREE TARGET MODELS

Datasets	**Citeseer**			**Cora**			**Cora-ML**		
Models	GCN	Jaccard-GCN	SimpGCN	GCN	Jaccard-GCN	SimpGCN	GCN	Jaccard-GCN	SimpGCN
Random	43.00	20.00	17.00	38.00	26.00	37.00	54.00	32.00	41.00
GradArgmax	47.00	26.00	19.00	49.00	33.00	58.00	60.00	37.00	52.00
Nettack	**73.00**	39.00	25.00	64.00	39.00	57.00	63.00	40.00	58.00
NAG-R	71.00	**42.00**	**26.00**	**71.00**	**48.00**	**60.00**	**67.00**	**42.00**	**65.00**

Here the best results are boldfaced.

TABLE V
ACCURACY(%) FOR THE GCN MODELS WITH DIFFERENT ATTACKS

Datasets	**Cora**		**Citeseer**		**Cora-ML**	
Clean	81.74/82.23		70.23/70.85		82.59/83.31	
Attack Budgets	1%	3%	1%	3%	1%	3%
RA	81.26 ± 0.3	80.76 ± 0.2	69.02 ± 0.4	69.36 ± 0.2	82.93 ± 0.3	82.43 ± 0.4
DICE	80.90 ± 0.4	80.25 ± 0.4	69.66 ± 0.3	68.91 ± 0.4	82.60 ± 0.4	81.95 ± 0.3
GradArgmax	78.62 ± 0.3	76.45 ± 0.3	68.80 ± 0.4	67.43 ± 0.3	78.47 ± 0.5	75.76 ± 0.4
EpoAtk	77.41 ± 0.5	73.25 ± 0.4	68.50 ± 0.4	65.75 ± 0.3	77.52 ± 0.4	73.27 ± 0.4
NAG-NR	$\mathbf{75.36 \pm 0.4}$	$\mathbf{70.46 \pm 0.4}$	$\mathbf{64.00 \pm 0.2}$	$\mathbf{59.80 \pm 0.3}$	$\mathbf{73.65 \pm 0.3}$	$\mathbf{71.30 \pm 0.2}$

Here the best results are boldfaced.

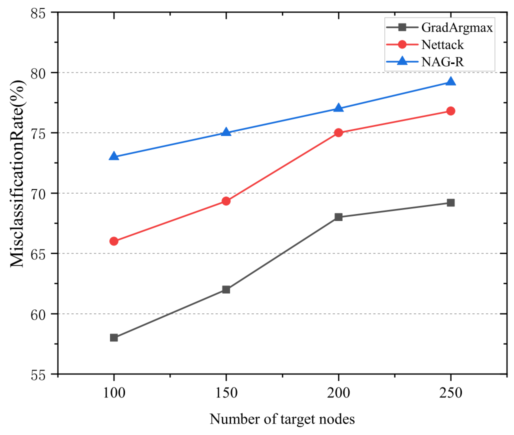

Fig. 4. Performance of NAG-R and two baseline methods attacking different numbers of target nodes on the Cora dataset.

G. Ablation Study

In this section, we analyze the performance of NAG-R in attacking target nodes under different conditions and parameters through an ablation study.

1) Number of Nodes Influence: We experimentally verify that our attack method NAG-R outperforms two baseline methods (Nettack, GradArgmax) for different numbers of target nodes in a targeted attack. As shown in Fig. 4, when the number of target nodes is 100, NAG-R has the best attack effect, with a misclassification rate 7% higher than Nettack and 13% higher than GradArgmax. This implies that NAG-R can maintain this good attack effect even if the number of target nodes is different.

2) Parameter Analysis: In (11), μ is the decay factor, which is used to describe the influence of the historical gradient, where the larger μ, the greater the influence of the historical gradient on the present. Fig. 6 provides a visualization of the performance of the NAG-R attack for different μ on the four benchmark datasets.

As mentioned before, the larger μ is, the more obvious is the effect of the accumulated gradient on the current gradient update. In Fig. 6, we can observe that the peaks of μ are different, indicating that the impact of accumulated gradient on the current gradient update is different for different datasets. Combined with the data in Table III, the accumulated gradient significantly improves the performance of the attack on the Cora, Cora-ML, and Pubmed datasets. On the Citeseer dataset, the peak of μ is around 0.3, so it has little effect on the update of the current gradient. The experimental results of NAG-R in Tables III and V were recorded at the peak of μ.

3) Nesterov Accelerated Gradient: We propose an attack strategy based on nesterov accelerated gradient in Section IV-A. Fig. 5(a) compares the performance of the strategy of generating an adversarial network by nesterov accelerated gradient iteration without rewiring operation (NAG-NR) with the traditional gradient-based strategy (GradArgmax) on four baseline datasets. From the data in Fig. 5(a), we can analyze that the performance of the attack is improved on each baseline dataset by introducing the rewiring-free NAG-NR compared to the GradArgmax. For example, NAG-R outperforms rewiring with randomly selected edges by 6% on the Cora dataset, and at least 2% higher on other datasets. Therefore, this demonstrates the effectiveness of introducing nesterov accelerated gradient into graph adversarial attacks.

4) Rewiring Optimization: The rewiring optimization proposed in Section IV-B also demonstrates effective attack results. Fig. 5(b) shows the performance comparison of rewiring with a similarity strategy (NAG-R), rewiring with randomly selected edges (Randomly), and rewiring in the Nesterov Accelerated Gradient attack module (NAG-GR). Among them, randomly selecting edges for rewiring means selecting k edges in the edge

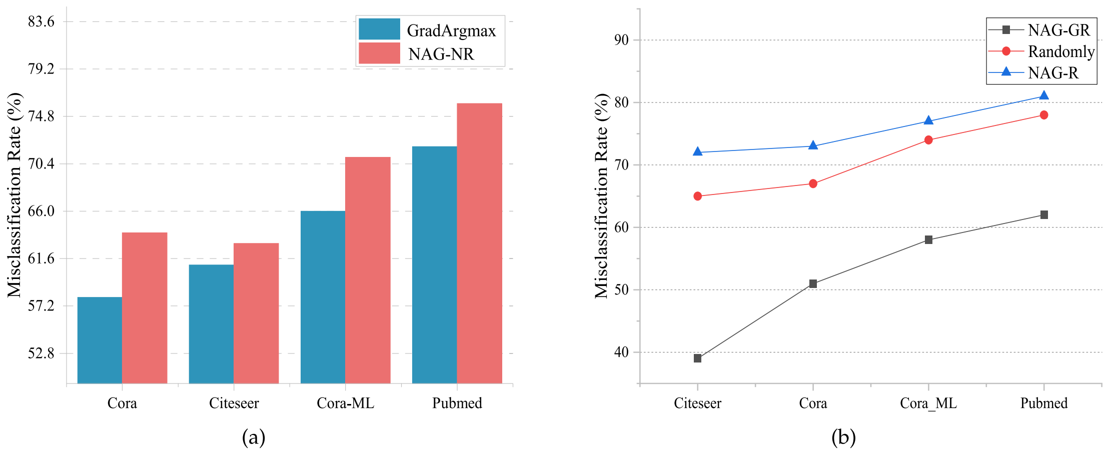

Fig. 5. (a): Performance comparison of nesterov accelerated gradient iterations are used to generate the adversarial network without rewiring(NAG-NR) and GradArgmax on four datasets. (b): Performance comparison of NAG-R with embedded rewiring attack in the gradient attack phase (NAG-GR) and rewiring attack using randomly selected edges (Randomly).

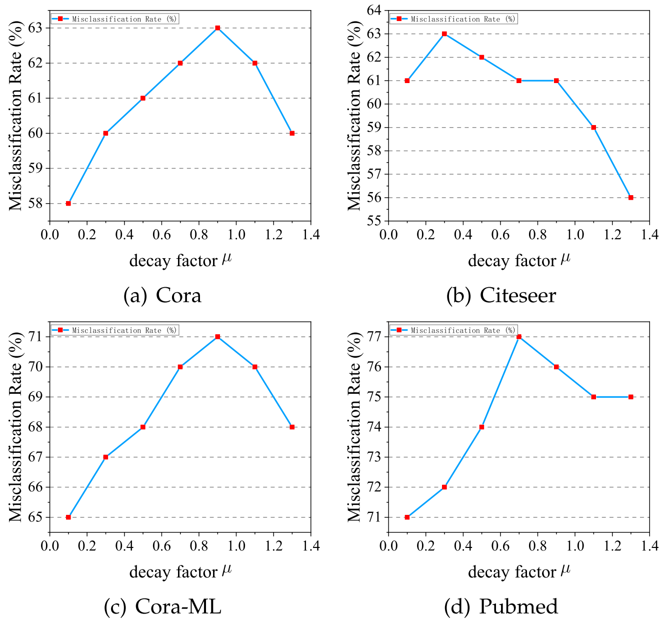

Fig. 6. (a)(b)(c)(d) show the result plots of the impact of different decay factor μ on the attack performance of the four benchmark datasets (Cora, Citeseer, Cora-ML, Pubmed), respectively.

set E with probability p_1 for rewiring under the same budget Δ. NAG-GR embeds rewiring in the Nesterov accelerated gradient attack module to perturb the target nodes while maintaining the overall perturbation budget. After modifying an edge through the gradient attack, the corresponding edge is restored using the similarity score.

The data in Fig. 5(b) demonstrates that our method significantly improves the success rate of the attack compared to NAG-GR and Randomly. For instance, NAG-R outperforms rewiring with randomly selected edges by 7% on the Citeseer dataset, and at least 3% higher on other datasets. In particular, it can be observed that embedding rewiring in the Nesterov Accelerated Gradient attack module introduces a significant amount of instability and uncertainty, which disrupts the stability and convergence of the gradient attack, leading to poor results.

In summary, through the above experimental analysis, each module of NAG-R plays an important role in improving the attack performance.

VI. Conclusion and Future Work

In this article, based on the fact that nesterov accelerated gradient outperforms gradient descent, we propose an adversarial attack framework NAG-R for graphs based on nesterov accelerated gradient. We divide the entire attack framework into two modules: first generating the adversarial network by performing an nesterov accelerated gradient attack on the surrogate model, and then optimizing the adversarial network by rewiring it, preserving some basic properties of the graph (e.g., the total degree of the graph). The experimental results of attacking multiple defense models in Section V show that our approach has a high success rate and transferability of attacks. In future, we will explore the following directions:

1) We will seek to investigate the application of gradient-based attacks to large-scale graph data.
2) The current experimental setup only considers the GCN-based model as the target model, which may limit the transferability of the proposed method to other graph neural network models such as GAT, GIN and MPNN. We will try to attack different types of graph neural network models to make the research more challenging and practical.

Acknowledgement

We acknowledge the High-performance Computing Platform of Anhui University for providing computing resources.

References

[1] M. Gori, G. Monfardini, and F. Scarselli, “A new model for learning in graph domains,” in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2005, pp. 729–734.

[2] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, “A comprehensive survey on graph neural networks,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[3] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *Adv. Neural Inf. Process. Syst.*, vol. 30, pp. 1024–1034, 2017.

[4] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *Stat*, vol. 1050, 2017, Art. no. 20.

[5] M. Zhang and Y. Chen, “Link prediction based on graph neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 5171–5181.

[6] W. Ju, J. Yang, M. Qu, W. Song, J. Shen, and M. Zhang, “KGNN: Harnessing kernel-based networks for semi-supervised graph classification,” in *Proc. 15th ACM Int. Conf. Web Search Data Mining*, 2022, pp. 421–429.

[7] S. Zhao, Z. Du, J. Chen, Y. Zhang, J. Tang, and P. Yu, “Hierarchical representation learning for attributed networks,” *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 3, pp. 2641–2656, Mar. 2023.

[8] D. Zügner, A. Akbarnejad, and S. Günnemann, “Adversarial attacks on neural networks for graph data,” in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 2847–2856.

[9] H. Dai et al., “Adversarial attack on graph structured data,” in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1115–1124.

[10] Z. Zhang, Q. Liu, Z. Huang, H. Wang, C.-K. Lee, and E. Chen, “Model inversion attacks against graph neural networks,” *IEEE Trans. Knowl. Data Eng.*, to be published, doi: 10.1109/TKDE.2022.3207915.

[11] L. Lin, E. Blaser, and H. Wang, “Graph structural attack by perturbing spectral distance,” in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 989–998.

[12] S. Tao, Q. Cao, H. Shen, J. Huang, Y. Wu, and X. Cheng, “Single node injection attack against graph neural networks,” in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, 2021, pp. 1794–1803.

[13] J. Sirignano and R. Cont, “Universal features of price formation in financial markets: Perspectives from deep learning,” *Quantitative Finance*, vol. 19, no. 9, pp. 1449–1459, 2019.

[14] M. Elbadawi, S. Gaisford, and A. W. Basit, “Advanced machine-learning techniques in drug discovery,” *Drug Discov. Today*, vol. 26, no. 3, pp. 769–777, 2021.

[15] J. Chen, Y. Wu, X. Xu, Y. Chen, H. Zheng, and Q. Xuan, “Fast gradient attack on network embedding,” 2018, *arXiv: 1809.02797*.

[16] D. Zügner and S. Günnemann, “Adversarial attacks on graph neural networks via meta learning,” in *Proc. Adv. 7th Int. Conf. Learn. Representations*, 2019. [Online]. Available: https://openreview.net/pdf?id=Bylnx209YX

[17] Z. Liu, Y. Luo, L. Wu, S. Li, Z. Liu, and S. Z. Li, “Are gradients on graph structure reliable in gray-box attacks?,” in *Proc. 31st ACM Int. Conf. Inf. Knowl. Manage.*, 2022, pp. 1360–1368.

[18] D. Whitley, “A genetic algorithm tutorial,” *Statist. Comput.*, vol. 4, no. 2, pp. 65–85, 1994.

[19] S. Thrun and M. L. Littman, “Reinforcement learning: An introduction,” *AI Mag.*, vol. 21, no. 1, pp. 103–103, 2000.

[20] S. Yu, J. Zheng, J. Chen, Q. Xuan, and Q. Zhang, “Unsupervised euclidean distance attack on network embedding,” in *Proc. IEEE 5th Int. Conf. Data Sci. Cyberspace*, 2020, pp. 71–77.

[21] Y. Ma, S., T. Derr, L. Wu, and J. Tang, “Graph adversarial attack via rewiring,” in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2021, pp. 1161–1169.

[22] J. Chen, Y. Wu, X. Lin, and Q. Xuan, “Can adversarial network attack be defended?,” 2019, *arXiv: 1903.05994*.

[23] Z. Deng, Y. Dong, and J. Zhu, “Batch virtual adversarial training for graph convolutional networks,” 2019, *arXiv: 1902.09192*.

[24] X. He, Z. He, X. Du, and T.-S. Chua, “Adversarial personalized ranking for recommendation,” in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 355–364.

[25] H. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, and L. Zhu, “Adversarial examples on graph data: Deep insights into attack and defense,” 2019, *arXiv: 1903.01610*.

[26] H. Chang et al., “A restricted black-box adversarial framework towards attacking graph embedding models,” in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 3389–3396.

[27] J. Lin, C. Song, K. He, L. Wang, and J. E. Hopcroft, “Nesterov accelerated gradient and scale invariance for adversarial attacks,” in *Proc. 8th Int. Conf. Learn. Representations*, Apr. 26–30, 2020.

[28] Y. Nesterov, “A method for unconstrained convex minimization problem with the rate of convergence o (1/k^2),” *Doklady AN USSR*, vol. 269, pp. 543–547, 1983.

[29] X. Lin et al., “Exploratory adversarial attacks on graph neural networks,” in *Proc. IEEE Int. Conf. Data Mining*, 2020, pp. 1136–1141.

[30] J. Chen et al., “MGA: Momentum gradient attack on network,” *IEEE Trans. Computat. Social Syst.*, vol. 8, no. 1, pp. 99–109, Feb. 2021.

[31] K. Xu et al., “Topology attack and defense for graph neural networks: An optimization perspective,” in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 3961–3967.

[32] H. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, and L. Zhu, “Adversarial examples for graph data: Deep insights into attack and defense,” in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Macao, China, Aug. 10–16, 2019, pp. 4816–4823.

[33] T. Takahashi, “Indirect adversarial attacks via poisoning neighbors for graph convolutional networks,” in *Proc. IEEE Int. Conf. Big Data*, 2019, pp. 1395–1400.

[34] M. Sun et al., “Data poisoning attack against unsupervised node embedding methods,” 2018, *arXiv: 1810.12881*.

[35] J. Chen et al., “GA-based Q-attack on community detection,” *IEEE Trans. Computat. Social Syst.*, vol. 6, no. 3, pp. 491–503, Jun. 2019.

[36] J. Chen, J. Zhang, Z. Chen, M. Du, and Q. Xuan, “Time-aware gradient attack on dynamic network link prediction,” *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 2, pp. 2091–2102, Feb. 2023.

[37] Z. Liu, Y. Luo, Z. Zang, and S. Z. Li, “Surrogate representation learning with isometric mapping for gray-box graph adversarial attacks,” in *Proc. 15th ACM Int. Conf. Web Search Data Mining*, 2022, pp. 591–598.

[38] J. Wang, M. Luo, F. Suya, J. Li, Z. Yang, and Q. Zheng, “Scalable attack on graph data by injecting vicious nodes,” *Data Mining Knowl. Discov.*, vol. 34, no. 5, pp. 1363–1389, 2020.

[39] Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar, “Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach,” in *Proc. Web Conf.*, 2020, pp. 673–683.

[40] A. Bojchevski and S. Günnemann, “Adversarial attacks on node embeddings via graph poisoning,” in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 695–704.

[41] J. Xu et al., “Blindfolded attackers still threatening: Strict black-box adversarial attacks on graphs,” in *Proc. 24th Conf. Innov. Appl. Artif. Intell.*, 2022, pp. 4299–4307.

[42] B. Wang, Y. Li, and P. Zhou, “Bandits for structure perturbation-based black-box attacks to graph neural networks with theoretical guarantees,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 13379–13387.

[43] J. Ma, S. Ding, and Q. Mei, “Black-box adversarial attacks on graph neural networks with limited node access,” 2020, *arXiv: 2006.05057*.

[44] H. Fan et al., “Reinforcement learning-based black-box evasion attacks to link prediction in dynamic graphs,” in *Proc. IEEE 23rd Int. Conf. High Perform. Comput. Commun.; 7th Int. Conf. Data Sci. Syst.; 19th Int. Conf. Smart City; 7th Int. Conf. Dependability Sensor, Cloud Big Data Syst. Appl.*, 2021, pp. 933–940.

[45] L. Chen et al., “A survey of adversarial learning on graphs,” 2020, *arXiv: 2003.05730*.

[46] B. Wang and N. Z. Gong, “Attacking graph-based classification via manipulating the graph structure,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 2023–2040.

[47] H. Zhang et al., “Towards data poisoning attack against knowledge graph embedding,” 2019, *arXiv: 1904.12052*.

[48] Y. Zhu et al., “BinarizedAttack: Structural poisoning attacks to graph-based anomaly detection,” in *Proc. IEEE 38th Int. Conf. Data Eng.*, 2022, pp. 14–26.

[49] C. Jiang, Y. He, R. Chapman, and H. Wu, “Camouflaged poisoning attack on graph neural networks,” in *Proc. Int. Conf. Multimedia Retrieval*, 2022, pp. 451–461.

[50] T. T. Nguyen et al., “Poisoning GNN-based recommender systems with generative surrogate-based attacks,” *ACM Trans. Inf. Syst.*, vol. 45, no. 3, pp. 1–24, 2022.

[51] H. Zhang, X. Yuan, C. Zhou, and S. Pan, “Projective ranking-based GNN evasion attacks,” 2022, *arXiv:2202.12993*.

[52] B. Wang et al., “Efficient evasion attacks to graph neural networks via influence function,” 2020, *arXiv: 2009.00203*.

[53] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *Proc. 5th Int. Conf. Learn. Representations*, 2017. [Online]. Available: https://openreview.net/pdf?id=SJU4ayYgl

[54] H. E. Robbins, “A stochastic approximation method,” *Ann. Math. Statist.*, vol. 22, pp. 400–407, 2007.

[55] Y. Chen, Q. Liu, and W. Zhang, “A powerful transferability adversarial examples generation method based on nesterov momentum optimization,” *Int. J. Mach. Learn. Comput.*, vol. 10, no. 3, pp. 431–436, 2020.

[56] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Netw.*, vol. 12, no. 1, pp. 145–151, 1999.

[57] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, pp. 93–93, 2008.

[58] A. Bojchevski and S. Günnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," 2017, *arXiv: 1707.03815*.

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

[60] W. Jin, T. Derr, Y. Wang, Y. Ma, Z. Liu, and J. Tang, "Node similarity preserving graph convolutional networks," in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, 2021, pp. 148–156.

Shu Zhao is a professor and supervisor of the PhD students with the School of Computer Science and Technology, Anhui University, China. Her research interests include network representation learning, knowledge graph, and social network analysis.

Wenyu Wang is working toward the postgraduate degree with the School of Computer Science and Technology, Anhui University. His research interests include adversarial attacks on graph.

Ziwei Du is working toward the postgraduate degree with the School of Computer Science and Technology, Anhui University. Her research interests include network representation learning, granular computing.

Jie Chen is an associate professor and supervisor of the master's students with the School of Computer Science and Technology, Anhui University, China. Her research interests include machine learning, text sentiment classification and granular computing.

Zhen Duan is a lecturer with the School of Computer Science and Technology, Anhui University. His current research interests include social network and machine learning.