Machine Learning (2025) 114:14
https://doi.org/10.1007/s10994-024-06706-9

SPA: A poisoning attack framework for graph neural networks through searching and pairing

Xiao Liu[1] · **Jun-Jie Huang**[1] · **Wentao Zhao**[1] · **Ziyue Wang**[1] · **Zihan Chen**[1] · **Yi Pan**[1]

Received: 27 February 2024 / Revised: 9 October 2024 / Accepted: 12 November 2024 /
Published online: 16 January 2025

Abstract
Graph Neural Networks (GNN) have played an important role in many fields, while GNNs also suffer from adversarial attacks that aim to malfunction the GNN model by changing the adjacency matrix (i.e. generating adversarial edges) or node features (i.e. generating adversarial features) in graph data. Although the gradient-based adversarial attack methods have achieved remarkable results in DNNs, optimizing discrete adversarial edges in graph data using continuous gradients may lead to sub-optimal solutions. In order to alleviate this situation, we propose a novel Searching and Pairing Attack (SPA) method to effectively generate adversarial edges by treating each adversarial edge as a combination of a pair of adversarial nodes. The proposed SPA method generates the adversarial edges through a Node Searching step and a Node Pairing step. The proposed Node Searching Ant Colony Optimization (NS-ACO) improves the attack effect by using the ability of heuristic algorithm to quickly find the approximate optimal solution, while in the Node Pairing (NP) step we propose a generative graph convolutional network with a novel Aggregate Cooperative (AC) layer to generate a set of nodes that meet the constraints, so as to obtain the perturbation set together with the Node Searching step. The proposed SPA method outperforms the state-of-the-art adversarial attack methods and achieves a misclassification rate of 32.5% in the poisoning attack on Cora dataset with a perturbation rate of 0.5%.

Keywords Graph convolutional networks · Machine learning security · Grey-box poisoning attack · Adversarial attack

Editor: Lijun Zhang.

✉ Wentao Zhao
wtzhao@nudt.edu.cn

[1] College of Computer Science and Technology, National University of Defense Technology, Deya Road, Changsha 410073, Hunan, China

1 Introduction

Graph-based data have provided an effective and flexible way to represent and analyze complex relationships between entities, such as social networks (Rossetti et al., 2017; Meng et al., 2019), academic publishing networks (McCallum et al., 2000; Sen et al., 2008), knowledge graph (Wu et al., 2020), and recommender systems (Harper & Konstan, 2015; Baltrunas et al., 2015). In a graph, nodes (also known as vertices) and edges are used to represent entities and their relationships, respectively. Compared to data in Euclidean space, non-Euclidean graph data could contain richer and more flexible relationships between elements that are difficult to be processed by Deep Neural Networks (DNNs) (Du et al., 2020; Yi et al., 2022) directly.

Graph Neural Networks (GNNs) can be used to effectively extract, process and analyze information in graph data. Its rapid development has facilitated the widespread use of graph analysis on real world tasks, including node-level tasks (Zhao et al., 2021; Xie et al., 2022; Zhou et al., 2019) which aim to predict characteristics in nodes, link-level tasks (Rossi et al., (2021); Cai & Ji, 2020) which aim to predict the connectivity between nodes and graph-level tasks (Xie et al., 2020; Wu et al., 2022) which focus on predicting properties of graphs. As the most representative GNN model, Graph Convolutional Networks (GCNs) have achieved state-of-the-art performance on node-level tasks. However, the existing research suggests that adversarial attacks can cause erroneous predictions in DNNs, which are mainly used in processing Euclidean data, by adding mild adversarial noise to the benign data. Similarly, GCNs are also susceptible to adversarial attacks. Since graphs possess a more flexible connection relationship, it is possible to malfunction GCNs by modifying the graph features (i.e., modifying feature matrix) as well as the connectivity (i.e., modifying adjacency matrix). Adversarial attacks on graphs can be categorized into two main categories: graph modification attacks (Liu et al., 2019; Chang et al., 2022; Wang et al., 2022) and graph injection attacks (Wang et al., 2021; Dai et al., 2022; Liu et al., 2024). The graph modification attacks generate adversarial examples by modifying the existing edges and features in the graph data, while the graph injection attacks achieve adversarial attacks by generating and injecting fake nodes and edges into the graph data. In this paper, we focus on white-box graph modification attack that modifies edges in static graphs. In the white-box setting, the assumption of extensive prior knowledge may not always be held in real-world situations. However, individuals with sufficient knowledge of a system, such as employees or contractors, may exploit that the knowledge for malicious purposes. There are also some security researchers that perform white-box analyses to identify vulnerabilities and improve security measures (Gosch et al., 2024; Goodfellow et al., 2014).

The gradient-based attack methods (Moosavi-Dezfooli et al., 2016; Su et al., 2019) have been shown to be highly effective in fooling DNNs across a wide range of applications for Euclidean data (Moosavi-Dezfooli et al., 2017; Luo et al., 2022; Chen et al., 2022). It typically involves computing the gradient of the loss function with respect to the input and then perturbing the input in the direction of the gradient so as to maximize the loss. Gradient-based methods can also be used for adversarial attacks on GNNs. In graph modification attacks, since the gradient values cannot be directly added to the binary adjacency matrix, some adversarial attack methods evaluate the attack potential of an adversarial edge based on the magnitude of the gradient value and modify the adjacency matrix edge-by-edge using greedy-based manners. Dai et al. (2018) first propose a gradient-based method, called GradArgmax, to iteratively select the adversarial edge corresponding to the largest

gradient value and retrain the surrogate model for the next iteration. Wu et al. (2019) introduce integrated gradients (Sundararajan et al., 2017), which integrate partial gradients with respect to input features from reference input to the actual input, and generalize the traditional adversarial attack methods, such as Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014) and Jacobian Saliency Map Attack (JSMA) (Papernot et al., 2016), to adopt to GNNs. Chen et al. (2018) propose to extract the gradient of pair-wised nodes based on the adversarial network and select the pair of nodes with a maximum absolute gradient to achieve Fast Gradient Attack (FGA).

However, due to the discrete nature of the adjacent matrix, gradient-based methods are facing two challenges: i) Perturbing the adversarial edge with the maximum gradient value is not guaranteed leading to the maximum loss variation; ii) The greedy-based adversarial attack methods often lack of global perspective thus lead to the methods becoming ensnared in local optima. The first challenge has been shown by Lin et al. (2023), according to which the gradient-based strategy that relies on the maximal gradient of the training loss may not generate the best results when attacking GNNs. For the second challenge, we illustrate it with experimental results of different adversarial attack methods on the KarateClub dataset (Zachary, 1977) in Table 1. The KarateClub dataset consists of 34 nodes (3 for training) and 78 edges. We perform the node classification task using a two-layer GCN model. Among the adversarial attack methods, Mettack (Zügner & Günnemann, 2019), PGD (Xu et al., 2019)), and Min-Max (Xu et al., 2019) are gradient-based adversarial attack methods, the Ant Colony Optimization (ACO) and the Genetic Algorithm (GA) are heuristic algorithms. Through comparative experiments, we have observed that heuristic methods outperform gradient-based adversarial attack methods in terms of misclassification rate. Although heuristic methods do not guarantee the discovery of optimal solutions, our experimental results demonstrate that they possess a stronger capability to escape from local optima. In terms of runtime performance, a direct comparison between heuristic and gradient-based methods is challenging due to disparities in hardware acceleration and code implementation. However, when evaluated using the publicly available codebase, heuristic methods exhibit a noticeable disadvantage in efficiency.

Overall, the gradient-based approach can generate adversarial examples with a fast processing speed, but the unreliability of the gradient on the graph data limits the performance of the gradient-based adversarial attacks. To address the limitations of the unreliable gradient, in this paper we consider an alternative by utilizing Heuristic Algorithms (Zügner et al., 2018; Dai et al., 2018; Chen et al., 2019; Yang & Long, 2021) for graph modification attack. Heuristic algorithms are problem-solving techniques that find a solution by exploring all possible paths and evaluating them based on a set of predefined rules or guidelines. By introducing domain-specific knowledge, heuristic algorithms can be adapted to different scenarios to efficiently obtain the solution. In graph adversarial attacks, the attacker aims to search for a perturbation set from all potential perturbations to achieve the best

Table 1 Misclassification rate (MSR, %) on GCN node classifier with gradient optimization attack methods (MettackZügner & Günnemann, 2019), PGD (Xu et al., 2019), Min-Max (Xu et al., 2019), and heuristic algorithm methods(ant colony optimization, genetic algorithm

Attack	Clean	Mettack (Zügner & Günnemann, 2019)	PGD (Xu et al,. 2019)	Min-Max (Xu et al., 2019)	ACO	GA
MSR	20.0	34.0	28.3	30.7	49.0	37.3
Time	–	1.01s	0.48s	1.01s	17.98s	38.29s

attack, which is also an optimization problem that can be solved by heuristic algorithms. Compared with gradient-based methods, heuristic-based methods optimize perturbations according to loss values or query results, avoiding the unreliability of gradients in graph adversarial attacks. In addition, heuristic-based methods treat the perturbation set as a whole and optimize multiple perturbations simultaneously, which can escape from the local optimum. However, since the heuristic algorithms are sensitive to the size of the search space, the heuristic-based methods have high computational complexity. Besides, the heuristic algorithms usually provide approximate solutions rather than guaranteeing the absolute optimal solution. Therefore, the heuristic algorithms are mainly used in the black-box adversarial attack setting when the gradients are not available.

We propose to solve the graph modification attacking problem by combining the merits of the generative model and heuristic algorithms. Specifically, we decompose the perturbation edge set into two perturbation node sets, which are generated by the generative model and the heuristic algorithm, respectively. Graph generative models are machine learning models that aim to capture the underlying structure and relationships within a given set of graphs and generate new nodes that meet the attacker's requirements. In a graph with N nodes, the number of potential adversarial edges can approach to N^2. Consequently, by decomposing the target problem, we transform the problem with complexity of $O(N^2)$ into two sub-problems, each with a complexity of $O(N)$. Although this decomposition does not reduce the total complexity, it offers distinct advantages of addressing the two sub-problems concurrently with different methods. For the generative model, reducing the problem space simplifies the computational process and constricts the search space for optimization, thereby streamlining the training phase and enhancing the overall performance. For heuristic algorithms, reducing the problem space significantly accelerates the operational speed of the algorithm allowing the algorithm to explore fewer options more intensively, thereby increasing the efficiency of the search process. In general, decomposing the problem to two sub-problems allows us to balance performance and computational cost, effectively optimizing the solution by leveraging the unique strengths of each approach, rather than simply relying on one single method that may not be efficient. Furthermore, heuristic algorithms eschew gradient computation, and generative models use gradient to optimize the continuous model parameters. Thus, the proposed adversarial attack method avoids the unreliability associated with optimizing discrete adversarial perturbations using continuous gradients.

In this paper, we provide a novel perspective by converting the hard adversarial edge search problem into two simpler Node Searching and Node Pairing problem with a searching space of $\mathcal{O}(N)$. Specifically, the proposed Searching and Pairing Attack (SPA) method contains two main steps: a Node Searching (NS) step and a Node Pairing (NP) step. The Node Searching step searches a subset $\mathcal{S}$ of adversarial nodes from all nodes. Then, the Node Pairing step finds a corresponding node for each node in $\mathcal{S}$ to form an adversarial edge. In the Node Searching step, we propose a Node Searching Ant Colony Optimization (NS-ACO) algorithm to find the node set which can obtain excellent attack performance. Inspired by graph generative model, we train a generative model in the Node Pairing step to find the corresponding nodes. In the generative model, gradients are used to optimize the model parameters instead of the discrete graph structure, which alleviates the problem of directly using unreliable gradient values.

We summarize the contributions of this paper as follows:

- We propose a novel Searching and Pairing attack method that decomposes the adversarial edge generation problem with complexity of $\mathcal{O}(N^2)$ into two simpler node gen-

eration problems with complexity of $\mathcal{O}(N)$. This decomposition allows the use of algorithms that are sensitive to problem space in adversarial attacks.
- The proposed SPA method contains two main steps: the Node Searching step uses an Ant Colony Optimization (ACO) algorithm to effectively find the adversarial node set, and in the Node Pairing step, we design a novel generative model that uses a graph convolutional network to aggregate node features and generates a corresponding node set for the given node set.
- Extensive experiments on GCNs for node classification tasks show that the proposed SPA method has achieved higher attack capacity compared with the state-of-the-art adversarial attack methods. The SPA method with 5% perturbation rate can cause the misclassification rate of 32.5%, 27.0%, 37.4%, and 26.7% on Cora, Cora-ML, Citeseer, and Polblogs datasets, respectively.

The rest of the paper is organized as follows: Sect. 2 reviews the adversarial attack methods against Graph Neural Networks. Section 3 presents some preliminary backgrounds. Section 4 introduces the details of the proposed SPA method. Section 5 shows the experimental results to evaluate the performance of our attack method. Section 6 concludes this paper.

2 Related work

In the following, we briefly introduce gradient-based attack methods and heuristic-based attack methods for Graph Neural Networks.

2.1 Gradient-based attack methods

Though gradient-based adversarial attack methods are effective on fooling Deep Neural Networks, they cannot be directly applied for graph modification attacks. Dai et al. (2018) first explore the gradient-based adversarial attack and propose the GradArgmax attack method which greedily and sequentially selects the adversarial edge corresponding to the maximum gradient in each iteration and then modifies the adjacency matrix according to the perturbed edge. Zügner and Günnemann (2019) propose the Meta Attack (Mettack) method which uses meta-gradients to select adversarial edges rather than gradient. Xu et al. (2019) propose to relax the binary adjacency matrix to a continuous matrix, and propose to perturb the adjacency matrix using gradient and then obtain the adversarial edge set by sampling. citelin2023exploratory report the unreliability issue of gradients in adversarial edges generation and propose Exploratory Adversarial Attack (EpoAtk) method that has a generation step, an evaluation step and a recombination step to avoid being misguided by the maximum gradient information. Liu et al. (2023) suggest to avoid spending too much attack budget on low confidence nodes by adding weights to the Cross-Entropy loss. Liu et al. (2022) analyze the grey-box attacks and propose Attacking by Shrinking Errors (AtkSE) method which aims to reduce the error caused by edges' discreteness, error caused by uncertainty of model optimization, and error caused by model's unrobustness. The above attack methods alleviate the unreliability of gradients in graph adversarial attacks, but still do not consider the cooperation relationship between the adversarial edges, which makes the attacks fall into the local optimum.

2.2 Heuristic-based attack methods

Although gradient-based methods are widely used in adversarial attacks, there are cases where the gradient is unavailable (in black-box attacks) or unreliable (as described in the previous section). Therefore, heuristic-based methods are proposed to be applied to generate adversarial examples for GNNs.

Current heuristic-based attacks are mostly applied in the black-box setting. Dai et al. (2018) first propose a genetic algorithm based adversarial attack method named GeneticAlg, which treats the set of modified graphs as populations, obtains the fitness of each graph by querying the attack results and then optimizes the perturbations by crossover and mutation operations. Yu et al. (2020) propose a genetic algorithm based Euclidean Distance Attack strategy (EDA) attack, which is able to maximize the Euclidean distance of nodes in the embedding space. Chen et al. (2019) propose a genetic algorithm based rewiring attack Q-Attack for attacking community detection algorithms.

In recent years, researchers have also explored the patterns of adversarial attacks on graphs to generate heuristic attacks. Through statistical analysis of the generated adversarial edges, Wu et al. (2019) observe that adding edges is more threatening than removing ones. Zügner et al. (2020) further show that the added edges are mostly between nodes with different classes. Zhan and Pei (2021) find untargeted attacks tend to perturb the graph unevenly such that the attacks modify a higher ratio of edges near the training set. Based on the above observations, Haoxi and Xiaobing (2021) propose a heuristic poisoning attack Fast Heuristic Attack (FHA) method which perturbs the adjacency matrix by connecting nodes with different labels and the unlabeled nodes are assigned pseudo-labels through a GCN model. The above heuristic algorithm based adversarial attack methods use loss values to evaluate the quality of perturbations instead of gradients, which avoids the unreliability of gradients. But the high time complexity of the heuristic algorithm makes it difficult to be applied in poisoning attacks. To the best of our knowledge, this is the first time that heuristic algorithms are introduced for poisoning attacks by decomposing the problem space. Benefiting from the advantage of the heuristic algorithm to escape from the local optimum, our SPA method is able to obtain better attack performance.

2.3 Integrating generative models with heuristic algorithms

Heuristic algorithms offer efficient approximations for tackling complex discrete problems and have played a significant role across various domains. However, the utility of heuristic algorithm is often confined to specific problem types, which restricts their adaptability.

Researchers have proposed to integrating generative models with heuristic algorithms. By leveraging generative models to learn the problem structure, the applicability of heuristic algorithms can be significantly broadened. Li et al. (2021) utilizes a novel recurrent generative model (RGM) that generates efficient heuristics to guide the sampling process of path planning algorithms, significantly improving performance in various 2D environments. Collins et al. (2024) present a method that integrates generative machine learning with heuristic crystal structure prediction, demonstrating faster compute times and lower energies for known compounds and hypothetical ones. Zhang et al. (2021) discuss the application of generative adversarial networks (GANs) for generating heuristics in sampling-based path planning algorithms, which is shown to be effective in predicting promising regions for non-uniform sampling. Wang et al. (2023) propose a

generative inverse reinforcement learning method for learning 2-opt heuristics in combinatorial optimization problems, combining GANs and deep reinforcement learning to learn effective policies and reward functions.

In this paper, we convert the adversarial edge search problem to two adversarial node search problems, allowing the heuristic algorithm to focus on searching for adversarial node sets within a problem space of $\mathcal{O}(N)$. Subsequently, we employ a generative model to obtain the corresponding node sets. This integrated approach reduces the search space for the heuristic algorithm, enhances its operational speed, and thereby achieves a more effective poisoning attack.

3 Preliminaries

In this section, we give some preliminaries on Graph Convolutional Networks (GCNs) and adversarial attacks on graph data. Prior to delving into the specifics, we provide a summary of the frequently used notations in Table 2.

Table 2 The definitions or descriptions of notations

Notation	Description
$G = \{A, X\}$	Graph data
A	Adjacency matrix
X	Node attribute matrix
N	Number of nodes
F	Number of features
$\mathcal{V}$	Set of nodes
$\mathcal{E}$	Set of edges
$\mathcal{Y}$	Labels of nodes
v	A node
y	Class label of a node
$\mathcal{V}_{train}$	Training set of nodes
$\mathcal{V}_{test}$	Testing set of nodes
$\mathcal{S}$	Adversarial nodes searched by NS step
$\mathcal{P}$	Adversarial nodes searched by NP step
$\mathbf{S}$	One-hot embedding of $\mathcal{S}$
$\mathbf{P}$	One-hot embedding of $\mathcal{P}$
f_θ	GCN model with parameter θ
Δ	Perturbation number
A'	Adversarial adjacency matrix
$G' = \{A', X\}$	Adversarial graph
$L'(\mathcal{S}, \mathcal{P})$	Adversarial loss with adversarial nodes $\mathcal{S}$ and $\mathcal{P}$
m	Number of ants in ACO
τ_i	Phenonment of node v_i
$\mathcal{O}(\cdot)$	Size of the problem space

3.1 Graph convolutional network

Let $G = \{A, X\}$ denote an attributed graph with adjacency matrix $A \in \{0,1\}^{N\times N}$ and attribute matrix $X \in \{0,1\}^{N\times F}$. N is the number of nodes and F is the dimension of node feature vector. The graph can also be represented by node set $\mathcal{V}$ and edge set $\mathcal{E}$, where $A_{i,j} = 1$ if there is an edge between node v_i and v_j otherwise 0.

Given a subset of labeled nodes $\mathcal{V}_{train} \subseteq \mathcal{V}$ with labels from K classes $\mathcal{Y} = \{y_1, y_2, \cdots, y_K\}$, the objective of GCNs (Kipf & Welling, 2016) is to learn a function $f_\theta : \mathcal{V} \to \mathcal{Y}$ which maps each node $v \in \mathcal{V}$ to a class in $\mathcal{Y}$. The GCN model is a typical transductive learning method since it leverages information from both the labeled and unlabeled nodes to train the model.

GCN learns the feature representation of each node by aggregating and transforming the information from its neighbor nodes. The $(l+1)$-th graph convolution layer of a GCN model can be expressed as:

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

where $\widetilde{A} = A + I_N$ is the adjacency matrix of the input graph G with self-loops, I_N being the identity matrix, $\widetilde{D} \in \mathbb{R}^{N\times N}$ is a diagonal matrix with $\widetilde{D}_{i,i} = \sum_j \widetilde{A}_{i,j}$, $W^{(l)} \in \mathbb{R}^{F\times h_l}$ represents the trainable weights, h_l is the hidden size in the $l-th$ layer, and $\sigma(\cdot)$ is the activation function, $H^{(0)}$ is set to the node attribute matrix X.

Following the surrogate model in Zügner et al. (2018), a two-layer GCN model with linear activation function is used for node classification and can be described as:

$$f_\theta(v_i, G) = \text{softmax}(\widehat{A}\widehat{A}XW^{(0)}W^{(1)})[i], \tag{2}$$

where $G = \{A, X\}$, $v_i \in \mathcal{V}$ is a node, $\widehat{A} = \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}$ and $\theta = \{W^{(0)}, W^{(1)}\}$ denotes the set of the input-to-hidden weights $W^{(0)}$ and the hidden-to-output weights $W^{(1)}$. $f_\theta(v_i, G)$ donates the predicted result of node v_i in graph G. By eliminating the ReLU layer, we aimed to simplify the surrogate model, which help the model to more directly learn the mapping between inputs and outputs without the complexity introduced by the ReLU nonlinearity. Besides, the removal of the ReLU layer can enhance the accuracy of gradient estimation. Since ReLU introduces zero gradients for negative inputs, their removal can provide a clearer gradient signal during the adversarial attack process. Liu et al. (2023) have discussed the wrong blocking and over transmission in the ReLU derivation processes.

3.2 Graph modification attack

A graph modification attack is achieved by adding or removing edges and modifying node features, with the objective of manipulating the outputs produced by the GNN model to achieve the desired results of the attacker. As adding or removing edges plays a main role in graph modification attacks (discussed by Wu et al. (2019)), we focus on the graph modification attacks that only involve adding or removing edges in the graph, i.e., $G' = \{A', X\}$. In this paper, we perform a white-box poisoning graph modification attack targeting on the node classification task and aim to reduce the classification accuracy on the testing set. White-box attack means that the attacker shares the same information with the model trainer, such as the dataset or hyperparameters of the GCN model. In a poisoning attack, the GCNs are trained on a modified graph G' so that the model parameters will

be influenced during the training phase, which finally leads to a high misclassification rate in the testing phase. We use a two-layer GCN model with linear activation function as the surrogate model. The adversarial attack can be described as follows:

$$\begin{aligned} &\max_{G' \in \Phi(G)} \sum_{v_i \in \mathcal{V}_{test}} \mathcal{L}(f_{\theta^*}(v_i, G'), y_i) \\ &s.t.\ \theta^* = \arg\min_{\theta} \sum_{v_j \in \mathcal{V}_{train}} \mathcal{L}(f_\theta(v_j, G'), y_j), \end{aligned} \tag{3}$$

where f_θ is the GCN model with learnable parameters θ shown in Eq. (2), $f_{\theta^*}(v_i, G')$ is the prediction of node v_i in graph G' with model parameter θ^*, $\Phi(G)$ is the set of admissible perturbations on the graph G, $\mathcal{L}(\cdot, \cdot)$ denotes the loss function, $\mathcal{V}_{test}$ and $\mathcal{V}_{train}$ denotes the testing set and the training set, respectively. The objective function in Eq. (3) indicates that the attacker searches the adversarial graph G' allowed to maximize the testing loss of the surrogate model. The constraint suggests that parameter θ^* is trained by minimizing the training loss with G'. Since the ground-truth label of testing data is not available, we use pseudo-labels of the testing set generated by the surrogate model as an alternative.

4 Proposed method

4.1 Overview

In this paper, we propose a Searching and Pairing Attack (SPA) to effectively decompose the hard adversarial edge generation problem into two simpler adversarial node generation problems. Our approach is based on the fact that an adversarial edge can be constructed with two adversarial nodes, whereas the quality of both adversarial nodes are essential to the final generated adversarial edge. Therefore, the overall architecture of our SPA method for graph modification attack contains two main steps: a Node Searching (NS) step to search the initial adversarial node set and a Node Pairing (NP) step to find the paring adversarial node set with the given initial node set. Specifically, given a graph G, the NS step aims to search an adversarial node set $\mathcal{S} = \{v_{s1}, v_{s2}, \cdots, v_{s\Delta}\}$ with a maximum number of perturbation Δ. With $\mathcal{S}$, the NP step aims to find a corresponding adversarial node set $\mathcal{P} = \{v_{p1}, v_{p2}, \cdots, v_{p\Delta}\}$. The nodes within set $\mathcal{P}$ are sequentially paired with those in set $\mathcal{S}$ on a one-to-one basis, such that v_{s1} corresponds to v_{p1}, and v_{si} corresponds to v_{pi}. The adversarial edges $\mathcal{E}' = \{e'_1, e'_2, \cdots, e'_\Delta\}$ can then be obtained from the adversarial node sets $\mathcal{S}$ and $\mathcal{P}$ where the adversarial edge $e'_i = (v_{si}, v_{pi})$ adds or removes the edge between node v_{si} and v_{pi}. We have thus transformed the challenging adversarial edge search problem into two simpler Node Searching (NS) and Node Pairing (NP) sub-problems.

It should be noted that although Node Searching and Node Pairing are both indispensable components of the proposed SPA method, their functions are distinct. The objective of Node Searching process is to search a high quality adversarial node subset $\mathcal{S}$ from the node set $\mathcal{V}$. The Node Pairing step involves generating a pairing adversarial node subset $\mathcal{P}$ based on the condition of nodes in $\mathcal{S}$ in a one-to-one manner, which can be considered as a conditional generation problem. Effective algorithms for solving both the NS and NP sub-problems is essential, and the selection of both adversarial node sets $\mathcal{S}$ and $\mathcal{P}$ significantly influence the attacking performance. For instance, randomly selected node sets $\mathcal{S}$ and $\mathcal{P}$ can also form an adversarial edge set, the adversarial graph generated in this manner are unlikely to effectively

impact the target model. In Sect. 5.3.1, we provide a detailed analysis of how the NS and NP steps individually influence the effectiveness of the SPA method. Experimental results show that when a randomly selected node set $\mathcal{S}$ is employed, the misclassification rate would decrease up to more than 9%.

Considering both the model effectiveness and efficiency, we propose to use the Ant Colony Optimization (ACO) algorithm to construct NS step in order to improve global searching capability and propose a generative-based NP model to improve the generation speed. The SPA method shown in Fig. 1 involves a training phase and a testing phase. In the training phase, the NP model is first trained with randomly sampled $\mathcal{S}$. The well-trained NP model is then used to update the pheromone list of NS algorithm. In the testing phase, Δ nodes with top pheromone scores are selected to form adversarial node set $\mathcal{S}$. Subsequently, the NP model is utilized to generate $\mathcal{P}$, and ultimately, the adversarial graph G' is obtained. By effectively combing Node Searching and Node Pairing, the proposed Searching and Pairing Attach (SPA) can achieve SOTA adversarial attack performance.

In both NS and NP steps, the adversarial loss $L'(\mathcal{S}, \mathcal{P})$ is used as the unified attack performance indicator, which is defined based on the training loss:

$$\begin{aligned} L'(\mathcal{S}, \mathcal{P}) &:= -\sum_{v_i \in \mathcal{V}_{train}} \mathcal{L}(f_{\theta^*}(v_i, G'), y_i), \\ s.t.\ \theta^* &= \arg\min_{\theta} \sum_{v_j \in \mathcal{V}_{train}} \mathcal{L}(f_{\theta}(v_j, G'), y_j), \\ G' &= \{A', X\}, \end{aligned} \tag{4}$$

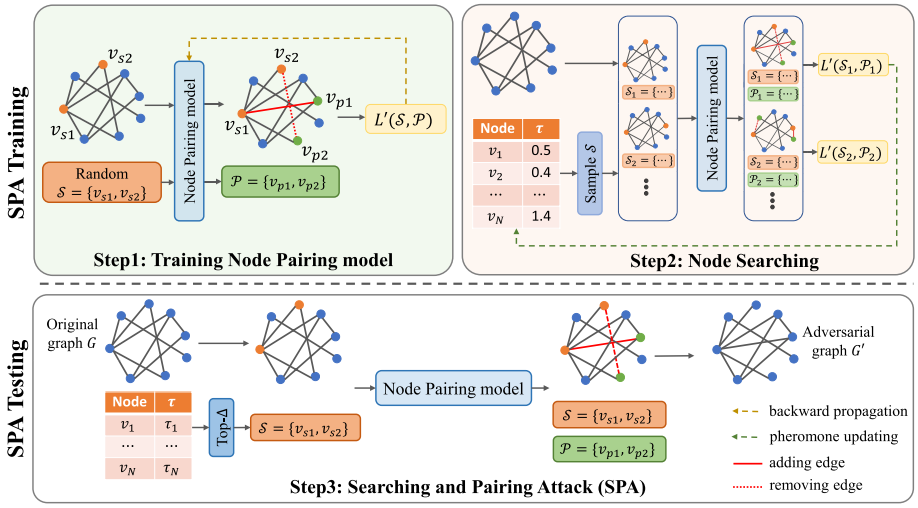


Fig. 1 The flow chart of the Searching and Pairing attack. In the training stage, the Node Pairing step learns a generative model which can generate node set $\mathcal{P}$ (green nodes) for a given node set $\mathcal{S}$ (red nodes), then the Node Searching step uses the Ant Colony Optimization algorithm to update the pheromone list by sampling the node set $\mathcal{S}$ and computing adversarial loss $L'(\mathcal{S}, \mathcal{P})$. In the testing stage, the proposed SPA method selects Δ nodes with the highest pheromone values to form the adversarial node set $\mathcal{S}$, which is used as the input of the NP model to generate the adversarial node set $\mathcal{P}$. The two adversarial node sets together form the adversarial perturbations to generate the adversarial graph

where A' is obtained by modifying A according to $\mathcal{S}$ and $\mathcal{P}$, $f(\cdot,\cdot)$ is the surrogate model defined in Eq. (2), $f_{\theta^*}(v_i, G')$ means the prediction of node v_i in graph G' with model parameter θ^*. In Eq. (4), the constraint describes the training process of the surrogate model. Given the adversarial graph G', the well-trained model parameter θ^* can achieve the minimized training loss. After model training, the loss function calculates the cross-entropy loss on the training set with model parameter θ^* and takes its negative value as the adversarial loss. The attacker aims to disrupt the training process of the target model as much as possible. To this end, the optimization objective of adversarial graph G' is minimizing the adversarial loss, which equals maximizing the training loss of the well-trained surrogate model.

In the remaining part of this section, we introduce the specifics of the NS and NP methodologies, and then we propose an iterative attack strategy designed to address scenarios with high perturbation rates.

4.2 Node searching algorithm

In this section, we introduce the proposed Node Searching method based on the Ant Colony Optimization (ACO) (Colorni et al., 1991) algorithm. Compared with gradient-based attach approach, the ACO algorithm aims to find a combination of perturbations rather than add perturbations to the perturbation set iteratively, which leads to better attack results. Ant Colony Optimization is a heuristic algorithm that uses the behavior of ants to solve optimization problems. The algorithm is inspired by the foraging behavior of ants, where they deposit pheromones on the ground as they explore their environment. By using this pheromone communication, ants can find the shortest path to a food source and create trails for other ants to follow.

In the proposed Node Searching ACO (NS-ACO) method, the adversarial loss $L'(\mathcal{S}, \mathcal{P})$ is treated as the reward for the NS-ACO algorithm to search adversarial node set $\mathcal{S}$. Before introducing the implementation of NS-ACO, we first give the definition of ants, reward, and pheromone in our problem.

- **Ant:** We define the adversarial node set $\mathcal{S}$ as the ant in NS-ACO algorithm. m is the number of ants in the ant colony. Each ant selects Δ nodes from node set $\mathcal{V}$ with the pheromones τ as the probabilities to form the solution $\mathcal{S}$ of the target problem.
- *Reward* In NS-ACO algorithm, reward is used to measure the quality of the solutions. Since a larger reward typically corresponds to a better solution, we use the negative adversarial loss $-L'(\mathcal{S}, \mathcal{P})$ as the reward of the solution $\mathcal{S}$.
- *Pheromone* The key in the NS-ACO algorithm lies in the updating of the pheromone list, as ants explore the problem space and deposit pheromones τ to indicate the quality of the solutions they find. For example, after getting reward $-L'(\mathcal{S}, \mathcal{P})$, the ant with solution $\mathcal{S}$ leaves pheromones on the nodes contained in $\mathcal{S}$ according to the reward to improve their probability of being selected. A pheromone list is used to record and quantify the contribution of each node $v_i \in \mathcal{V}$ in executing the poisoning attack.

The goal of the NS-ACO algorithm is therefore to find a subset $\mathcal{S} \subset \mathcal{V}$ that maximizes the reward:

$$\begin{aligned} &\max_{\mathcal{S}} \ -L'(\mathcal{S},\mathcal{P}), \\ &s.t. \ \ \mathcal{S}\subseteq\mathcal{V}, |\mathcal{S}|=\Delta. \end{aligned} \tag{4}$$

where $\mathcal{P}=NP(\mathcal{S})$ denotes node set generated by the Node Pairing model given the node set $\mathcal{S}$.

As depicted in Eq. (4), the assessment of the reward necessitates the utilization of the Node Pairing model. Accordingly, as illustrated in Fig. 1, training Node Pairing model is the first step for training SPA. With an efficacious Node Pairing model in place, the pheromone list within our proposed NS-ACO algorithm is optimized. Further insights into the Node Pairing mechanism will be elaborated in the subsequent section.

Algorithm 1 Node searching ACO algorithm

Require: Node Pairing model $NP(\cdot)$, graph G, iteration number T, ant number m, evaporation rate η
Ensure: node set $\mathcal{S}$
1: Initialize τ_j^0 with initial concentration C for $v_j \in \mathcal{V}$
2: **for** $t = 1$ to T **do**
3: **for** $i = 1$ to m **do**
4: $\mathcal{S}_i^t \leftarrow$ Sample Δ nodes from $\mathcal{V}$ based on τ^t
5: $\mathcal{P}_i^t \leftarrow NP(\mathcal{S}_i^t)$
6: evaluate $L'(\mathcal{S}_i^t, \mathcal{P}_i^t)$
7: **end for**
8: update τ^t according to Eq. (5)
9: **end for**
10: $\mathcal{S} \leftarrow Top-\Delta$ nodes in τ^T
11: **return** $\mathcal{S}$

Algorithm 1 shows the details of the proposed Node Searching ACO algorithm. We initialize the pheromones with a constant to prevent the algorithm from falling into a local optimum prematurely. At the beginning of each iteration, a group of virtual ants is released onto the problem space, each ant selects Δ nodes with normalized pheromone values as probabilities to form the adversarial node set $\mathcal{S}_i$. For each set $\mathcal{S}_i$, the Node Pairing algorithm searches the corresponding pairing node set $\mathcal{P}_i$. Then the algorithm generates the adversarial graph and calculates $-L'(\mathcal{S}_i, \mathcal{P}_i)$ as the reward of $\mathcal{S}_i$. At the end of each iteration, the pheromones on the nodes will evaporate with the rate η and are updated based on the reward of the solutions:

$$\begin{aligned} \tau_i^t &= \eta\tau_i^{t-1} + \sum_{j=1}^{m} e_{i,j}^{t-1}, \\ e_{i,j}^{t-1} &= \begin{cases} 0 & v_i \notin \mathcal{S}_j, \\ -L'(\mathcal{S}_j^{t-1}, \mathcal{P}_j^{t-1}) & v_i \in \mathcal{S}_j, \end{cases} \end{aligned} \tag{5}$$

where m is the number of ants in the algorithm. In the final step of the NS-ACO algorithm, the adversarial node set $\mathcal{S}$ consisting of Δ nodes with the highest amount of pheromone is considered as the solution to the optimization problem.

During the NS-ACO algorithm, the reward $-L'(\mathcal{S}, \mathcal{P})$ achieved by the node set $\mathcal{S}$ is treated as the attack capability of all the nodes contained in $\mathcal{S}$. For a node with higher attack capability, the node set in which it is located is able to obtain a higher reward, thus assigning a higher pheromone to that node. In the proposed SPA attack, we therefore treat the Δ nodes with the highest pheromone values as adversarial node set with the highest attack capability and obtain the final $\mathcal{S}$.

4.3 Node pairing algorithm

The proposed Node Pairing (NP) algorithm aims to find a corresponding node set $\mathcal{P} \subset \mathcal{V}$ with Δ nodes given an adversarial node set $\mathcal{S}$. In this section, we propose a generative-based NP model that extracts and aggregates features of nodes in $\mathcal{S}$ to generate $\mathcal{P}$. By training the feature extraction layer and aggregation weights, the NP model can learn the effects of v_{si} and $\mathcal{S}$ on v_{pi} to generate $\mathcal{P}$ with higher attack performance. Note the adversarial node $v_{pi} \in \mathcal{P}$ subject to the following constraints: v_{pi} will form an adversarial edge with v_{si} (i.e., delete edge (v_{si}, v_{pi}) from G if the edge exists, otherwise add the edge), and form a perturbation set $\mathcal{E}'$ together with other adversarial edges.

The proposed generative Node Pairing model is illustrated in Fig. 2. It contains two branches: the upper branch extracts features from the input graph with two graph convolutional layers, while the lower branch takes the node set generate by Node Searching step and aggregate cooperative features from the intermediate features of the upper branch. The two set of features are then combined and used to predict the pairing node set. The pairing node set $\mathcal{P}$, in conjunction with the input node set $\mathcal{S}$, forms the adversarial edges. Subsequently, a surrogate model is trained on the perturbed graph G' to calcualte the adversarial loss $L'(\mathcal{S}, \mathcal{P})$. During the backpropagation phase, the model computes the gradient of the adversarial loss with respect to the parameters of the NP model and optimizes the model parameters using the Adam optimizer to minimize the adversarial loss.

4.3.1 Model architecture

In the upper branch, the graph convolutional layer defined in Eq. (1) is used to extract the features (which are defined by blue and orange bars in Fig. 2):

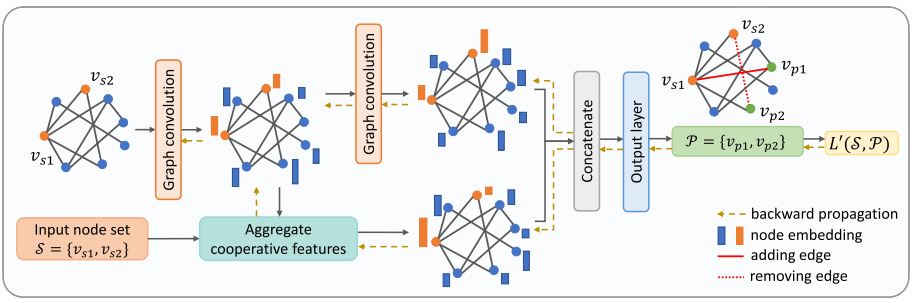


Fig. 2 Overview of the Node Pairing model. Here, we illustrate the Node Pairing model with $\Delta = 2$ adversarial nodes. The model contains two graph convolution layers to extract features of the nodes. To introduce cooperation between adversarial nodes, the model aggregates the features of the adversarial nodes through a learnable $\Delta \times \Delta$ matrix and finally uses Gumble-Softmax to obtain corresponding nodes

$$H^{(1)} = \sigma(\widehat{A}XW^{(0)}), \tag{6}$$

where $H^{(\cdot)}$ is the node embedding, $W^{(\cdot)}$ is the parameter of the NP model.

In the lower branch, the cooperative relationship between adversarial nodes are considered in generating adversarial node set $\mathcal{P}$ since the adversarial perturbations work jointly to implement an adversarial attack. We propose a Aggregate Cooperative (AC) Layer by using a learnable weight matrix $W^{(coop)}$ to aggregate cooperative features between adversarial nodes in $\mathcal{S}$:

$$H^{(coop)} = W^{(coop)}\mathbf{S}H^{(1)}, \tag{7}$$

where the $W^{(coop)} \in \mathbb{R}^{\Delta\times\Delta}$ is the learnable matrix, and $\mathbf{S} \in \{0,1\}^{\Delta\times N}$ is the one-hot embedding matrix of $\mathcal{S}$. Specifically, the i-th row of $\mathbf{S}$ indicates the one-hot embedding of adversarial node v_{si}, the index of value 1 indicates the position of v_{si} in the graph. Since each row in the feature matrix $X \in \{0,1\}^{N\times F}$ indicates a feature vector of a node, $X[j]$ is the feature of node v_{si} if $\mathbf{S}[i][j] = 1$. Thus, $\mathbf{S}H^{(1)}$ extracts the embedding of the nodes in $\mathcal{S}$ after the graph convolutional layer. Then $W^{(coop)}$ aggregates the features of different nodes in $\mathcal{S}$ to finally get the cooperative features.

Finally, we concatenate the node features with the cooperation features to generate the corresponding pairing nodes:

$$\begin{aligned} H^{(2)} &= \mathbf{S}\sigma(\widehat{A}H^{(1)}W^{(1)})\copyright H^{(coop)}, \\ Z &= H^{(2)}W^{(2)}, \end{aligned} \tag{8}$$

where © is the concatenate operation. $Z \in \mathbb{R}^{\Delta\times N}$ is the output matrix in which each row represents the probability distribution of the corresponding node for the input nodes in $\mathcal{S}$.

In order to convert probabilities $z_k \in Z$ to a one-hot vector, a common practice is to use the arg max function:

$$\arg\max_i z_k := \{i | \forall j : z_k[j] \le z_k[i]\}. \tag{9}$$

Since the arg max function is non-differentiable, the gradient cannot be back-propagated through it. In the output layer, we use a binary special case of the Gumbel-Softmax reparameterization trick which is soft and differentiable as an alternative. The output of the NP model is:

$$\mathbf{P} = \mathrm{GS}(\mathbf{Z}), \tag{10}$$

where $\mathrm{GS}(\cdot)$ denotes the Gumbel-Softmax function. $\mathbf{P} \in \{0,1\}^{\Delta\times N}$ is the one-hot represent of adversarial node set $\mathcal{P}$. Let $\mathbf{p}_k \in \{0,1\}^N$ denotes the k-th row in $\mathbf{P}$, the i-th element in $\mathbf{p}_k$ is calculated through Gumbel-Softmax by:

$$\mathbf{p}_k[i] = \frac{\exp((g_i + \log z_k[i])/t)}{\sum_j^N \exp((g_j + \log z_k[j])/t)}, \tag{11}$$

where g_i and is the Gumbel noise, t is the temperature coefficient, z_k is the k-th row in $\mathbf{Z}$. To evaluate gradients, the straight-through estimator is used in the Gumbel-Softmax trick. In the forward phase, the relaxed sample is rounded to get the one-hot vector. In the backward phase, actual gradients are directly passed to the relaxed sample instead of the one-hot vector.

Since labels of testing nodes are not available in training the target GCN model, adding or deleting edges between testing nodes has limited performance in poisoning attacks, which is discussed by Zhan and Pei (2022). To circumvent the generation of insignificant adversarial perturbations, the output of Gumbel-Softmax layer is restricted to nodes in the training set by masking the testing nodes.

4.3.2 Training phase of NP model

During the training phase of NP model, we initially randomly select Δ nodes from the node set $\mathcal{V}$ to form the adversarial node set $\mathcal{S}$ as the input. Based on the adjacency matrix A and feature matrix X along with the given adversarial node set $\mathcal{S}$, the NP model generates node set $\mathcal{P}$. Revisiting Eq. (4), the gradient of adversarial loss $L'(\mathcal{S}, \mathcal{P})$ relative to the parameters θ_{NP} of the NP model can be expressed as:

$$\nabla_{\theta_{NP}} L'(\mathcal{S}, \mathcal{P}) = \left(\frac{\partial L'(\mathcal{S}, \mathcal{P})}{\partial G'} + \frac{\partial L'(\mathcal{S}, \mathcal{P})}{\partial \theta^*} \cdot \frac{\partial \theta^*}{\partial G'} \right) \cdot \frac{\partial G'}{\partial \mathbf{P}} \cdot \frac{\partial \mathbf{P}}{\partial \theta_{NP}}, \tag{12}$$

where θ^* represents the parameters of the surrogate model trained using adversarial graph G'. Since the computation of gradients involves the training process of the surrogate model, higher-order derivatives should be calculated to capture the impact of the adversarial graph G' on the training process of the substitute model. Considering a surrogate model optimized using vanilla gradient descent, the optimization process can be expressed as:

$$\theta^{t+1} = \theta^t - \alpha \nabla_{\theta^t} L(\theta^t, \mathcal{V}_{train}, G'), \tag{13}$$

where α is the learning rate and $L(\theta^t, \mathcal{V}_{train}, G') = \sum_{v_j \in \mathcal{V}_{train}} \mathcal{L}(f_\theta(v_j, G'), y_j)$ is the training loss. After T iterates, the surrogate model is well trained with parameter $\theta^* = \theta^T$. In this scenario, the partial derivative $\frac{\partial \theta^{t+1}}{\partial G'}$ is given by:

$$\frac{\partial \theta^{t+1}}{\partial G'} = \frac{\partial \theta^t}{\partial G'} - \alpha \frac{\partial^2 L(\theta^t, \mathcal{V}_{train}, G')}{\partial G' \partial \theta^t}. \tag{14}$$

Given the one-hot forms of sets $\mathcal{S}$ and $\mathcal{P}$, denoted as $\mathbf{S} \in \{0,1\}^{\Delta \times N}$ and $\mathbf{P} \in \{0,1\}^{\Delta \times N}$, with the index of 1 indicating the corresponding position of the node in the adjacency matrix A. The adversarial matrix A' can be computed as follows:

$$A' = A + (J - I - 2A) \circ (\mathbf{S}^{\mathrm{T}} \mathbf{P} + \mathbf{P}^{\mathrm{T}} \mathbf{S}), \tag{15}$$

where $\circ$ denotes the element-wise product, $J = 1^{N \times N}$ corresponds to the all-ones matrix and I_N is the identity matrix. Since the proposed SPA method focuses on modifying the adjacency matrix, we have $\frac{\partial G'}{\partial \mathcal{P}} = \frac{\partial A'}{\partial \mathcal{P}}$. In summary, we have computed the derivative of the adversarial loss with respect to the parameters of the NP model, which can be used to optimize the parameters using an optimizer.

4.4 Iterative attack

As the perturbation rate increases, there may be a situation where perturbation number $\Delta > N$, the Node Searching method cannot obtain a subset $\mathcal{S}$ (with Δ nodes) from $\mathcal{V}$ (with N nodes). We therefore propose an iterative attack strategy which iteratively executing SPA method with a smaller perturbation rate (e.g., 5%). Algorithm 2 demonstrates the proposed

iterative attack method, where the SPA(G, Δ_b) denotes using SPA method on graph G to generate Δ_b adversarial edges. In the iterative attack, the adversarial graph generated in the previous iteration is considered the initial graph data for the next iteration. The SPA method is executed repetitively until the total perturbation number is Δ.

Algorithm 2 Iterative attack method

Require: graph G, perturbation number Δ, batch perturbation number Δ_b
Ensure: adversarial graph G'
while $\Delta > 0$ **do**
  $G' \leftarrow \text{SPA}(G, \Delta_b)$
  $G \leftarrow G'$
  $\Delta \leftarrow \Delta - \Delta_b$
  if $\Delta < \Delta_b$ **then**
    $\Delta_b = \Delta$
  end if
end while
return G'

5 Experiments

To evaluate and analyze the proposed SPA method, we implement our method and perform experimental comparisons based on DeepRobust (Li et al. (2021)) which is an adversarial library for attack and defense methods on images and graphs. The comparison methods and data preprocessing method (e.g., feature normalize) used in the experiment are all implemented based on DeepRobust.

5.1 Experimental setup

5.1.1 Dataset

We evaluate our SPA method on four datasets: Cora (McCallum et al., 2000), Cora-ML (Bojchevski & Günnemann, 2017), Citeseer (Sen et al., 2008) and Polblogs (Adamic & Glance 2005). The statistics of the four datasets are shown in Table 3. Following the preprocessing steps in DeepRobust, only the Largest Connected Component (LCC) from each

Table 3 Statistics of different datasets

Dataset	Nodes	Features	Edges	Classes
Cora Mc (Callum et al., 2000)	2708	1433	5249	7
Cora-ML (Bojchevski & Günnemann, 2017)	2995	2879	8416	7
Citeseer (Sen et al., 2008)	3312	3073	4715	6
Polblogs (Adamic & Glance, 2005)	1490	–	19025	2

graph dataset is considered and each dataset is randomly splitted into a training set (10%), a validation set (10%), and a testing set (80%).

5.1.2 Variants of SPA

In the SPA methodology, both the reward evaluation of the NS-ACO and the loss calculation of the NP model necessitate training the surrogate model on the adversarial graph G'. The frequent retraining has an adverse effect on the runtime efficiency of the SPA method. Specifically, during each iteration of the NS-ACO process, each ant generates an adversarial graph and subsequently trains a surrogate model to compute the reward, resulting in the surrogate model training process of $T_{NS} \times m$ times. In the NP model, the computation of gradients involves capturing the higher-order derivatives during the surrogate model training, requiring the surrogate model training process of T_{NP} times in total. The T_{NS} and T_{NP} denote the iteration times of NS-ACO and NP model, respectively. It is evident that the NS-ACO method is more affected by the training process of the substitute model.

To strike a balance between runtime efficiency and attack effectiveness, we introduce two variants of the SPA method: SPA-cl and SPA-re. Both methods utilize the same NP model but differ in their NS-ACO reward computation approaches.

In the SPA-cl variant, the NS-ACO initially trains a clean target model using the original dataset. For each adversarial graph generated during the algorithm execution, the clean model is directly employed to compute the training loss. Conversely, in the SPA-re variant, we follow the description provided in Eq. (4), where a victim model is trained using the adversarial graph, after which the training loss on the victim model is calculated.

5.1.3 Comparison methods

We compare the proposed SPA method with six state-of-the-art poisoning attack methods in graph data:

DICE (Deletes Internally and Connects Externally) (Waniek et al. (2018)) is a heuristic method that randomly decides to delete edges that connect nodes with the same class or connect nodes with different classes.

PGD (Xu et al. (2019)) is a white-box attack method that uses a Projected Gradient Descent algorithm to optimize the perturbations, and is designed for evasion attacks as the surrogate model remains unchanged, but can also be used in poisoning attacks.

Min-Max (Xu et al. (2019)) is the poisoning version of the PGD attack. It solves the bi-level optimization problem where the inner maximization is solved by gradient ascent, and the outer minimization is handled by PGD. Specifically, it retrains the surrogate model after getting an adversarial edge in each iteration to capture the influence of perturbations on the model training phase.

Mettack (Zügner and Günnemann (2019)) is a grey-box poisoning attack method where the adjacency matrix is modified iteratively based on the mate-gradient. Note that various variants of Mettack have been proposed in Zügner and Günnemann (2019), we use the most effective attack setting, i.e. Meta-Self attack, as the comparison method.

EpoAtk (Lin et al. (2023)) is a white-box attack method that generates adversarial edges through generation, evaluation and recombination with the goal of sidestepping the possible misinformation that the maximal gradient provides.

AtkSE (Liu et al. (2022)) is a grey-box attack method that uses edge discrete sampling, semantic invariance and momentum gradient ensemble to reduce the error in the adversarial edge generation process to improve the quality of the adversarial edges.

5.1.4 Target model

We use three different target models, i.e. GCN-linear, GCN, and GAT:

GCN-linear is a two-layer GCN model with linear activation function, and is the surrogate model used in the experiments.

GCN is one of the most representative GNNs. We use a two-layer GCN model with the ReLU activation function and a hidden size of 32.

GAT is an advanced neural network model which dynamically computes attention coefficients for different nodes, allowing the model to focus on the most relevant information. We use a two-layer GAT model with a hidden size of 8 and head number of 8.

Following the default setting in DeepRobust, the hidden size of GCN-linear models is 16, and the parameters of the target models are updated by Adam optimizer with a learning rate of 0.01 and a weight decay of 5×10^{-4}.

5.1.5 Evaluation

All experiment results are recorded as the average performance under 5 different splits according to the following steps:

- We first train a surrogate model with clean graph data. This model will be used as the target model for adversarial attack methods in subsequent experiments.
- We then generate adversarial edges and get adversarial adjacency matrix A'. Following the setting in Xu et al. (2019), the predicted labels of testing nodes can be used during the attack for all the adversarial attack methods.
- We train the GCN or GAT models with adversarial graph obtained by different attack methods and get multiple poisoned models.
- We finally verify the prediction accuracy on the testing set to evaluate the attack performance of the adversarial attack methods.

5.2 Attack performance

5.2.1 Attack performance on GCN

The attack results of different attack methods on the GCN model and the GCN-linear model by modifying 5% edges are shown in Tables 4, 5, respectively. The GCN-linear model and the GCN model is used to evaluate the white-box attack performance and the grey-box attack performance, respectively. The misclassification rate and the average rank are used as the evaluation metrics. Note that since the perturbation number is large in the PolBlogs dataset, we use the iterative attack to implement the attack.

We can see that the proposed SPA-re method and SPA-cl method achieve the best and the second best results among all comparison methods. AtkSE and Mettack achieve the highest attack performance among other comparison methods, followed by MinMax, PGD, EpoAtk, and DICE. When the target model is GCN-linear, our SPA-re method increases the misclassification rate over the AtkSE method on datasets Cora, Cora-ML, Citeseer, and

Table 4 Misclassification rate (MSR, %) and the average rank of different methods with perturbation rate 5% for poisoning attacks on GCN model. The iterative attack is applied on the PolBlogs dataset. (The best result in each column is in bold.)

Methods	GCN				
	Cora	Cora-ML	Citeseer	PolBlogs	Avg. rank
Clean	17.0 ± 0.3	15.5 ± 0.3	26.4 ± 0.5	7.0 ± 0.5	9
AtkSE (Liu et al., 2022)	22.8 ± 0.6	22.8 ± 0.5	33.3 ± 0.8	25.7 ± 1.2	3.75
EpoAtk (Lin et al., 2023)	19.5 ± 0.5	15.9 ± 0.4	32.7 ± 0.5	12.8 ± 1.0	6.75
Mettack (Zügner & Günnemann, 2019)	26.8 ± 0.4	19.4 ± 0.3	35.4 ± 0.3	16.5 ± 1.1	3.5
PGD (Xu et al., 2019)	20.7 ± 0.5	17.8 ± 0.4	29.2 ± 0.6	16.2 ± 0.7	5.75
MinMax (Xu et al., 2019)	24.1 ± 0.5	16.7 ± 0.3	29.9 ± 0.4	14.9 ± 0.5	5.5
DICE (Waniek et al., 2018)	17.9 ± 0.3	16.4 ± 0.1	27.2 ± 0.3	12.1 ± 0.7	7.75
SPA-cl	29.0 ± 0.8	26.0 ± 0.1	36.9 ± 0.7	**28.0 ± 1.1**	1.75
SPA-re	**32.5 ± 0.5**	**27.0 ± 0.4**	**37.4 ± 0.9**	26.7 ± 1.0	1.25

Table 5 Misclassification rate (MSR, %) and the average rank of different methods with perturbation rate 5% for poisoning attacks on GCN-linear model. The iterative attack is applied on the PolBlogs dataset. (The best result in each column is in bold.)

Methods	GCN-linear				
	Cora	Cora-ML	Citeseer	PolBlogs	Avg. Rank
Clean	19.0 ± 0.4	16.6 ± 0.2	27.2 ± 0.3	8.4 ± 0.2	9
AtkSE (Liu et al., 2022)	25.9 ± 0.7	23.7 ± 0.2	35.9 ± 0.6	25.2 ± 0.3	3.5
EpoAtk (Lin et al., 2023)	22.7 ± 0.5	20.2 ± 0.4	35.8 ± 0.7	17.1 ± 0.3	5.25
Mettack (Zügner & Günnemann, 2019)	27.4 ± 0.5	21.4 ± 0.2	35.4 ± 0.5	16.0 ± 0.6	4.5
PGD (Xu et al., 2019)	23.1 ± 0.5	20.8 ± 0.2	31.6 ± 0.4	16.6 ± 0.2	5.5
MinMax (Xu et al., 2019)	26.9 ± 0.3	18.9 ± 0.3	33.1 ± 0.5	15.2 ± 0.2	6.5
DICE (Waniek et al., 2018)	19.5 ± 0.3	18.2 ± 0.3	28.5 ± 0.5	11.9 ± 0.2	7.5
SPA-cl	31.3 ± 0.8	31.7 ± 0.3	**40.1 ± 0.7**	31.6 ± 0.6	1.75
SPA-re	**35.0 ± 0.8**	**31.8 ± 0.2**	39.0 ± 0.4	**32.5 ± 0.6**	1.25

PolBlogs by 9.1%, 8.1%, 3.1%, and 7.3%, respectively. The results of white-box experiments show that the proposed SPA methods are able to capture the effect of adversarial perturbations on the model retraining process and find a better combination of perturbations. When the target model is GCN, the SPA-re method can increase the misclassification rate over the Mettack method on the four datasets by 5.7%, 7.6%, 2.0%, and 11.5%, respectively. The grey-box results show that the proposed SPA methods can achieve effective transferring attack when the attacker does not know the hyperparameters of the target model.

We also find that despite the fact that the SPA-cl relaxes the restriction on the loss values in the NS step by directly evaluating the training loss using the clean model without retraining, the SPA-cl can still achieve similar misclassification rates as the SPA-re. This may be due to the fact that the SPA-cl causes a large bias in the model parameters by choosing perturbations that maximize the training loss, thus affecting the prediction accuracy on the testing set.

Among these comparison methods, we note that EpoAtk does not achieve the results shown by Lin et al. (2023). This may be due to the fact that we uniformly used the GCN model from the DeepRobust framework (Li et al. (2021)) as the target model in the experiments. For a fair comparison, we executed the EpoAtk attack using the GCN model provided by Lin et al. as the target model with the same hyperparameter settings and dataset partitioning shown in Sect. 5.1. The misclassification rates achieved by the EpoAtk attack on the four datasets at a perturbation rate of 5% are Cora 25.08, Cora-ML 22.09, Citeseer 33.84, and Polblogs 13.91.

5.2.2 Attack performance on GAT

In this section, we present experimental results on Graph Attention Network (GAT) model for both transfer attacks and white-box attacks.

Table 6 presents the experimental results under the transfer attack setting with a perturbation rate of 5%. In the transfer attack scenario, a GCN-linear model is utilized as the surrogate model to generate adversarial graphs. Then we perform attacks on the GAT model. Compared with the attack results on the GCN model in Table 4, it is observed that while the proposed SPA-cl method and SPA-re method still ranks highly, they no longer exhibit significant advantages. This is attributed to the fact that in adversarial attacks, when there is a notable discrepancy between the proxy and the target models, the adversarial perturbations may not poison the target model as intended.

Among the attack methods, EpoAtk, PGD, and Min-Max use first-order gradients in optimizing adversarial perturbations. In contrast, AtkSE, Mettack, SPA-cl, and SPA-re utilize second-order gradients. The experimental outcomes depicted in Table 6 suggest that attack methods using second-order gradient possess a better attack capability for transfer attacks. This is because the second-order gradients capture the influence of adversarial perturbations on the model training process, thereby enhancing the attack performance of the adversarial perturbations.

Table 7 illustrates the experimental results under the white-box attack setting with a perturbation rate of 5%. In the white-box attack, we utilize the GAT-linear model, a variant of the GAT model with linear activation layers, as the surrogate model for generating adversarial perturbations. Then the adversarial perturbations are assessed on the GAT model. Due to the complex nature of the GAT model, the process of computing second-order gradients incurs considerable space complexity. This has been identified as a limiting factor for the Mettack and AtkSE methods, preventing them from completing white-box attacks on the GAT model within the constraints of a 40GB graphics memory limit. As a result, our comparative analysis focuses solely on first-order optimization methods, namely EpoAtk, PGD, and Min-Max. Similar to the Mettack method, the proposed SPA-re method uses second-order gradients to optimize the model parameters during the NP model training process. However, as is shown in Eq. (12), the NP model computes second-order gradients on the matrix $\mathbf{P} \in \{0,1\}^{\Delta \times N_{train}}$. In contrast to directly calculating second-order gradients on the adjacency matrix $A \in \{0,1\}^{N \times N}$, the NP model exhibits a lower spatial complexity. The reduced complexity enables our SPA-re methods to perform white-box adversarial attacks on the GAT model. Table 7 illustrates that our SPA-re method achieves the most effective attack performance on the GAT model.

Comparing Table 7 with Table 6, we observe that attack methods based on first-order optimization perform even worse in white-box attacks than in transfer attacks. This phenomenon can be attributed to the GAT model's mechanism of calculating attention

coefficients between nodes to determine the feature aggregation process. By altering these attention coefficients, the GAT model can mitigate the impact of adversarial attacks. In contrast, our SPA-re method takes into account the model's training process, including the computation of attention coefficients, and is capable of generating adversarial perturbations that significantly disrupt the model.

5.2.3 Attack performance with different perturbation rates

Figure 3 shows the attack performance on the GCN model by increasing the perturbation rate from 1% to 10% with a step size 2%. It is noticeable that the misclassification rate of all methods generally increases with the elevation of the perturbation rate, and at all perturbation rates, the SPA-re method and SPA-cl method obtain the best and second best attack performance respectively. Since with the increase of the perturbation rate, the perturbation

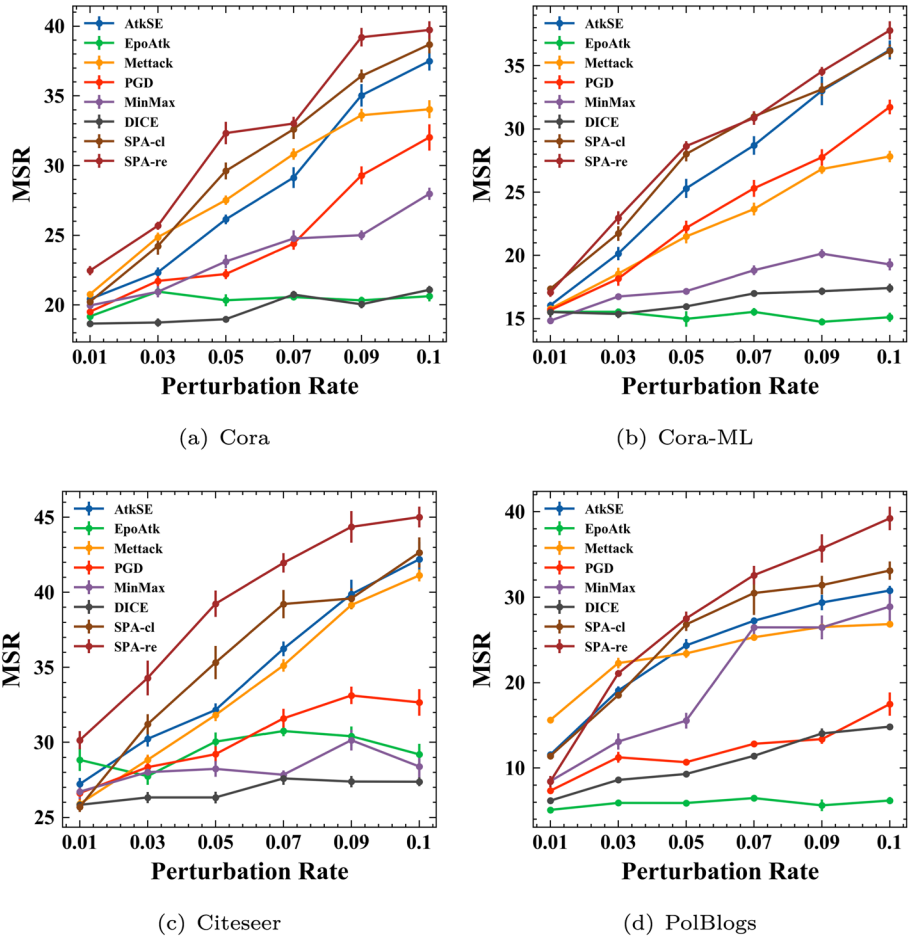


Fig. 3 The misclassification rate (MSR, %) of different methods with perturbation rate within [1%, 10%] evaluated on the GCN model with grey-box attacking setting. These error bars represent the standard deviation of the misclassfication rate

Table 6 Misclassification rate (MSR, %) and the average rank of different methods with perturbation rate 5% for transfer attacks. The GCN-linear model is used as the surrogate and the GAT model acts as the victim model. The iterative attack is applied on the PolBlogs dataset. (The best result in each column is in bold.)

Methods	GAT(transfer attacks)				
	Cora	Cora-ML	Citeseer	PolBlogs	Avg. Rank
Clean	17.8 ± 0.5	16.1 ± 0.5	26.8 ± 0.4	6.0 ± 0.6	8.75
AtkSE (Liu et al., 2022)	19.5 ± 1.0	**23.1 ± 0.4**	30.4 ± 0.7	19.8 ± 1.4	3.125
EpoAtk (Lin et al., 2023)	18.4 ± 0.4	15.9 ± 0.4	28.6 ± 1.3	8.2 ± 2.8	7.75
Mettack (Zügner & Günnemann, 2019)	22.9 ± 0.7	21.7 ± 0.7	**33.1 ± 0.6**	18.6 ± 2.3	2.25
PGD (Xu et al., 2019)	18.3 ± 0.7	18.1 ± 0.6	30.1 ± 0.6	16.7 ± 0.8	5.75
MinMax (Xu et al., 2019)	19.5 ± 0.6	17.3 ± 0.5	29.2 ± 0.5	11.1 ± 1.3	5.875
DICE (Waniek et al., 2018)	18.8 ± 0.5	17.3 ± 0.4	27.9 ± 0.6	13.0 ± 2.9	6.75
SPA-cl	21.2 ± 0.9	20.4 ± 1.1	31.8 ± 0.8	**30.1 ± 5.1**	2.5
SPA-re	**23.7 ± 0.6**	19.6 ± 0.5	32.0 ± 0.9	29.1 ± 3.0	2.25

Table 7 Misclassification rate (MSR, %) and the average rank of different methods with perturbation rate 5% for poisoning attacks on GAT model. The iterative attack is applied on the PolBlogs dataset. (The best result in each column is in bold.)

Methods	GAT(white-box attacks)				
	Cora	Cora-ML	Citeseer	PolBlogs	Avg. Rank
Clean	17.6 ± 0.4	16.0 ± 0.3	26.8 ± 0.4	6.3 ± 0.4	4.75
EpoAtk (Lin et al., 2023)	17.0 ± 0.5	16.0 ± 0.4	26.9 ± 0.6	6.6 ± 1.3	4.25
PGD (Xu et al., 2019)	18.3 ± 0.3	16.8 ± 0.4	28.2 ± 0.6	8.9 ± 1.0	2.75
MinMax (Xu et al., 2019)	18.7 ± 0.5	16.6 ± 0.4	28.6 ± 0.7	9.6 ± 1.2	2.25
SPA-cl	25.1 ± 0.7	23.5 ± 0.7	28.6 ± 0.7	9.6 ± 1.2	2.25
SPA-re	**27.1 ± 0.7**	**24.8 ± 0.8**	**37.3 ± 0.9**	**20.5 ± 4.5**	1

number may be larger than the node number, we use the iterative version to implement SPA-cl and SPA-re to obtain adversarial attacks. Note the misclassification rate does not increase linearly with the perturbation rate, this is caused by the uncertainty of the perturbation generation process and the model training process.

5.3 Ablation study

To verify the effectiveness of different components in SPA, we conduct ablation studies to analyze the influence of each component and parameter.

5.3.1 Contribution of NS and NP

In this experiment, we remove the NS module, the NP module, and the retraining strategy respectively from the proposed SPA method to verify the contribution of each module to the adversarial perturbations. For the NS and NP modules, we replace them with random

Table 8 Misclassification rate (MSR, %) of SPA method with perturbation rate 0.05. The components in the SPA are replaced with a random selection algorithm, respectively. The component re means retraining the surrogate model in the NS step

Components of SPA			Dataset			
NS	NP	re	Cora	Cora_ML	Citeseer	PolBlogs
✓	✓	✓	31.8 ± 0.7	27.1 ± 0.5	37.4 ± 0.8	25.6 ± 1.3
✓	✓	✗	28.8 ± 0.8	25.8 ± 0.4	36.9 ± 0.6	28.2 ± 1.1
✓	✗	✓	17.6 ± 0.4	17.0 ± 0.3	28.3 ± 0.4	10.9 ± 0.9
✓	✗	✗	18.4 ± 0.5	16.1 ± 0.4	28.0 ± 0.4	10.4 ± 0.8
✗	✓	✗	24.9 ± 0.8	21.5 ± 0.3	32.3 ± 0.6	15.9 ± 0.6
✗	✗	✗	17.5 ± 0.4	16.3 ± 0.3	26.8 ± 0.5	10.9 ± 1.0

Table 9 Misclassification rate (MSR, %) on Cora dataset after limiting search scope to $\mathcal{V}$, $\mathcal{V}_{train}$, and $\mathcal{V}_{unlabled}$, separately

	NP-all	NP-train	NP-unlabeled
NS-all	25.3 ± 1.0	28.5 ± 0.5	18.1 ± 0.3
NS-train	18.5 ± 0.4	19.7 ± 0.5	18.7 ± 0.3
NS-unlabeled	25.2 ± 0.6	28.4 ± 0.7	17.8 ± 0.3

selection algorithms, i.e., select Δ nodes randomly from the node set $\mathcal{V}$ to form the adversarial node set $\mathcal{S}$ or $\mathcal{P}$. For the retraining strategy, we remove it directly since the retraining strategy means retraining the surrogate model in the NS step. Note the retraining strategy will not be adopted if the NS module is not used.

Table 8 shows the misclassification rate with 5% edges modified in the four graph datasets. It is clear from the results that the NP module plays an important role in the proposed SPA method. This may be due to the fact that the choice of perturbations is not unique when implementing the adversarial attacks, and even for a randomly selected $\mathcal{S}$, the NP model is able to generate a corresponding subset of nodes with high attack capability. Despite the lower contribution of the NS module in the attack, it is difficult further to increase the misclassification rate within a limited perturbation budget, so we believe it is valuable to spend more time searching the node set $\mathcal{S}$ using a heuristic algorithm rather than simply using random selection.

5.3.2 Influence of search scope

To verify the influence of search scope on the SPA, we individually restrict the search scope for the NP and NS modules to $\mathcal{V}$, $\mathcal{V}_{train}$, and $\mathcal{V}_{unlabled}$, separately. $\mathcal{V}_{unlabled}$ means nodes without ground-truth label, is typically composed of $\mathcal{V}_{test}$ and $\mathcal{V}_{val}$. According to the division of the dataset, the training set contains 10% of nodes while the remaining 90% are unlabeled nodes. The experimental results of the proposed SPA on the Cora dataset are presented in Table 9. Note that when limiting the search scope of the NS algorithm to $\mathcal{V}_{train}$, we perform iterative attack since the number of nodes in the training set is smaller than the perturbation number.

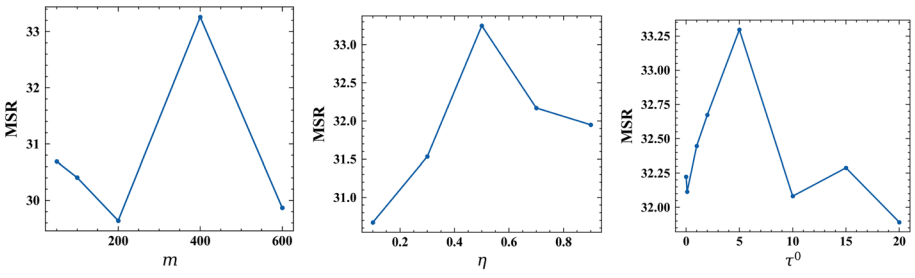
It can be seen that the best attack effect appears in the combination of NS-all and NP-train, which are determined by the characteristics of NS and NP. The NS step needs to find more combinations of adversarial nodes in a wider search scope, while the NP step needs to narrow the search scope to speed up the model convergence. Therefore, when NS

is restricted to $\mathcal{V}_{train}$ or NP is restricted to $\mathcal{V}_{unlabeled}$, the attack is difficult to achieve effect. In addition, the combination of NS-unlabeled and NP-unlabeled achieves the worst attack effect, proving that the nodes in $\mathcal{V}_{unlabeled}$ have little impact on the model training process, making poisoning attack difficult to implement. By implementing the restriction on NP model to $\mathcal{V}_{train}$, we can eliminate the adversarial edges where both nodes are unlabeled nodes. Following Zhan et al. Zhan and Pei (2022), we have conducted a statistical analysis to the proportion of adversarial edges composed of two testing nodes, generated by various comparative methods under a 5% perturbation rate. The experimental results indicate that none of the adversarial edges generated by EpoAtk, AtkSE, MinMax, and Mettack are composed of two testing nodes. In contrast, the PGD method yielded 5.2% adversarial edges composed of two testing nodes. Moreover, the Random method exhibited a significantly higher proportion with 81.4%, followed closely by DICE at 81.0%. This demonstrates that eliminating adversarial edges where both nodes are in the testing set does not cause difference when comparing with SOTA attack methods, and the comparative analysis remains fair.

5.3.3 Influence of hyperparameters in NS-ACO

To elucidate the impact of hyperparameters on the performance of our Node Searching Ant Colony Optimization (NS-ACO) algorithm, we conducted an ablation study focusing on the number of ants m, the evaporation rate of pheromones η, and the initial pheromones τ^0. The experimental results of Cora dataset on GCN model are shown in Fig. 4. To preclude interference from Node Pairing setp, we have employed the same NP model throughout the entirety of our experimental process. We systematically varied each hyperparameter while maintaining others at their default values, as determined from preliminary experiments. The number of ants are adjusted to evaluate the algorithm's behavior with different population sizes, ranging from 50 to 600 ants. The evaporation rate is modified to assess the balance between exploration and exploitation, with values spanning from 0.1 to 0.9. The initial pheromone is tuned to determine the exploratory breadth of the algorithm's early stages, with settings from 0.01 to 20. For each configuration, the ACO algorithm was executed over 10 independent runs to ensure the statistical robustness of the results.

The performance is evaluated based on the misclassification rate of the attack. The ablation study revealed that the choice of the number of ants m and the evaporation rate η significantly influences the convergence speed and solution quality. With the increase



(a) Ant Number (b) Evaporation Rate (c) Initial Pheromones

Fig. 4 The misclassification rate (MSR, %) of SPA-re attacks on GCN model with different hyperparameter settings

of the number of ants m, the misclassification rate tends to first decrease, then increase, and finally decrease again. This pattern is determined by the pheromone mechanism. Since there are "lucky" nodes that may be selected multiple times even they do not have significant adversarial capabilities, making them have higher pheromone values. As the number of ants increases, this type of nodes will appear more frequently, consequently diminishing the attack performance. However, a higher number of ants allows the algorithm to explore a broader solution space, enabling the discovery of more nodes with high attacking capabilities, thereby increase the attack performance. The interplay between these two types of nodes results in the MSR curve in Fig.4 (a). Therefore, more attempts should be made to select the ant number in order to reduce the number of "lucky" nodes. As the evaporation rate η increases, the misclassification rate shows a trend of first increasing and then decreasing, proving that a moderate evaporation rate can significantly improve the performance of the algorithm. In contrast, the misclassification rate changed from 31.8 to 33.3 with the variation of the initial pheromones τ^0. This indicates that the influence of the initial pheromone on the model's performance is relatively minor. Finally, we find an optimal balance at $m = 400$, $\eta = 0.5$, and $\tau^0 = 20$.

5.4 Convergence analysis

In this section, we analyze the convergence of the NP model and the NS-ACO algorithm with different perturbation rates.

5.4.1 Convergence analysis of NP model

Figure 5 shows the training loss curve of the NP model on different datasets at perturbation rates of 1%, 5%, and 10%, respectively. As shown in the figures, when the perturbation rate is 5%, the training loss decreases reasonably and maintains a low value. This indicates that the NP model can effectively extract node features and generate corresponding nodes, which also explains the high misclassification rate that the NP-random attack achieved in Table 8. Then we reduce the perturbation rate to 1% and analyze the loss value during model training. As shown in the figure, the training loss is difficult to converge to a stable value at low perturbation rates. Recall the loss function of the NP model is the negative training loss of the surrogate model (in Eq. (4)). Influenced by the randomness of the surrogate model parameter updates, the loss value $L'(\mathcal{S}, \mathcal{P})$ contains the random noise introduced in the training process. When the perturbation rate is high (e.g., 5%), the loss value is mainly determined by the quality of the perturbations, while at low perturbation rates (e.g., 1%), the random noise has a large impact on the back-propagation process of the NP model, and the parameters of NP model are difficult to be optimized accurately.

We further increase the perturbation rate to 10%. As shown in the figure, the NP model is able to reduce the loss value almost linearly as the perturbation increases. For example, for the Cora dataset at 5% perturbation rate, the NP model converges at a loss value of -0.5 (0.4 reductions from initial), and when the perturbation rate increases to 10%, the model converges at a loss value of -0.7 (0.6 reductions from initial). For the PolBlogs dataset, the perturbation number is larger than the node number at 10% perturbation rate, and according to Sect. 4.4, we do not generate all perturbations at the same time, but implement the attack iteratively and therefore do not analyze its convergence.

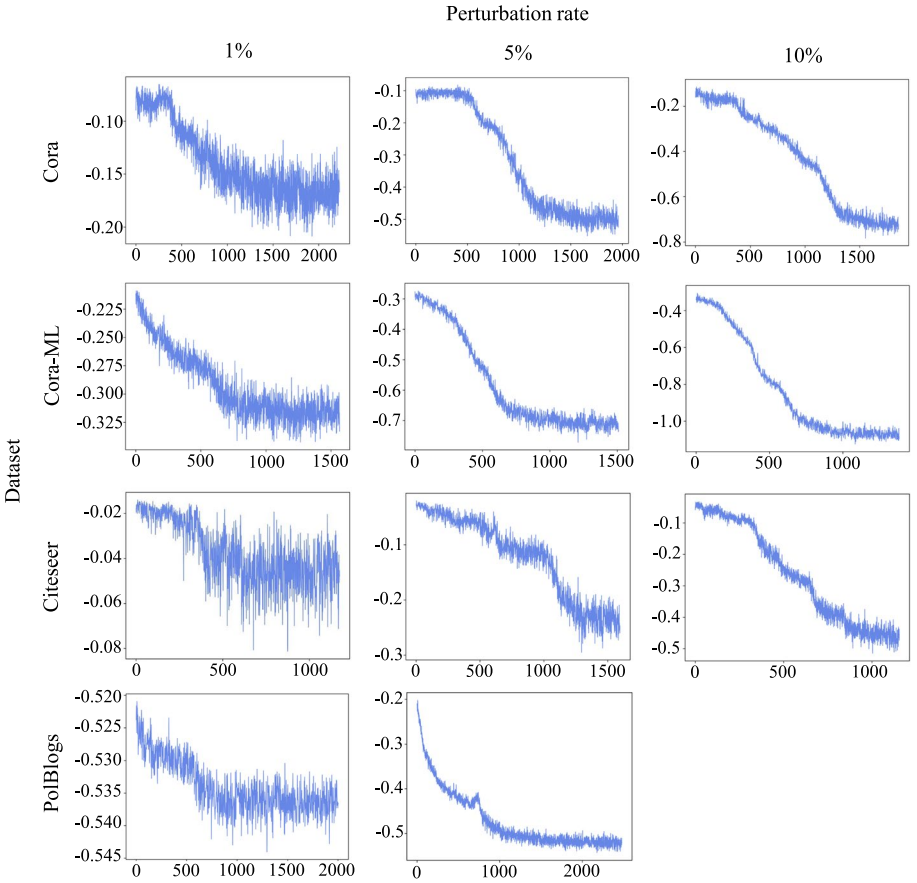


Fig. 5 Training loss curve of the NP model on Cora, Cora-ML, Citerseer, and PolBlogs with perturbation rate of 1%, 5%, and 10%. The horizontal coordinate indicates the number of iterations and the vertical coordinate indicates the loss value

5.4.2 Convergence analysis of NS-ACO algorithm

We record the changes in the average scores, best scores, and top-k scores during the NS-AOC algorithm, as is shown in Figure. 6, where the top-k score indicates the score of the node set which contains Δ nodes with the highest pheromone. As the number of iterations increases, the scores keep increasing, indicating that nodes with higher attack capabilities are given higher pheromones and thus have a higher probability to be selected by the ants. Amount all the results, the top-k score stays the highest, so we use the top Δ nodes in the pheromone table as the adversarial node set $\mathcal{S}$ selected at the end of the algorithm. We also find that the NS-ACO algorithm has difficulty in improving the quality of the perturbation as the perturbation rate increases. For example, the curve is almost horizontal when the perturbation rate is 10%. Recall that the NS algorithm selects a subset of the node set $\mathcal{V}$ as the adversarial node set $\mathcal{S}$. When $\mathcal{S}$ becomes larger, it contains most of the nodes with high attack capability, making the gap between ants smaller and the difference in pheromones left at each node also become smaller, which results in difficulty in selecting node set with

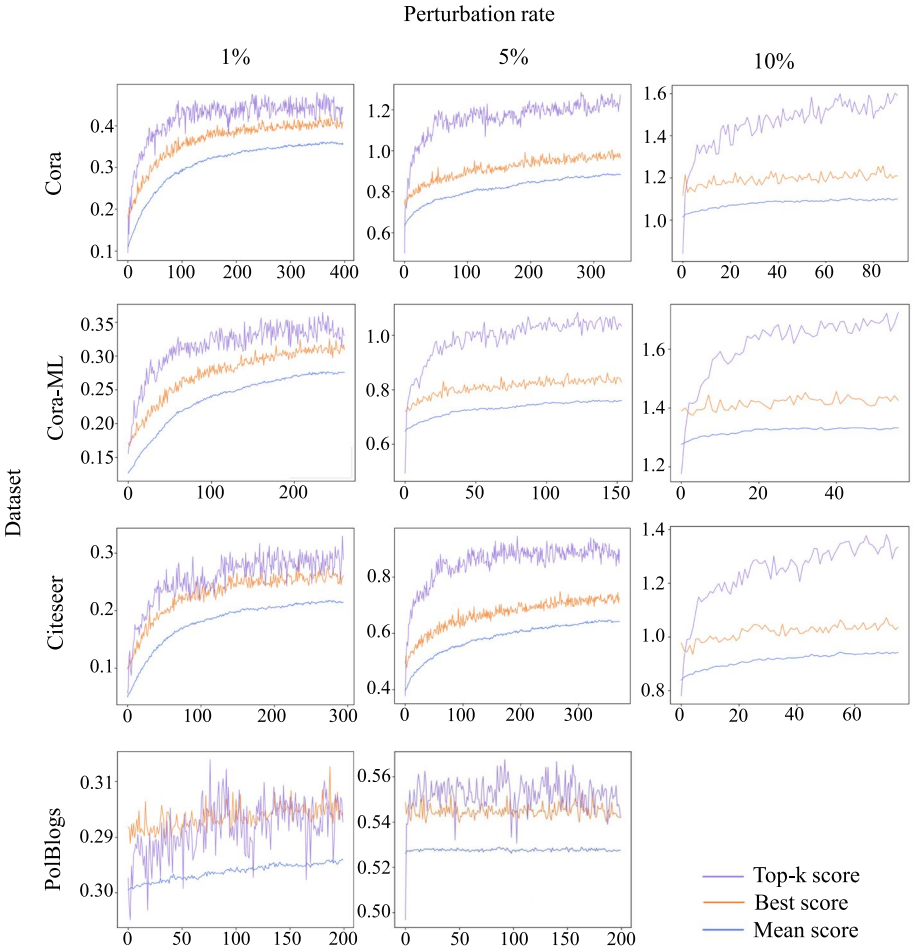


Fig. 6 Scores of ants in NS-ACO algorithm on Cora, Cora-ML, Citerseer, and PolBlogs with perturbation rate 1%, 5%, and 10%. The horizontal coordinate indicates the number of iterations and the vertical coordinate indicates the loss value

high attack performance. This proves the necessity of using a smaller perturbation rate to execute the iterative attack.

6 Conclusion

In this paper, we proposed a novel Searching and Pairing attack (SPA), which converts the hard adversarial edge search problem into two simpler Node Searching and Node Pairing problems to avoid optimizing adversarial edges using gradient directly. The SPA method considers an adversarial edge as a combination of two adversarial nodes, and generates adversarial nodes by using a proposed Node Searching Ant Colony algorithm (NS-ACO) method and a generative model based Node Pairing (NP) method. In the NS-ACO method, by searching different combinations of nodes, the NS method

aims to obtain the set $\mathcal{S}$ with better attack effect. In the NP method, by extracting and aggregating the features of the nodes in $\mathcal{S}$, the model generates node set $\mathcal{P}$ to form the adversarial edge set $\mathcal{E}'$ together with $\mathcal{S}$. The SPA method then modifies the adjacency A according to $\mathcal{E}'$ and gets the adversarial graph G'. Extensive experiments on four graph datasets show that the proposed SPA method can achieve better attack performance than the state-of-the-art methods.

In the future, it is interesting to explore the integration of the SPA method with adversarial attack methods on large-scale graphs. By employing techniques such as sampling or graph coarsening to simplify large-scale graphs, the SPA method can then be applied on the simplified graphs, thereby extending the SPA method to large-scale graph applications.

Funding This work is supported by the NUDT Innovation Science Foundation (23-ZZCX-JDZ-08), the NUDT Research Project (ZK22-56) and the National Natural Science Foundation of China under Grant Nos. 62201600

Declarations

Conflict of interest The authors declare no Conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

Adamic, L.A., Glance, N. (2005). The political blogosphere and the 2004 us election: Divided they blog. In: Proceedings of the 3rd International Workshop on Link Discovery, pp. 36–43.

Baltrunas, L., Church, K., Karatzoglou, A., Oliver, N. (2015). Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild. arXiv preprint arXiv:1505.03014.

Bojchevski, A., Günnemann, S. (2017). Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. arXiv preprint arXiv:1707.03815.

Cai, L., Ji, S. (2020). A multi-scale approach for graph link prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 3308–3315.

Chang, H., Rong, Y., Xu, T., Huang, W., Zhang, H., Cui, P., Wang, X., Zhu, W., & Huang, J. (2022). Adversarial attack framework on graph embedding models with limited knowledge. *IEEE Transactions on Knowledge and Data Engineering, 35*, 4499.

Chen, Z., Wang, Z., Huang, J., Zhao, W., Liu, X., Guan, D. (2022). Imperceptible adversarial attack via invertible neural networks. arXiv preprint arXiv:2211.15030.

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

Chen, J., Chen, L., Chen, Y., Zhao, M., Yu, S., Xuan, Q., & Yang, X. (2019). Ga-based q-attack on community detection. *IEEE Transactions on Computational Social Systems, 6*(3), 491–503.

Collins, C.M., Sayeed, H., Darling, G., Claridge, J.B., Sparks, T.D., Rosseinsky, M.J. (2024). Integration of generative machine learning with the heuristic crystal structure prediction code fuse. Faraday Discussions.
Colorni, A., Dorigo, M., Maniezzo, V., et al. (1991). Distributed optimization by ant colonies. In: Proceedings of the First European Conference on Artificial Life, vol. 142, pp. 134–142. Paris, France.
Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., Song, L. (2018). Adversarial attack on graph structured data. In: International Conference on Machine Learning, pp. 1115–1124. PMLR.
Dai, J., Zhu, W., & Luo, X. (2022). A targeted universal attack on graph convolutional network by using fake nodes. *Neural Processing Letters, 54*(4), 3321–3337.
Du, X., Yu, J., Yi, Z., Li, S., Ma, J., Tan, Y., & Wu, Q. (2020). A hybrid adversarial attack for different application scenarios. *Applied Sciences-Basel*. https://doi.org/10.3390/app10103559
Goodfellow, I.J., Shlens, J., Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.
Gosch, L., Geisler, S., Sturm, D., Charpentier, B., Zügner, D., Günnemann, S. (2024). Adversarial training for graph neural networks: Pitfalls, solutions, and new directions. Advances in Neural Information Processing Systems **36**.
Haoxi, Z., Xiaobing, P. (2021). Fha: Fast heuristic attack against graph convolutional networks. In: International Conference on Discovery Science, pp. 151–165. Springer.
Harper, F. M., & Konstan, J. A. (2015). The movielens datasets: History and context. *Acm Transactions on Interactive Intelligent Systems (tiis), 5*(4), 1–19.
Kipf, T.N., Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.
Li, Y., Jin, W., Xu, H., Tang, J. (2021). Deeprobust: A pytorch library for adversarial attacks and defenses. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 16078–16080.
Li, Z., Wang, J., Meng, M.Q.-H. (2021). Efficient heuristic generation for robot path planning with recurrent generative model. In: 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 7386–7392. IEEE.
Lin, X., Zhou, C., Wu, J., Yang, H., Wang, H., Cao, Y., & Wang, B. (2023). Exploratory adversarial attacks on graph neural networks for semi-supervised node classification. *Pattern Recognition, 133*, 109042.
Liu, X., Huang, J.-J., Zhao, W. (2024). Gcia: A black-box graph injection attack method via graph contrastive learning. In: ICASSP 2024–2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6570–6574. 10.1109/ICASSP48485.2024.10446876.
Liu, Z., Luo, Y., Wu, L., Li, S., Liu, Z., Li, S.Z. (2022). Are gradients on graph structure reliable in gray-box attacks? In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, pp. 1360–1368.
Liu, Z., Luo, Y., Wu, L., Liu, Z., Li, S.Z. (2023). Towards reasonable budget allocation in untargeted graph structure attacks via gradient debias. arXiv preprint arXiv:2304.00010.
Liu, X., Si, S., Zhu, X., Li, Y., Hsieh, C.-J. (2019). A unified framework for data poisoning attack to graph-based semi-supervised learning. arXiv preprint arXiv:1910.14147.
Liu, H., Ge, Z., Zhou, Z., Shang, F., Liu, Y., & Jiao, L. (2023). Gradient correction for white-box adversarial attacks. *IEEE Transactions on Neural Networks and Learning Systems*. https://doi.org/10.1109/TNNLS.2023.3315414
Luo, C., Lin, Q., Xie, W., Wu, B., Xie, J., Shen, L. (2022). Frequency-driven imperceptible adversarial attack on semantic similarity. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15315–15324.
McCallum, A. K., Nigam, K., Rennie, J., & Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval, 3*(2), 127–163.
Meng, Z., Liang, S., Bao, H., Zhang, X. (2019). Co-embedding attributed networks. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pp. 393–401. ACM.
Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., Frossard, P. (2017). Universal adversarial perturbations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1765–1773.
Moosavi-Dezfooli, S.-M., Fawzi, A., Frossard, P. (2016). Deepfool: A simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2574–2582.
Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A. (2016). The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy (EuroS &P), pp. 372–387. IEEE.
Rossetti, G., Pappalardo, L., Pedreschi, D., & Giannotti, F. (2017). Tiles: An online algorithm for community discovery in dynamic social networks. *Machine Learning, 106*, 1213–1241.

Rossi, A., Barbosa, D., Firmani, D., Matinata, A., & Merialdo, P. (2021). Knowledge graph embedding for link prediction: A comparative analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD), 15*(2), 1–49.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine, 29*(3), 93–93.

Sundararajan, M., Taly, A., Yan, Q. (2017).Axiomatic attribution for deep networks. In: International Conference on Machine Learning, pp. 3319–3328. PMLR.

Su, J., Vargas, D. V., & Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation, 23*(5), 828–841.

Wang, Z., Hao, Z., Wang, Z., Su, H., Zhu, J. (2021). Cluster attack: Query-based adversarial attacks on graphs with graph-dependent priors. arXiv e-prints, 2109 .

Wang, B., Li, Y., Zhou, P. (2022). Bandits for structure perturbation-based black-box attacks to graph neural networks with theoretical guarantees. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13379–13387.

Wang, Q., Hao, Y., & Zhang, J. (2023). Generative inverse reinforcement learning for learning 2-opt heuristics without extrinsic rewards in routing problems. *Journal of King Saud University - Computer and Information Sciences, 35*(9), 101787. https://doi.org/10.1016/j.jksuci.2023.101787

Waniek, M., Michalak, T. P., Wooldridge, M. J., & Rahwan, T. (2018). Hiding individuals and communities in a social network. *Nature Human Behaviour, 2*(2), 139–147.

Wu, J., Li, S., Li, J., Pan, Y., Xu, K. (2022) A simple yet effective method for graph classification. arXiv preprint arXiv:2206.02404.

Wu, Y., Lian, D., Xu, Y., Wu, L., Chen, E. (2020). Graph convolutional networks with markov random field reasoning for social spammer detection. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 1054–1061.

Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., Zhu, L. (2019). Adversarial examples on graph data: Deep insights into attack and defense. arXiv preprint arXiv:1903.01610.

Xie, T., Wang, B., Kuo, C.-C.J. (2022). Graphhop: An enhanced label propagation method for node classification. IEEE Transactions on Neural Networks and Learning Systems.

Xie, Y., Yao, C., Gong, M., Chen, C., & Qin, A. K. (2020). Graph convolutional networks with multi-level coarsening for graph classification. *Knowledge-Based Systems, 194*, 105578.

Xu, K., Chen, H., Liu, S., Chen, P.-Y., Weng, T.-W., Hong, M., Lin, X. (2019). Topology attack and defense for graph neural networks: an optimization perspective. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp. 3961–3967.

Yang, R., & Long, T. (2021). Derivative-free optimization adversarial attacks for graph convolutional networks. *PeerJ Computer Science, 7*, 693.

Yi, Z., Li, S., Ma, J., Yu, J., Tan, Y., & Wu, Q. (2022). Towards an efficient and robust adversarial attack against neural text classifier. *International Journal of Pattern Recognition and Artificial Intelligence*. https://doi.org/10.1142/S021800142253007X

Yu, S., Zheng, J., Chen, J., Xuan, Q., Zhang, Q. (2020). Unsupervised euclidean distance attack on network embedding. In: 2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC), pp. 71–77. IEEE.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research, 33*(4), 452–473.

Zhan, H., Pei, X. (2021). Black-box gradient attack on graph neural networks: Deeper insights in graph-based attack and defense. arXiv preprint arXiv:2104.15061.

Zhang, T., Wang, J., & Meng, M.Q.-H. (2021). Generative adversarial network based heuristics for sampling-based path planning. *IEEE/CAA Journal of Automatica Sinica, 9*(1), 64–74.

Zhan, H., & Pei, X. (2022). Dealing with the unevenness: Deeper insights in graph-based attack and defense. *Machine Learning, 113*, 1–33.

Zhao, J., Wang, X., Shi, C., Hu, B., Song, G., Ye, Y. (2021). Heterogeneous graph structure learning for graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 4697–4705.

Zhou, F., Cao, C., Zhang, K., Trajcevski, G., Zhong, T., Geng, J. (2019). Meta-gnn: On few-shot node classification in graph meta-learning. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 2357–2360.

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

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

Zügner, D., Borchert, O., Akbarnejad, A., & Günnemann, S. (2020). Adversarial attacks on graph neural networks: Perturbations and their patterns. *ACM Transactions on Knowledge Discovery from Data (TKDD), 14*(5), 1–31.