TRUSTWORTHY GRAPH LEARNING: ADVERSARIAL ATTACK, EMPIRICAL DEFENSE, AND CERTIFIED ROBUSTNESS

YUNI LAI

PhD

The Hong Kong Polytechnic University

2025

The Hong Kong Polytechnic University
Department of Computing

Trustworthy Graph Learning: Adversarial Attack, Empirical Defense, and Certified Robustness

Yuni LAI

A thesis submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
Nov 2024

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: Yuni LAI

Abstract

With the prevalence of graph data on the Internet, graph learning models, particularly Graph Neural Networks (GNNs), have emerged as powerful tools for various applications, including social network analysis, recommender systems, and anomaly detection. These applications often play critical roles in ensuring system security and reliability, making the trustworthiness of graph models paramount. However, recent studies have revealed the vulnerability of graph learning models to adversarial attacks, highlighting the urgent need to enhance their robustness. This thesis delves into the vulnerability and robustness of graph learning models from three key perspectives: adversarial attacks, empirical defenses, and certifiable defenses.

First, we introduce **coupled-space attack** to investigate the vulnerability of the random-walk-based anomaly detection (RWAD) model. A unique characteristic of RWAD is that it can operate on both pre-existing and feature-derived graphs, which present two potential attack surfaces: graph-space and feature-space. Our proposed coupled-space attacks are the first to investigate the interplay between graph-space and feature-space attacks. We prove the NP-hardness of attacking RWAD and propose efficient strategies to solve the bi-level optimization problem associated with the attacks.

Secondly, we propose a powerful **MetaC attack** for both GNN-based and matrix-factorization-based recommender systems. Leveraging insights from our vulnerability analysis, we design a robust recommender system with empirical defense named **PDR system**. *GraphRfi*, a state-of-the-art robust GNN-based recommender system, was proposed to mitigate the effects of injected fake users. Unfortunately, we demonstrate that *GraphRfi* is still vulnerable to strong attacks likes MetaC due to the supervised nature of its fraudster detection component, where the clean labels

it relies on are hard to obtain in practice. Then, we design an adjustable fraudster detection module that explicitly considers label uncertainty. This module can serve as a plug-in that can be easily integrated into different models, resulting in the PDR system.

Thirdly, while empirical and certified robustness techniques have been developed to defend against graph modification attacks (GMAs), the problem of certified robustness against graph injection attacks (GIAs) remains largely unexplored. To bridge this gap, we introduce the **node-aware bi-smoothing framework**, which is the *first* certifiably robust approach for general node classification tasks against GIAs. Specifically, it randomly deletes nodes and edges of the graphs to obtain smoothed predictions with a large amount of random inputs. Through rigorous theoretical analysis, we establish the certifiable conditions of our smoothing scheme.

Finally, we present the *first* **collective certificate** against GIAs, significantly improving the certified performance compared to existing sample-wise certificates. Our collective certificate certifies a set of target nodes simultaneously, overcoming the limitations of previous approaches. We formulate the problem as a binary integer quadratic constrained linear programming (BQCLP) and develop a customized linearization technique to relax it into efficiently solvable linear programming (LP).

Through extensive evaluations, we demonstrate the effectiveness of our proposed adversarial attacks and defense techniques, paving the way for developing more robust and trustworthy graph learning models for real-world applications.

[475 words]

Publications

Publications Arising from the Thesis

1. **Yuni Lai**, Marcin Waniek, Liying Li, Jingwen Wu, Yulin Zhu, Tomasz P Michalak, Talal Rahwan, and Kai Zhou. "Coupled-Space Attacks against Random-Walk-based Anomaly Detection", *IEEE Transactions on Information Forensics and Security (TIFS)*, 2024.

2. **Yuni Lai**, Yulin Zhu, Wenqi Fan, Xiaoge Zhang, and Kai Zhou. "Towards Adversarially Robust Recommendation from Adaptive Fraudster Detection", *IEEE Transactions on Information Forensics and Security (TIFS)*, 2023.

3. **Yuni Lai**, Yulin Zhu, Bailin Pan, and Kai Zhou. "Node-aware Bi-smoothing: Certified Robustness against Graph Injection Attacks", in *The Forty-fifth IEEE Symposium on Security and Privacy (S&P)*, 2024.

4. **Yuni Lai**, Bailin Pan, Kaihuang Chen, Yancheng Yuan, and Kai Zhou. "Collective Certified Robustness against Graph Injection Attacks", in *The Forty-first International Conference on Machine Learning (ICML)*, 2024.

Other Publications during My PhD Study

- **Yuni Lai**, Jialong Zhou, Xiaoge Zhang, and Kai Zhou. "Towards certified robustness of graph neural networks in adversarial aiot environments", *IEEE Internet of Things Journal (IoTJ)*, 2023.

- Yulin Zhu, **Yuni Lai**, Kaifa Zhao, Xiapu Luo, Mingquan Yuan, Jian Ren, and Kai Zhou. "Binarizedattack: Structural poisoning attacks to graph-based anomaly detection", in *The IEEE 38th International Conference on Data Engineering (ICDE)*, 2022.

- Yulin Zhu, **Yuni Lai**, Kaifa Zhao, Xiapu Luo, Mingquan Yuan, Jian Ren, and Kai Zhou. "From Bi-Level to One-Level: A Framework for Structural Attacks to Graph Anomaly Detection", *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2024.

- Jialong Zhou, **Yuni Lai**, Jian Ren, and Kai Zhou. "Black-box attacks against signed graph analysis via balance poisoning", *International Conference on Computing, Networking and Communications (ICNC)*, 2024.

- Yuwei Han, **Yuni Lai**, Yulin Zhu, and Kai Zhou. "Cost Aware Untargeted Poisoning Attack Against Graph Neural Networks", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.

- Yulin Zhu, **Yuni Lai**, Xing Ai, and Kai Zhou. "Universally Robust Graph Neural Networks by Preserving Neighbor Similarity", in submission to *the 34th International Joint Conference on Artificial Intelligence (IJCAI)*, 2025.

Acknowledgments

I am extremely grateful to my supervisor, Dr. Kai Zhou, for providing me the opportunity to pursue a doctoral degree, which marked the beginning of my PhD journey in AI security. I would like to express my sincere appreciation to Dr. Kai Zhou for his strong support, meticulous guidance, warm encouragement, and kind understanding. He is the best advisor I have ever met, and I feel so lucky to be his first PhD student. I have learned a lot from him.

I would also like to express my gratitude to all my co-authors, especially Dr. Yulin Zhu. I am truly thankful for their valuable discussions, collaborative efforts on research ideas, experimental implementation, and thorough paper proofreading.

My sincere thanks go to all my friends, particularly Miss Qian Zhou. Their companionship has given me the courage and strength to navigate the ups and downs, enriching my life beyond academics.

Lastly, I would like to thank my grandmother, who brought me up, for her love and encouragement. Her encouragement has been a persistent source of strength and inspiration throughout my life. Her wisdom and guidance have shaped me into the person I am today.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Massive data in the form of graphs are ubiquitous in various domains, including social networks, financial systems, and e-commerce platforms. These graphs represent the complex relationships between entities, providing valuable insights for various applications. With the rapid development of Artificial Intelligence (AI), graph learning, particularly Graph Neural Networks (GNNs) [4, 5], has emerged as a powerful tool for learning and predicting from graph data [6]. GNNs have achieved significant success in fields closely related to people's daily life, such as social network analysis [7], financial fraudster detection [8, 9], and e-commerce recommender systems (RS) [10, 11, 12, 13, 14]. However, the increasing reliance on these graph machine learning models raises concerns about their trustworthiness and vulnerability to adversarial attacks. Many of these applications are critical for ensuring system security. Consequently, ensuring the trustworthiness of those graph learning models is of paramount importance.

The adversarial robustness of graph learning models has been a topic of significant research interest [15, 16, 17, 18]. Existing research [19, 20, 21] has demonstrated the effectiveness and prevalence of adversarial attacks on graph learning models. These attacks manipulate graph data to influence model predictions, enabling attackers to achieve malicious goals. For instance, in a recommendation system, attackers can create fake user accounts and generate positive reviews to promote specific products, manipulating the model's predictions and gaining unfair advantages [22, 23, 24]. Such attacks undermine the trustworthiness of graph learning models and

pose significant security risks to online activities. Addressing these threats is crucial for ensuring the reliability and security of AI systems that rely on graph data.

A mainstream approach to achieving trustworthy graph learning is to enhance the robustness of graph learning models. This involves two progressive levels: empirical robustness and certified (provable) robustness [25]. Empirical robustness [26, 27, 28, 15] focuses on adopting specific techniques to make the model resistant to attacks, with defense capability evaluated through extensive experiments. However, a key challenge with empirical robustness is that attackers can adapt their strategies, potentially compromising the model's defenses [29]. Certified robustness [30, 31, 32, 33, 34, 35] builds upon empirical robustness by establishing a theoretical framework that provides a guarantee for the model's defense effectiveness: As long as the attack intensity remains within a certain range, the model's predictions remain unchanged. This characteristic distinguishes certified robustness from empirical robustness, and no further potential attack will break the robustness. Enhancing both the empirical and certified robustness of graph learning models can significantly improve their trustworthiness and mitigate the potential harm caused by adversarial attacks.

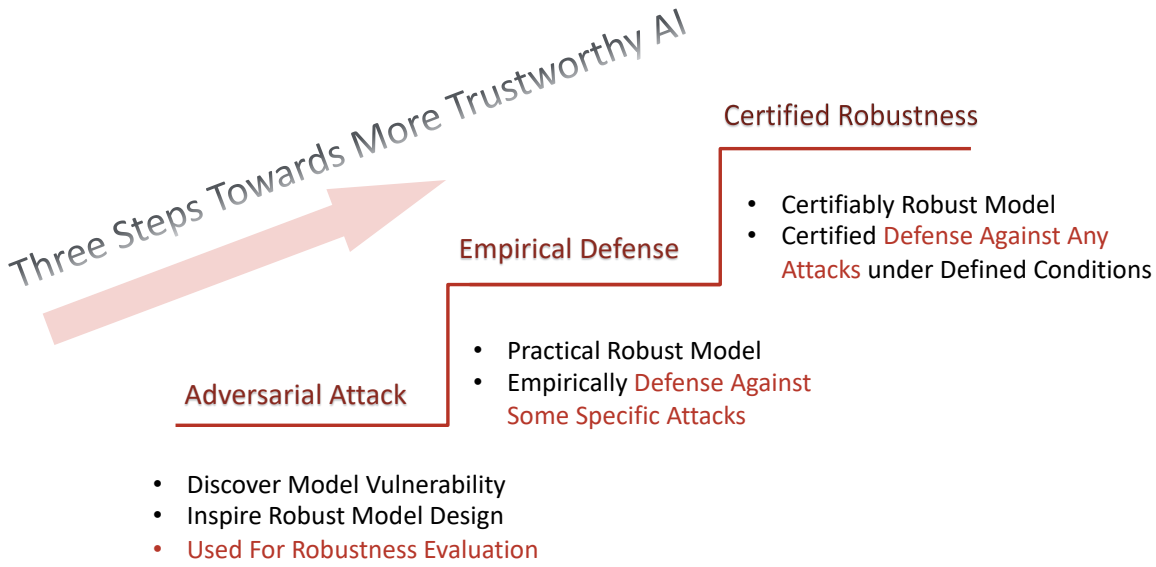


Figure 1.1: Relationship among adversarial attacks, empirical defense, and certified robustness.

However, current research on the vulnerability and robustness of graph learning models faces significant limitations due to the inherent richness of graph data and the evolving

complexity of adversarial environments. This thesis aims to investigate adversarial attacks, empirical defense, and certified defense in the face of the diversity of graph models. We propose adversarial attacks, aiming to evaluate and analyze the robustness of graph learning models. To defend against the attacks, we propose empirical defense by designing more robust graph learning models. Nevertheless, new and adaptive attacks might occur and break the empirical defense. To further safeguard the model, we propose certified defenses that are provably robust to some attacks in a specific format. Empirical defense is more popular (efficient but without theoretical guarantee), while certified defense is more rigorous (provably robust but high cost). Both of these defenses play essential roles in safeguarding graph learning. We further illustrate the relation among the three aspects in Figure. 1.1.

1.1 Adversarial Attack

The first goal of this thesis is to further investigate the vulnerability of graph models by designing fine-grain adversarial attacks in a more complex adversarial environment. Specifically, this thesis investigates two kinds of important target models: Random-walk-based anomaly detection models (RWAD) and recommender system (RS) models. Both of these two graph learning models exploit complex interdependency between multiple data dimensions. In the RWAD model, the graph and feature data space are interdependent and interacting. In the RS models, the graph structure and the ratings work together to affect the recommendation. We propose advanced attacks that can better capture the complex interdependency between the data dimensions.

Random Walks-based Anomaly Detection (RWAD) is graph-based models that commonly used to identify anomalous patterns in various applications [36, 37, 38, 39, 40, 41]. An intriguing characteristic of RWAD is that the input graph can either be pre-existing graphs or feature-derived graphs constructed from raw features [42, 43, 44, 45, 46]. Consequently, there are two potential attack surfaces against RWAD: graph-space attacks and feature-space attacks. It is worth noting that in the latter case, where the graph is not directly accessible, feature-space attacks are deemed more realistic. However, previous research treats attacks in the graph space

and feature space rather separately. On the one hand, many existing works have investigated *structural attacks* [19, 47, 48, 49] against a wide range of graph learning models. On the other hand, another line of research has focused on studying *feature manipulation attacks* [50, 51, 52] primarily in the computer vision domain, where the data objects represented by features are independent of each other. Unfortunately, no existing adversarial attack considers the interdependency between the graph space attack and feature space attack.

In the content of feature-derived graph-based models, in which the graph and feature data are interdependent in a complete way, we are motivated to explore the vulnerability by investigating the interplay between graph-space and feature-space attacks. To this end, we conduct a thorough complexity analysis, proving that attacking RWAD is NP-hard. Then, we proceed to formulate the graph-space attack as a bi-level optimization problem and propose two strategies to solve it: alternative iteration (alterI-attack) or utilizing the closed-form solution of the random walk model (cf-attack). Finally, we utilize the results from the graph-space attacks as guidance to design more powerful feature-space attacks (i.e., graph-guided attacks). In addition, we conduct transfer attack experiments in a black-box setting, which shows that our feature attack significantly decreases the anomaly scores of target nodes. Our study opens the door to studying the coupled-space attack against graph anomaly detection in which the graph space relies on the feature space.

Recommender systems (RS) are widely used in our daily lives, and they save people's time by showing them the most relevant information or products. However, adversarial attacks are ubiquitous among RS. Despite the development of vulnerability analysis for RS [22, 23, 24], existing approaches generate edges (which items the attacker chooses to insert ratings) and ratings (which rating scores the attacker gives) independently. These attacks ignore the complex interdependency between the edges and the ratings. As a result, we design a stronger attack, named MetaC, to optimize the attack profiles by considering the edges and ratings at the same time. Specifically, we adapt metattack [53], one of the state-of-the-art attacks originally designed for node classification tasks, to be suitable and powerful for RS attacks. We employ a continuous vector to encode the discrete ratings during the optimization and then discretize the ratings after the optimization. Our proposed MetaC attack is suitable for both GNN-based [54]

and Martix-Faxtorization-based [55, 56] recommender systems, showing the versatility of our attack.

1.2 Empirical Defense

Against the background of adversarial attacks, various defense mechanisms [57, 58] have been proposed to improve the adversarial robustness of RS against node injection attacks. Recently, *GraphRfi* [54], a Graph-Neural-Network-based (GNN-based) recommender system, was proposed and shown to mitigate the impact of injected fake users effectively. However, we demonstrate that *GraphRfi* remains vulnerable to MetaC attacks due to the supervised nature of its fraudster detection component, where obtaining clean labels is challenging in practice.

The second goal of this thesis is to explore the key research question of how to effectively integrate fraudster detection into RS, where the labels employed in training contain uncertainty. We first analyze why *GraphRfi* fails under MetaC attack. Then, based on our insights obtained from vulnerability analysis, we design an adaptive fraudster detection module that explicitly considers label uncertainty. This module can serve as a plug-in for different recommender systems, resulting in a robust framework named Posterior-Detection Recommender (PDR). Overall, our research presents a practical framework for integrating fraudster detection into recommendation systems to achieve adversarial robustness.

1.3 Certified Robustness

In recent years, although significant progress has been made in the field of certified robustness in graph learning [59, 34, 33, 35], the attacks targeted by these methods are mainly common graph modification attacks. However, another prevalent form of attack, called graph injection attack (GIA), is gaining attention [3, 60, 61]. Unlike graph modification attacks (GMA), which only add or delete edges on the original graph, GIAs involve injecting new nodes into the original graph. This characteristic makes GIAs more powerful and imperceptible [3, 60, 61]. Therefore, it is an urgent problem to study certified robustness that is applicable to GIAs.

Unfortunately, the existing certified robustness methods designed for graph structural attacks have limited effectiveness when applied to GIAs. The fundamental reason is that previous GMAs only manipulate the graph in a single way (e.g., deleting edges), while GIAs involve mixed-mode manipulations (e.g., adding nodes and edges simultaneously). As a result, the main method relied upon by previous certified robustness approaches, randomized smoothing [59, 62, 35], cannot handle such mixed-mode data manipulations, making it unable to provide effective theoretical guarantees for graph learning models against more diverse GIAs.

Therefore, **the third goal** of this thesis is to construct a more generalizable certified robust model and explore new theoretical proof methods to provide theoretical robustness guarantees in the face of complex and diverse attack methods. To bridge this gap, we introduce the *node-aware bi-smoothing* framework, which is the *first* certifiably robust approach for general node classification tasks against GIAs. Notably, the proposed node-aware bi-smoothing scheme is model-agnostic and is applicable for both evasion and poisoning attacks. Through rigorous theoretical analysis, we establish the certifiable conditions of our smoothing scheme. We also explore the practical implications of our node-aware bi-smoothing schemes in two contexts: as an empirical defense approach against real-world GIAs and in the context of recommendation systems. Furthermore, we extend two state-of-the-art certified robustness frameworks to address node injection attacks and compare our approach against them.

Current certified robustness methods have low certified ratios and need to improve their certified ratios to make them truly practical. Another major issue with existing certified robustness methods is that their certified ratios (i.e., the proportion of predictions that can be theoretically verified) are very low. This means that certified robustness is still in its early stage of theoretical usability and still has a long way to go before it becomes practically applicable. For example, as a representative work, [59] the certified ratio of the model drops to zero [59] when the number of deleted edges exceeds 25 (which is only about 0.3% of the original edges). In other words, this model can only guarantee stable predictions under minor attack strengths, which is far from practical requirements. The main reason for the low certified ratios in current methods is that it is difficult to estimate the attacker's strength accurately in the process of making assumptions and modeling the attacker. In order to facilitate theoretical proofs, we often overestimate the

attacker's strength to reduce the complexity of modeling, which directly leads to much lower certified ratios than expected in practice.

Therefore, **the fourth goal** of this thesis is to explore new approaches to improve the certified ratios of certified robustness models and promote the practical application of certified robustness as a valuable tool with theoretical guarantees. Existing research only provides *sample-wise* certificates [59, 62, 33] by verifying each node independently, leading to very limited certifying performance. In this thesis, we present the first *collective* certificate, which certifies a set of target nodes simultaneously. To achieve it, we formulate the problem as a binary integer quadratic constrained linear programming (BQCLP). We further develop a customized linearization technique that allows us to relax the BQCLP into linear programming (LP) that can be efficiently solved. Our thesis marks a crucial step towards making provable defense more practical.

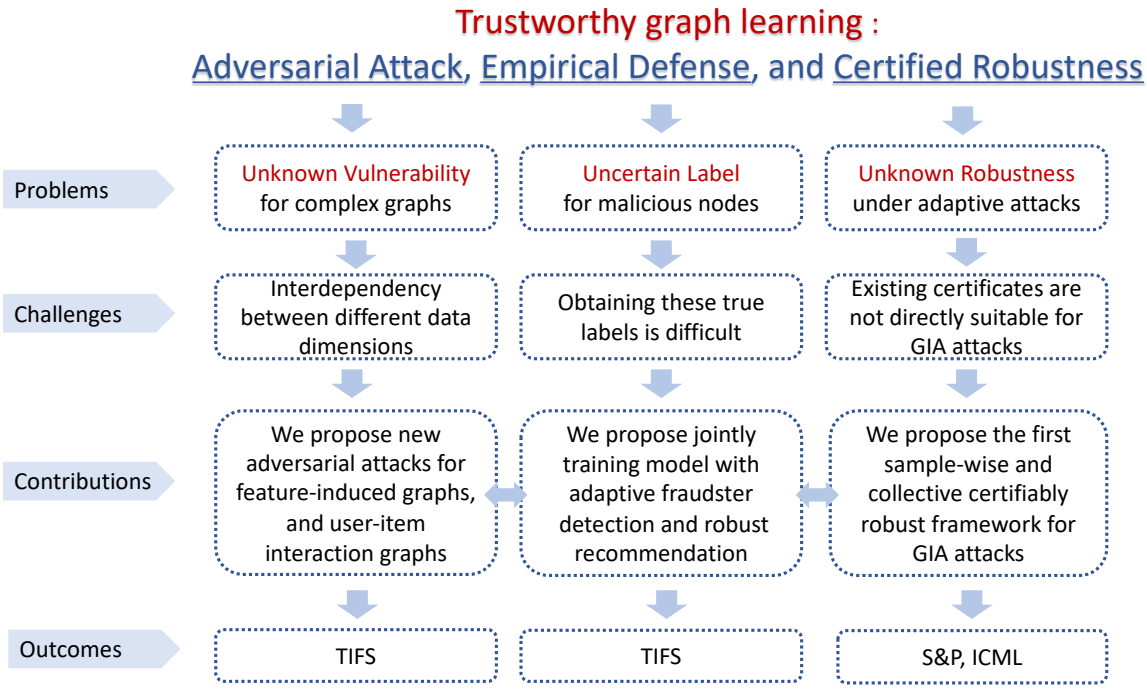


Figure 1.2: Overview of the thesis.

In summary, this thesis aims to systematically analyze the vulnerability by investigating powerful attacks and enhance the robustness of graph learning models by developing both empirical and certified defenses. The overview of this thesis is presented in Figure. 1.2. The contributions

of the thesis are listed as follows:

- To investigate the vulnerability of the graph learning model, we design **coupled-space** attacks for the random-walk-based graph anomaly detection model and **MetaC** attack for the GNN-based recommender system model. Comprehensive experiments demonstrate that our proposed attacks are effective with a limited attack budget.
- To investigate empirical defense against adversarial attacks, we propose an empirically robust recommender system (**PDR**). Comprehensive experiments show that our defense approach outperforms other benchmark methods under attacks.
- To investigate new certified defense, we propose **node-aware smoothing**, a certifiably robust graph model against graph injection attacks. Extensive evaluations demonstrate the effectiveness of our proposed certificates.
- To further improve the certifiable performance, we design a **collective certification** that models the locality of message-passing GNNs. Through comprehensive experiments, we demonstrate that our proposed scheme significantly improves certification performance with minimal computational overhead.

Chapter 2

Background and Related Works

In this chapter, we provide the necessary background and related work of graph learning models, especially in the context of adversarial attack, empirical defense, and certified robustness.

2.1 Random-Walk-based Anomaly Detection

In this section, we introduce the necessary background on unsupervised random-walk-based anomaly detection (RWAD). We first present an overview of the framework with an emphasis on the role of random walk (RW) in anomaly detection. Then, we give two concrete representative RWAD models, which are also the target models considered in this thesis.

Random-walks-based anomaly detection (RWAD) [38, 43, 40, 42, 44, 63, 45] , a classical unsupervised graph anomaly detection (GAD) technique, is widely used to safeguard the system from attacks by detecting abnormal data. RWAD, as discussed in this chapter, exploits random walks as a similarity or connectivity measurement. Traditional *feature-based* techniques [64, 65, 66] utilize statistical features, such as in and out node degrees, to extract structural information from graphs and transform the GAD to usual anomaly detection problem. For example, OddBall [64] built a regression model based on the density power law to estimate anomalous local patterns. These labor-intensive handcrafted features have limitations on generalizing to

unknown anomalies. Beyond handcrafted features, *network-representation-based* techniques, such as DeepWalk [67] and Node2Vec [68], are widely exploited to extract a more flexible feature representation which can be used for downstream anomaly detection tasks[69]. Most recent work mainly focuses on investigating *deep learning based* anomaly detection, such as DOMINANT [70], GAL [71], TAM [72], GLAD [73], and GAD-NR [74].

Nevertheless, RWADs have an irreplaceable role in GAD because of their simplicity, unsupervised, and effective features. These motivate us to investigate their vulnerability further in Section 3.1.

2.1.1 Input data as a graph

In general, RWAD takes a *plain* graph as input and produces anomaly scores for the nodes in the graph as output. In practice, the input graph could be either directly available or constructed from raw data. Depending on the levels of accessibility of the graph, we divide RWAD systems into two types:

- RWAD over *directly accessible graph* (*Di-RWAD*) [38, 39]: In this case, the input to RWAD is a graph that represents relational data in a specific application. For instance, in recommender systems, the rating towards products given by customers on E-commerce platforms can be modeled as a *bipartite graph*.
- RWAD over *indirectly accessible graph* (*InDi-RWAD*) [42, 43, 44, 45]: In this case, the input to RWAD are raw features of entities, and a graph is constructed as a data preprocessing step in the pipeline of anomaly detection (Fig. 3.1, top). Typically, given the feature vectors, a *proximity graph* is constructed, where the nodes represent entities and an edge exists between two nodes only if they are similar enough in certain similarity metrics.

We note that in both cases, RWAD will operate on graphs; however, the difference lies in whether the graph is directly accessible. Later we will see that such a difference is crucial for determining the attacker's ability when designing attacks.

2.1.2 RW as a similarity measurement

The core of unsupervised anomaly detection is to identify data points that are significantly different from the rest of the population. RW has been shown to be an effective method for measuring the similarities of nodes in a graph. Specifically, given a graph $G = (V, E)$ with its adjacency matrix denoted as W, we define the transition matrix $P = (p_{ij})_{|V|\times|V|}$ as the column-normalized version of the adjacency matrix W, where $p_{ij} = w_{ij}/\sum_{t=1}^{|V|} w_{i,t}$. If vertex i has no outgoing edges (i.e., $\sum_{t=1}^{k+n} w_{i,t} = 0$), we set the transition probability to 0. The widely used Page-Rank algorithm with restart can be represented as follows:

$$\vec{s} = (1-\alpha)P\vec{s} + \alpha\vec{r}, \tag{2.1}$$

where α is the restart rate, a hyper-parameter that controls the probability of restart; the vector $\vec{r}$ specifies the restart strategy, and $\vec{s}$ characterizes the node similarities. With the similarity, the anomaly score of a node is calculated as the opposite of its average similarity to all other nodes, or the average similarity among its neighbors.

2.1.3 Adversarial attacks on random walk

Our work belongs to the category of targeted and poisoning adversarial attacks. Here, we include the most related existing attacks on graph models. There are some previous research efforts on the random walk (RW) based models. [75] reformulate the DeepWalk model as a matrix factorization form to reduce the bi-level optimization to single-level, and then optimize the untargeted attack loss by optimizing the graph spectrum. [76] make further improvements to make the spectrum-based attack work in a black-box system. Different from our attacks on RW-based anomaly detection, they mainly focus on attacking node embedding generated by RW.

In addition to RW-based model, Nettack [19], Metattack [20] are two strong poisoning attacks for the GCN-based models. Nettack greedily selects the perturbation edges among the candidate sets with the largest gradient obtained by incremental updates. Metattack greedily selects the perturbation edges with the largest gradient obtained by meta-gradient. Note that both of these

methods can be extended to attack node features. However, Nettack does not introduce the attack node selection, and Metattack is only applicable to binary features. Furthermore, the proximity graph is different from other graphs. The proximity graph is changing along with features, while the node feature attack in [19],[20] have fixed graph structures. For belief propagation models, [77] introduced a poisoning attack for graph data. For another classical graph-based anomaly detection model called OddBall, [49] proposed BinarizedAttack, which is well-designed for the binary property of edges. For graph contrastive learning, [78] attacks the graph embedding by greedily choosing the most informative edges. Beyond gradient-based methods, perturbing the intrinsic property of graphs, such as spectral changes [79] shows to be more effective, but it is only suitable for untargeted attacks. These works are orthogonal to our study.

2.2 Recommender Systems

In this section, we introduce the recommender system (RS) background by first providing an overview and then introducing two representative RSs.

The recommender system (RS) plays an important role in our daily lives as it saves us time by preventing us from accessing tremendous amounts of information. However, the vulnerability of RS gives the opportunity for malicious attackers to manipulate the recommendation. For this reason, this thesis aims to investigate their vulnerability further and propose a stronger defense strategy.

A recommender system (RS) typically operates on a weighted bipartite graph $\mathcal{G} = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$, where $\mathcal{U} = \{u_1, \cdots, u_n\}$ is a set of n users, $\mathcal{V} = \{v_1, \cdots, v_m\}$ is a set of m items, and the edge set $\mathcal{E} = \{e_{ij} = (u_i, v_j, r_{ij})\}$ is a collection of observed ratings with $r_{ij} \in \{1, 2, \cdots, r_{max}\}$ denoting the rating from user u_i to item v_j. Each user u_i is also associated with a feature vector $\mathbf{x}_i$ summarizing this user's behavioral information. The task of recommendation thus amounts to predicting the weights of missing edges and recommending highly ranked items to users.

Recommender systems can be implemented using various techniques. One of the most classical methods is Matrix Factorization (MF) [56]. More recently, graph representation learning tech-

niques, such as Graph Neural Networks (GNNs), have been increasingly utilized to improve prediction performance [11, 12, 13, 14]. In the following, we introduce some representative models to provide background on the techniques used in recommender systems.

2.2.1 MF-based RS

Matrix factorization (MF)-based recommendation models, such as SVD [80], Regularized SVD [81], and Improved Regularized SVD [82], are widely used in recommendation systems due to their simplicity and effectiveness. For instance, Regularized SVD predicts missing values in the history rating matrix by decomposing it into user embedding matrix U and item embedding matrix V. The embedding matrices U and V are learned by regression on existing/history ratings as follows:

$$\arg\min_{U,V} \sum_{\forall(u,v)\in\mathcal{E}} (r_{uv} - U_u^T V_v)^2 + \beta(\|U\|_2 + \|V\|_2), \tag{2.2}$$

where the U_u and V_v denote the embeddings for user u and item v respectively, the matrices $U \in \mathbb{R}^{|U|\times d}$ and $V \in \mathbb{R}^{|V|\times d}$ represent the collection of user and item embeddings, d is the factor number, and β is the regularization factor.

2.2.2 GNN-based RS

We use *GraphRfi* as a representative to introduce GNN-based RS. To mitigate *node injection attacks*, a robust RS *GraphRfi* was introduced that combines recommendation with fraudster (i.e., fake users) detection. In particular, *GraphRfi* has two essential components: a rating prediction component based on GNN and a fraudster detection component based on Random Neural Forest (RNF) [83]. The main idea of *GraphRfi* is to treat the anomaly score of a user (from the fraudster detection component) as her weight in estimating the ratings, thus mitigating the effects of anomalous users. The backbone of this RS model is GNNs with attention aggregators. It encodes the discrete ratings by learnable embeddings $e_r \in \mathbb{R}^{d'}$, $r \in \{1, 2, \cdots, r_{max}\}$, where d' is the embedding dimension. For each user/item, it concatenates the rating embedding with user/item embedding, and then two single-layer GNNs are employed to learn the user's and

item's representation,

$$z_u = ReLU(W_1 \cdot Agg(\text{MLP}(x_v \oplus e_{r_{uv}}), \forall v \in \mathcal{N}(u)) + b_1),$$
$$z_v = ReLU(W_2 \cdot Agg(\text{MLP}(x_u \oplus e_{r_{uv}}), \forall u \in \mathcal{N}(v)) + b_2),$$

where x_u/x_v is initial user/item embedding, $\oplus$ is concatenation, and $Agg(\cdot)$ is attention aggregation function,

$$Agg(h_k, \forall k \in \mathcal{N}(s)) = \sum_{k \in \mathcal{N}(s)} \alpha_{ks} h_k,$$

and the α_{ks} is the weight learned by attention layer,

$$a_{ks} = W_4 \cdot \sigma(W_3 \cdot (h_k \oplus z_s) + b_3) + b_4,$$
$$\alpha_{ks} = \frac{exp(a_{ks})}{\sum_{k' \in \mathcal{N}(s)} exp(a_{k's})}.$$

In other words, GNNs are used to learn the embeddings of both users and items denoted as z_u and z_v, which are further used to compute the predicted rating r'_{uv} from user u to item v through a multi-layer perceptron (MLP):

$$r'_{uv} = W_5 \cdot \text{MLP}(z_u \oplus z_v),$$

where W_i, b_i are the learnable parameters, $\mathcal{N}(s)$ is the neighbor set of node s.

Given the user embedding z_u learned by GNNs, a classifier (i.e, RNF) is used to estimate the probability that a user u is normal, denoted as $\mathbb{P}[y = 0|z_u, \theta]$, where θ is the model parameter, and $y = 0$ indicates that a user is normal. Finally, the prediction and detection components are jointly trained in an *end-to-end* manner by minimizing the following loss function consisting of two parts:

$$\begin{aligned} \mathcal{L}(\theta, \mathcal{G}) &= \mathcal{L}_{\text{rating}} + \lambda \cdot \mathcal{L}_{\text{fraudster}} \\ &= \tfrac{1}{|\mathcal{E}|} \textstyle\sum_{\forall (u,v) \in \mathcal{E}} \mathbb{P}[y = 0|z_u, \theta] \cdot (r'_{uv} - r_{uv})^2 \\ &+ \lambda \cdot \tfrac{1}{|\mathcal{U}|} \textstyle\sum_{\forall u \in \mathcal{U}, y_u \in \mathcal{Y}} \left(-\log \mathbb{P}[y = y_u|z_u, \theta]\right), \end{aligned} \tag{2.3}$$

where $\mathcal{L}_{\text{rating}}$ summarizes the weighted mean squared error of rating and $\mathcal{L}_{\text{fraudster}}$ is the cross-entropy loss for anomaly detection with y_u denoting the ground-truth label of user u.

The probability $\mathbb{P}[y = 0|z_u, \theta]$ serves as the weight for user u. As a result, a user with a high anomaly score (i.e., $1-\mathbb{P}$) contributes less to the prediction, which can enhance the robustness of the recommendation under node injection attacks. We can also notice that the fraudster detection component is supervised in nature as the ground-truth labels are required during training; in Section 3.2, we will show the defects of this design by designing a powerful attack **MetaC**.

2.2.3 Adversarial attacks on recommender systems

Injecting nodes into the recommender system is the major attacking approach, as it could be easily implemented in practice. The difficulty, however, lies in the selection of items and the ratings they give. Earlier attacks [84, 85] rely on choosing filler items by heuristic rules and giving the highest/lowest ratings to the target items, depending on the goal of pushing or nuking items. However, these attacks are not effective enough as shown by [86] and [87, 88].

Recently, more sophisticated methods have been proposed based on techniques such as optimization, generative models, and so on. For instance, [89] proposed a method to optimize the selection of filler items using approximated gradients to attack MF-based RS. [90] train a poisoned RS model to predict the ratings for filler items. In addition, another line of works [91, 88, 87] explore the utilization of generative models (e.g., GAN [92]) to generate fake users profiles, which are injected into the system. However, previous works mainly focus on MF-based models due to their simplicity and could not be easily extended to GNNs-based systems. For example, the approximating gradients proposed in [89, 91] cannot be directly applied to GNNs-based models like *GraphRfi*. This motivates us to design new attack for RS in Section 3.2.

2.2.4 Empirical defenses on Recommender System

The primary method to achieve the adversarial robustness of RS is through adversarial training, which has been tested to be effective in many other machine learning systems. [93] and [58] perform adversarial training by adding perturbation noise to model parameters in each training iteration to improve the robustness of different target models. [57] train a robust MF-based RS

via injecting some defense users based on the calculation of influence functions. Again, it is nontrivial to extend such an idea of defense to GNNs-based systems due to the complexity of estimating the Hessian matrix in the influence function.

The detection of fraudsters (or anomalies) is closely related to defense, as it is a natural way to identify injected fake users. Anomalous users may exhibit patterns that deviate from those of genuine users, such as rating several items in a similar pattern or providing overly positive or negative reviews. Feature-based anomaly detection methods [94, 95] extract the user features based on user behavior, such as the number of ratings, rating time, and review content, and then apply classification techniques to identify the abnormal users. In addition, embedding-based anomaly detection methods based on user-item rating graphs have been developed [96, 97]. These methods aim to learn embeddings that capture user behavior patterns in graphs, where the node embedding of the fraudster deviates from that of normal users. Due to difficulty in obtaining the label, unsupervised methods such as clustering [98, 99, 100, 96] and semi-supervised methods [101, 102] are widely used in detection. However, we emphasize that anomaly detection is often employed as a preprocessing step.

Given the current under-explored status of GNNs-based RS, in Chapter 4, we aim to investigate the adversarial robustness of the representative model *GraphRfi*.

2.3 Node classification task

Beyond the anomaly detection and recommendation model, the graph learning model is versatile and can be used for various node classification tasks, such as social network, transaction network, and citation network analysis [103, 17]. A further study on node classification tasks broadens the research scope of this thesis. In this section, we first formally define the node classification task and then introduce the most representative graph neural network.

A graph with n nodes is represented as $G = (\mathcal{V}, \mathcal{E}, X) \in \mathbb{G}$, where $\mathcal{V} = \{v_1, \cdots, v_n\}$ is the set of nodes, $\mathcal{E} = \{e_{ij} = (v_i, v_j)\}$ is the set of edges with each edge e_{ij} linking v_i and v_j, and $X \in \mathbb{R}^{n \times d}$ are node features with dimension d. The graph structure of G can also be encoded

by adjacency matrix $A \in \{0,1\}^{n\times n}$ with $A_{ij} = 1$ if $e_{ij} \in \mathcal{E}$ and $A_{ij} = 0$ if $e_{ij} \notin \mathcal{E}$. Some of the nodes are associated with a label $y \in \mathcal{Y} = \{1, \cdots, C\}$. The task of node classification is to predict the missing node labels. To this end, a graph-based classifier $f : \mathbb{G} \to \{1, \cdots, C\}^n$ takes graph G as input and is used to predict the labels. We consider both the inductive and transductive settings. Specifically, in the transductive setting, the model trained on graph G can only make predictions for the current nodes in G, while the inductive classifier can make predictions for graphs with new nodes.

2.3.1 Message-Passing Graph Neural Networks

Graph Neural Networks (GNNs) are the mainstream model for node classification. In this thesis, we study certified robustness approaches that are applicable to the most commonly used GNNs that operate under the message-passing framework based on neighbor aggregation. These message-passing GNNs [4, 104, 105, 106] encode the local information of each node by aggregating its neighboring node features (i.e., embedding) through various aggregation functions. To generate the embedding at the higher layer of node v, GNNs aggregate the current node embedding $h_v^{(l)}$ and its neighboring node embedding $h_u^{(l)}, \forall u \in \mathcal{N}(v) := \{u | A_{uv} = 1\}$. We generally represent the aggregation layers in k-layer GNNs as follows:

$$
\begin{aligned}
&h_v^{(0)} = X_v, \\
&h_v^{(l+1)} = \Psi(Agg_{u\in\mathcal{N}(v)}^{(l+1)}(h_v^{(l)}, h_u^{(l)})),\ (\forall\, l = 0, \cdots, k-1),
\end{aligned}
$$

where X_v is the feature vector of node v, $Agg(\cdot)$ represents the general aggregation function depending on the type of GNN model, and $\Psi(\cdot)$ represents the learnable feature extraction function. These aggregation layers all together form the k-layer GNNs as parametrized functions $f(\cdot)$ for the K-class classification models by taking the maximum index of the final representation $h_v^{(k)}$: $f_v(G) = \arg\max_c h_{v,c}^{(k)}$. Note that, well-trained GNNs under inductive learning schemes are able to give predictions for new graphs/nodes without any retraining. During the inference, the receptive field of a node v in k-layer GNN is just its k-hops neighbors, and the nodes/edges beyond the receptive field would not affect the prediction of the node when the model is given. This locality enables the application of collective certificates.

2.3.2 Graph Injection Attack

Graph injection attack (GIA) is a type of attack that manipulates the structure of a graph by injecting malicious nodes with carefully crafted features to degrade the performance of node classification. For instance, a representative example, HAOGIA [3], first formulates an adversarial objective function comprising two key components: an attack objective and an unnoticeable objective. The attack objective guides the attacker in accomplishing their malicious goal, while the unnoticeable objective aids the attacker in evading detection by defenders. Subsequently, the method utilizes gradient descent to iterate and locate the optimal edges and node features that maximize the objective function. We note that a GIA can occur at test time (i.e., evasion attack [3, 61, 107]) and training time (i.e., poisoning attack [108, 109, 110, 111, 112, 113, 114]). Specifically, the former will manipulate the testing data to disrupt a trained model, while the latter will manipulate the training data causing changed model parameters. Our proposed scheme is applicable to both evasion and poisoning attacks.

A common requirement for GIA is to remain stealthy such that the injected malicious nodes can not be spotted by detectors. To ensure stealthiness, many existing attacks impose a constraint that the degree of an injected node should not exceed the average degree of the clean graph [108, 115, 110]. In particular, attack methods like G-NIA[115], G^2A2C [61], and G^2-SNIA [107] adopt a default strategy of inserting a single malicious node with a single edge. *This motivates us to investigate certified robustness against such constrained while more realistic graph injection attacks in Chapter 5.*

2.4 Certified Robustness of Graph Learning Models

To defend against attacks, numerous approaches have been proposed, especially empirical approaches, such as adversarial training [116], robust models [4, 26, 27, 28], and anomaly detectors [54]. Additionally, certified approaches provide provable robustness within a predefined perturbation set, representing the attacker's capabilities. The techniques for achieving certified robustness can be divided into two main categories: precise certification [31] and probability

certification based on random smoothing [30, 117]. Precise certification is usually designed for specific models, so it has limitations. On the other hand, probability certification based on random smoothing is more scalable and flexible, making it applicable to any graph model, and has become a mainstream approach. The research in this thesis belongs to the latter category.

Probability certification based on random smoothing attempts to estimate the probabilities of model outputs under given attack capabilities by using randomization techniques, such as introducing Gaussian noise. Then, theoretical conditions for maintaining unchanged predictions can be derived.

2.4.1 Randomized Smoothing

A mainstream technique to achieve certified robustness is *randomized smoothing* [59, 34, 33, 35, 117, 118]. It provides probabilistic certified robustness by adding random noise to the input samples. One representative smoothing scheme designed for graph data is the sparsity-aware smoothing method [59]. It offers an l_0-ball guarantee for graph modification attacks (GMAs). This guarantee specifies the maximum number of edges that can be added or deleted among existing nodes while maintaining consistent predictions. It achieves this by creating a smoothed classifier through randomization, which involves randomly deleting or adding edges to the input samples. Based on this randomization, denoted as $\phi(\cdot)$, it constructs a smoothed classifier:

$$g_v(G) := \underset{y \in \{1,\cdots,C\}}{\arg\max}\ \mathbb{P}(f_v(\phi(G)) = y), \tag{2.4}$$

where $f : \mathbb{G} \to \{1, \cdots, C\}^n$ returned the class y given a randomized graph $\phi(G)$, and smoothed classifier $g(\cdot)$ returns the "majority votes" of the base classifier $f(\cdot)$. Then it estimates the probability of the model's output given the perturbed graph based on the prediction on the clean graph and the sample overlap probability. The smoothed classifier has a certified consistent prediction if the top class probability p'_A is larger than the runner-up class p'_B under any perturbed graph in the perturbation set.

However, most existing work in this area has focused on computer vision, particularly on classification problems for images [119, 120, 121, 122] and tabular data [123, 124]. Research on

the certified robustness of machine learning tasks on graph data, such as graph classification, node classification, and link prediction, is still in its early stage of development. Specifically, recent research on certified robustness of graph data has mainly focused on graph structure attacks, with representative works including [59, 34, 33, 35, 125]. Among them, Wang et al.[34] first extended the random smoothing method to graph data and established certified robustness guarantees for any graph neural network for node classification and graph classification. Jia et al.[33] used a similar approach to study the certified robustness of graph community detection. Bojchevski et al.[59] improved the computational efficiency of [34, 33] by reducing the number of constant likelihood ratio partitions and can handle perturbations on both graph structure and node attributes. However, as mentioned earlier, these research approaches only apply to graph modification attacks (GMAs), and extending them to graph injection attacks (GIAs) would result in low certified ratios.

Additionally, the certified robustness mentioned earlier only applies to evasion attacks (attacks that only manipulate test samples), and only a few studies [126, 127, 128, 129] provide certified robustness against poisoning attacks (attacks that manipulate training samples) on image data. For example, Rosenfeld et al.[126] first extended the random smoothing method to label-flipping poisoning attacks. Wang et al.[127] and Weber et al.[128] extended the random smoothing method to defense against backdoor (poisoning) attacks. Levine et al. [129] proposed an aggregation-based certification scheme to defend against poisoning attacks, including label flipping and sample insertion. However, due to the significant differences between image data and graph data, the above methods cannot be directly and effectively extended to graph learning models. To the best of our knowledge, this thesis (Section 5.1) is the first to study the certified robustness of graph learning models against graph injection attacks, which can be applied to both evasion and poisoning attacks, demonstrating high originality and uniqueness.

2.4.2 Collective Certified Robustness of Graph Learning Models

Current work on certified robustness for graph learning models [59, 34, 33, 121, 32] has mainly focused on **sample-wise certification** against graph structure attacks, which essentially exam-

ines the predictions for each individual node. A critical assumption in this approach is that the attacker can create a different perturbed graph each time to attack a single node. However, in reality, the attacker can only generate one perturbed graph and disrupt the predictions of a group of target nodes simultaneously. This difference motivates the development of a groundbreaking approach to improve the certified robustness, known as **collective certification** [130, 35].

Schuchardt et al. [130] introduced collective certification by considering a more realistic and constrained attacker who can only use one perturbed input at a time to disrupt the predictions of as many nodes as possible in the graph. This significantly improves the certified robustness. However, this approach is not applicable to graph injection attacks. This is because the certification scheme assumes that the graph neural network has a fixed receptive field, while graph injection attacks involve adding edges after injecting nodes, inevitably expanding the receptive field.

Despite the progress made in defending against graph modification attacks (GMAs) [130], the robustness against graph injection attacks (GIAs) has received relatively little attention. [2, 131] further extended it to certify against GIAs. However, these models provide sample-wise certificates instead of collective ones. In essence, this greatly exaggerates the actual capabilities of the attacker, resulting in limited effectiveness in achieving certified robustness. To the best of our knowledge, there is currently no collective certificate designed for GIAs. This also inspires us to design a collective certification method that is applicable to GIAs to enhance the certified robustness in Section 5.2.

Chapter 3

Adversarial Attacks on Graph Models

This chapter aims to further investigate the vulnerability of graph models by designing fine-grain adversarial attacks in a more complex adversarial environment. Specifically, this thesis investigates two kinds of important target models: Random-walk-based anomaly detection models (RWAD) and recommender system (RS) models.

3.1 Coupled-Space Attacks against Random-Walk-based Anomaly Detection

Graph-based Anomaly Detection (GAD) has gained significant research attention in recent years due to the widespread use of graph data across various application domains. GAD algorithms are designed to identify anomalies in a graph, where nodes represent entities, and edges indicate their relations. Essentially, a GAD algorithm works by initially measuring the similarities among nodes and then identifying nodes that are less similar to the rest as anomalous. Despite the development of supervised GADs, such as GADs based on graph neural networks (GNNs) [132], unsupervised GADs still have advantages in their simplicity, unsupervised property, and effectiveness. Random Walks (RWs), such as PageRank [133], have emerged as a powerful tool for measuring node similarities over graphs and have become a fundamental

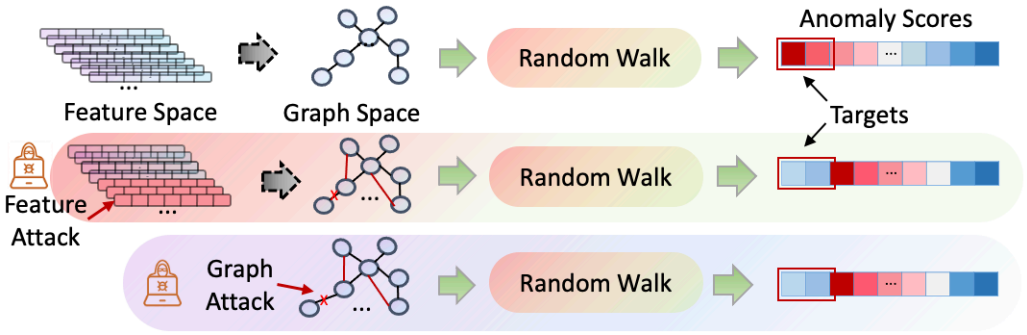


Figure 3.1: Illustration of RW-based anomaly detection and the distinction between graph-space and feature-space attacks.

component of many unsupervised GAD systems that are extensively employed in diverse applications. Notably, Random-Walk-based Anomaly Detection (RWAD) has been employed in detecting money laundering within the financial industry [36], identifying fraudsters in online shopping [37], uncovering fake accounts in social networks [134, 135, 136, 137, 138], and serving as a general unsupervised outlier detection method for bipartite graphs [38, 39] (e.g., review data in recommender systems, stock market transaction data, and short message service), multivariate time series data [40, 41] (e.g., electrocardiograms data), and the most common feature data [42, 43, 44, 45, 46] (e.g., network intrusion detection data). Moreover, random walk has also been adopted to improve large-scale graph anomalies detection [139] and enhance deep-learning-based anomalies detection [71, 140]. These diverse applications underscore the important role of RWAD in ensuring system security.

As the accuracy of predictions produced by the RWAD methods is crucial for system security, it is essential to assess their robustness in a real-world adversarial environment. In fact, the individuals that RWAD aims to detect may have both the incentive and capability to evade detection. For instance, adversaries controlling bank accounts to be used in money laundering schemes may wish to remain undetected to continue their malicious activities. They could carefully manage the everyday transactions on the accounts to make them appear similar to normal ones, causing the system to falsely classify them as benign. In essence, in an adversarial environment, attackers can intentionally manipulate the input data to RWAD in order to mislead its predictions, leading to what is known as *data poisoning attacks* in the literature. However,

studying the adversarial robustness of RWAD imposes new challenges due to an intriguing characteristic of RWAD. Specifically, in an RWAD system, the graph is often not directly accessible and needs to be *constructed* from raw data. As illustrated in Fig. 3.1 (top), entities in the system are represented as vectors in a feature space, and a graph is then constructed based on the relationships among the entities *as determined by their feature vectors*. This kind of graph is termed as feature-derived graph. For instance, a proximity graph can be constructed based on feature similarity. This graph is then fed into the RWAD system, which produces anomaly scores for each node.

Consequently, there are *two potential attack surfaces* against RWAD: **graph-space** attacks and **feature-space** attacks. In graph-space attacks (Fig. 3.1, bottom), the attacker can directly modify the structure of the graph, which is a common assumption made by previous works [19, 20, 49] that design structural attacks on graphs. In feature-space attacks (Fig. 3.1, middle), the attacker does not have direct control over the graph but can modify the features, which indirectly affects the graph's structure. It is worth noting that in the latter case, where the graph is not directly accessible, feature-space attacks are deemed more realistic (further explained in Section 3.1.4.1).

Unfortunately, previous research treats attacks in the graph space and feature space rather separately. On the one hand, many existing works have investigated *structural attacks* [19, 47, 48, 49] against a wide range of graph learning models. On the other hand, another line of research has focused on studying feature manipulation attacks [50, 51, 52] primarily in the computer vision domain, where the data objects represented by features are independent of each other. In contrast, one unique characteristic of RWAD is that it examines data objects that are interdependent. Specifically, the data processing pipeline of RWAD involves transforming the features into graphs, over which the random walk operates. That is, the data in the feature space and the data in the graph space are interdependent in the sense that any modifications to the features will be reflected in the changes in the constructed graphs. This unique interdependency makes the interplay between the graph-space and feature-space attacks possible.

Thus, for the first time, we aim to investigate the adversarial robustness of RWAD under ***coupled-space*** attacks, where the attackers can explicitly exploit the interdependency between two cou-

pled data spaces to effectively achieve their malicious goals. Our main motivations for exploring coupled-space attacks are twofold. First, data manipulation in the feature space is more realistic since the graph is constructed *virtually* in the pipeline of which the attacker does not have direct control. Second, since random walks directly run over the constructed graphs, an attacker can potentially leverage the anticipated data manipulation in the graph space to guide the modifications in the feature space, which can make the attack more effective.

Towards this end, we begin with a formal analysis of graph-space attacks. Specifically, we define the attacks in the graph space as a decision problem. We ask whether an attacker can reduce the anomaly scores of the target nodes below a certain threshold, thereby classifying them as benign, by modifying a limited number of edges in a given graph. Our in-depth complexity analysis shows that this problem is NP-hard for both *directed* and *undirected* graphs. Furthermore, since feature-space attacks ultimately modify edges, they can be viewed as special cases of this problem, and the hardness results remain applicable. The hardness results serve as the anchor for us to investigate efficient attack algorithms in both the graph space and feature space.

We then proceed to design effective graph-space attacks, which are formulated as an optimization problem with the objective of minimizing the target nodes' anomaly scores output by RWAD. Solving this optimization problem encounters several challenges. Firstly, random walk (PageRank) is an iterative algorithm that operates on an input graph; thus, any changes made to the graph will require the iterations to be re-executed. Consequently, attacks against RWAD will result in a *bi-level* optimization where the inner layer involves complex iteration. Second, the discrete nature of graph structure further complicates the solving of the optimization. To address these challenges, we propose two efficient attacks: **alterI**-attack and **cf**-attack. The former is an iterative approach that optimizes the attack objective by projected gradient descent (PGD) [77] and updates the random walk model alternatively. The latter utilizes the closed form of the random walk model to transform the bi-level optimization into a single-level problem.

Finally, we investigate the more realistic feature-space attacks. Our major innovation is to use the results from the *virtual* graph-space attack as our guidance to design more powerful feature-

space attacks. Specifically, we utilize the guidance from two aspects: selecting the attack nodes and formulating an effective attack objective. Through extensive experiments, we demonstrate that by fully exploring the dynamics between attacks in coupled spaces, more powerful attacks could be designed, revealing more realistic security threats against RWAD systems.

The main contributions are summarized as follows:

- We study the adversarial robustness of RWAD, for the first time, exploring the interplay between attacks in coupled spaces.
- We present a deep theoretical analysis of the hardness of attacking RWAD, which is proved to be NP-hard on both directed and undirected graphs.
- We propose effective attacks in coupled spaces. In particular, we innovatively utilize the results from the graph-space attacks as guidance to design more powerful feature-space attacks.
- We conduct comprehensive experiments to demonstrate the effectiveness of our proposed attacks. Especially we also transfer our attacks to other anomaly detection methods in the feature space. It is shown that our graph-guided feature-space attack remains effective even without knowing the target models, demonstrating a realistic threat in real-world application scenarios.

In summary, our work uncovers a unique vulnerability of RWAD and unleashes the power of attackers by exploring the interplay between attacks in coupled spaces, significantly advancing our knowledge of the adversarial robustness of RWAD in deployment.

Road Map: Problem statements (3.1.1) $\Rightarrow$ Complexity analysis of attacks (3.1.2) $\Rightarrow$ Effective graph-space attacks (3.1.3) $\Rightarrow$ Graph-guided feature-space attacks (3.1.4) $\Rightarrow$ Evaluation (3.1.5) $\Rightarrow$ Limitation and future work (3.1.6) $\Rightarrow$ Conclusion (3.1.7).

3.1.1 Problem Statements

In this section, we first present two representative models to instantiate the *Di-RWAD* and *InDi-RWAD* systems, which have been introduced in Section 2.1. Then, we introduce the adversarial environment in which random-walk-based anomaly detection (RWAD) operates and then formally define the attack problem.

3.1.1.1 RWAD Target Models

1) Di-RWAD. We consider bipartite graphs as a representative example of directly accessible graphs. We next describe how to apply the RWAD algorithm to the bipartite graphs of this kind, which we term as *BiGraphRW* model. In the following, we define the notation and briefly explain the *BiGraphRW* model.

To begin, we define a bipartite graph $G = (U \cup V, E)$ as a graph with two disjoint sets of vertices $U = \{u_i | 1 \leq i \leq k\}$ and $V = \{v_i | 1 \leq i \leq n\}$, and a set of edges $E \subseteq U \times V$ that connect the vertices in U to the vertices in V. We represent the edges in E as a binary edge matrix $M = (m_{ij})k \times n$, where $m_{ij} = 1$ if $\langle i, j \rangle \in E$, and $m_{ij} = 0$ otherwise. Then, the adjacency matrix for a bipartite graph can be constructed as $W = (w_{ij})_{(k+n)\times(k+n)} = \begin{pmatrix} 0 & M \\ M^T & 0 \end{pmatrix}$.

For each node $u \in U$, *BiGraphRW* applies Eq. (2.1) with $\vec{r} = \vec{e}_u$, where $\vec{e}_u$ is a vector with zeros element except node u, which means that it always restarts from node u. The resulting vector $\vec{s}_u = (1 - \alpha)P\vec{s}_u + \alpha\vec{e}_u$ represents the connectivity scores of node pairs $\{\langle u, t \rangle | t \in U \cup V\}$, which quantifies the similarity between node u and others. By assumption, a node v tends to have a lower mean similarity score among its neighbors if it is anomalous. We denote the average neighbor similarity as $\bar{S}_v$:

$$\bar{S}_v = \frac{\sum_{i=1}^{k} M_{iv} \sum_{j=1, i\neq j}^{k} M_{jv} \vec{s}_{u_i}(u_j)}{\sum_{i=1}^{k} M_{iv} \sum_{j=1, i\neq j}^{k} M_{jv}}, \tag{3.1}$$

where $\vec{s}_{u_i}(u_j)$ represent the element corresponding to node u_j in $\vec{s_{u_i}}$, which is the similarity between node u_i and u_j. Anomaly score of node v is in contract to the mean similarity score $\bar{S}_v$,

so we denoted it by

$$\mathcal{A}(v) = 1 - \bar{S}_v = \begin{cases} \text{anomaly}, & \text{if } \mathcal{A}(v) \geq \theta, \\ \text{normal node}, & \text{if } \mathcal{A}(v) < \theta, \end{cases} \tag{3.2}$$

where the parameter θ is a given and fixed threshold of the anomaly detection model.

2) InDi-RWAD. A representative way to apply RWAD to non-graph data is by constructing a proximity graph. We call this variant as *ProxGraphRW* model. In this approach, the input feature data is represented as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$, $\mathbf{x}_i \in \mathbb{R}^d$.

The first step is to construct a proximity graph according to the similarity or distance measurement between each pair of samples. To construct a proximity graph $G = (V, E)$, the vertices V represent data samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, and the edges imply the similarity among vertices. This can be achieved through similarity measures, such as Euclidean distance, cosine similarity, or correlation coefficient. We denote the similarity function between $\mathbf{x}_i$ and $\mathbf{x}_j$ as $sim(\mathbf{x}_i, \mathbf{x}_j)$. Then, proximity graphs can be constructed by different rules. In this thesis, we take ϵ-Graph [43, 42] as an example, where for every data sample $\mathbf{x}_i$, an edge is connected to $\mathbf{x}_j$ if $sim(\mathbf{x}_i, \mathbf{x}_j) > \epsilon$. We define the weighted adjacency matrix as $W = (w_{ij})_{n \times n}$, where $w_{ij} = sim(\mathbf{x}_i, \mathbf{x}_j) \cdot \mathbb{I}(sim(\mathbf{x}_i, \mathbf{x}_j) > \epsilon)$, and $\mathbb{I}(\cdot)$ is an indicator function. With the proximity graph constructed, *ProxGraphRW* applies the Eq. (2.1) with $\vec{r} = \frac{1}{n}$, which means that the RW restart from any node with equal probability. The resulting vector $\vec{s} = (1 - \alpha)P\vec{s} + \frac{\alpha}{n}$ contains the connectivity scores of all nodes, where each element $\vec{s}(v)$ quantifies the overall similarity of node v to all other nodes. Finally, based on the hypothesis that anomalies have low connectivity to most others, the anomaly score of node v is

$$\mathcal{A}(v) = 1 - \vec{s}(v), \tag{3.3}$$

where $\vec{s}(v)$ is the element corresponding to node v in $\vec{s}$.

3.1.1.2 Threat Model

We consider a system consisting of two parties: an analyst who runs an RWAD algorithm to detect potential anomalies and an attacker who aims to evade the detection. In practice, the analyst would first collect data from the environment and construct a graph, which is fed into the RWAD system for anomaly detection. However, the attacker could tamper with the data collection process which will result in a poisoned graph, leading to the malfunction of the system. For instance, in online shopping platforms, the attacker may manipulate some users to provide fake ratings for target items. The resulting poisoned data can lead to biased recommendations from the recommender system.

We further introduce the threat model by specifying the attacker's knowledge, goal, and capability. By Kerckhoffs's principle, we assume a worst-case scenario where the attacker knows all the data as well as the anomaly detection model, which is a common assumption employed by many previous attacks [49, 18]. We assume that the attacker has a set of target nodes in mind. Initially, the target nodes would have been determined as abnormal by the RWAD system if no data was manipulated. The attacker then tries to decrease the anomaly scores of those target nodes in the hope that they would evade the detection. To this end, the attacker can manipulate the data constrained by a certain budget. Specifically, depending on whether the graph is directly accessible or not, we divide the attacks into two types:

- *Graph-space* attack: the attacker can directly modify the structure of the graph by adding and deleting the edges under a budget constraint K.
- *Feature-space* attack: the attacker can only modify the features of a set of attack nodes, which will indirectly cause changes in the graph structure. Considering a practical scenario that the targeted anomaly nodes are crafted to have specific malicious functions, we can not modify their features arbitrarily. Therefore, an indirect feature attack, aiming to decrease the anomaly scores of target nodes while keeping their features unchanged, is ideal for such a problem. Hence, we restrict the selection of attack nodes to those other than the target nodes.

3.1.1.3 Problem definition

To facilitate our theoretical analysis, we formally define the attacks against RWAD as follows.

Definition 1 (**PA-RWAD**: poisoning attacks against RWAD)**.** *An instance of the problem is defined by a tuple,* $(G, \mathcal{T}, \mathcal{A}, \Theta, K, \widehat{A}, \widehat{R})$*, where* $G = (V, E)$ *is a network,* $\mathcal{T} \subseteq V$ *is the set of targets,* $\mathcal{A} : \mathbb{G} \times V \to \mathbb{R}$ *is the anomaly score function,* $\Theta \in \mathbb{N}$ *is the safety threshold,* $K \in \mathbb{N}$ *is the budget specifying the maximum number of edges that can be added or removed,* $\widehat{A} \subseteq (V \times V) \setminus E$ *is the set of edges that can be added, and* $\widehat{R} \subseteq E$ *is the set of edges that can be removed. The goal is then to identify two sets,* $A^* \subseteq \widehat{A}$ *and* $R^* \subseteq \widehat{R}$*, such that* $|A^*| + |R^*| \leq K$*, and for* $G^* = (V, (E \cup A^*) \setminus R^*)$ *we have:*

$$\left|\left\{v_i \in V : \forall_{v_j \in \mathcal{T}} \mathcal{A}(G^*, v_i) > \mathcal{A}(G^*, v_j)\right\}\right| \geq \Theta.$$

In practice, the top-Θ nodes ranked by their anomaly scores in descending order are determined as anomalous. Then, the goal of **PA-RWAD** is to find a way of modifying the network by adding and removing edges, so that there are at least Θ nodes with anomaly scores greater than any of the target nodes. In other words, the target nodes are considered as benign.

We note that although **PA-RWAD** emphasizes modifying the structure of the graph, a feature-space attack is still an instance of **PA-RWAD**, since the modification of features will ultimately lead to the changes of the graph.

3.1.2 Complexity Analysis

We now proceed to analyze the computational complexity of the attacks against RWAD. We summarize the hardness results in Tab. 3.1.

Theorem 2. *The* ***PA-RWAD*** *problem is NP-hard given a directed graph.*

Proof. We will prove that the problem is NP-hard by showing a reduction from the NP-complete *3-Set Cover* problem. An instance of this problem is defined by a collection of subsets $Q =$

Table 3.1: Hardness results of **PA-RWAD**.

	Directed graph	Undirected graph
PA-RWAD	NP-hard **(Lemma 1 & Theom. 1)**	NP-hard **(Theom. 2)**

$\{Q_1, \ldots, Q_{|Q|}\}$ of the universe $U = \{u_1, \ldots, u_{|U|}\} = \bigcup_{Q_i \in Q} Q_i$ such that $\forall_i |Q_i| = 3$, and a number $k \in \mathbb{N}$. The goal is to determine whether there exist at most k elements of Q that cover the entire universe, i.e., $Q^* \subseteq Q$ such that $|Q^*| \leq k$ and $U = \bigcup_{Q_i \in Q^*} Q_i$.

Let (Q, k) be a given instance of the 3-Set Cover problem. We will now construct an instance of the **PA-RWAD** problem. In what follows, let $Q(u_i)$ be the subsets in Q that contain u_i, i.e., $Q(u_i) = \{Q_j \in Q : u_i \in Q_j\}$. Let us also assume that $|Q| \geq 4$, as all smaller instances can be easily solved in constant time. First, we construct a directed network $G_Q = (V, E)$, where:

- $V = U \cup \bigcup_{Q_i \in Q} \{Q_i, q_i, o_i\} \cup \{h_1, h_2, h_3\} \cup \bigcup_{u_i \in U} \bigcup_{j=1}^{|Q|-|Q(u_i)|} \{x_{i,j}, y_{i,j}, z_{i,j}\}$,
- $E = \bigcup_{u_i \in Q_j} \{(Q_j, u_i)\} \cup \bigcup_{o_i \in V} \bigcup_{h_j \in V} \{(o_i, h_j)\} \cup \bigcup_{x_{i,j} \in V} \{(x_{i,j}, u_i), (x_{i,j}, y_{i,j}), (x_{i,j}, z_{i,j})\}$.

An example of this construction (e.g., $|U| = 5$, $|Q| = 3$) is presented in Fig. 3.2. Now, consider

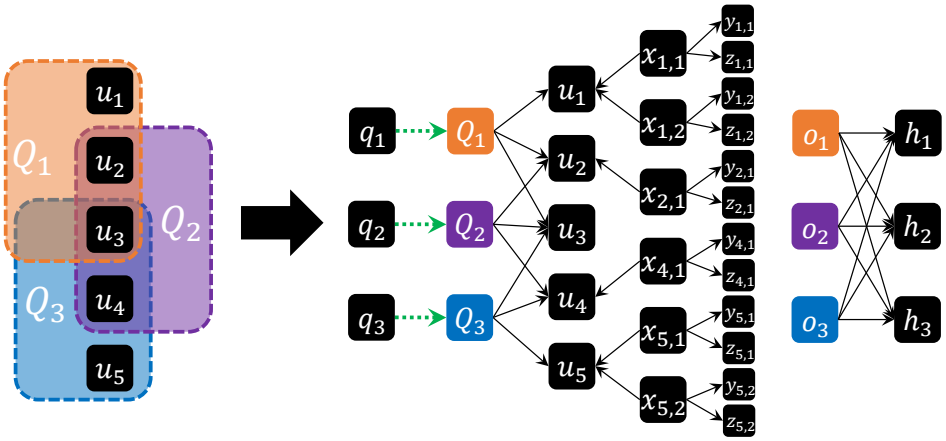


Figure 3.2: An example of the construction used in the proof of Theorem 2. The green dotted arrows represent edges that can be added.

the instance $(G_Q, \mathcal{T}, \mathcal{A}, \Theta, K, \widehat{A}, \widehat{R})$ of the **PA-RWAD** problem, where:

- G_Q is the network we just constructed,
- $\mathcal{T} = U$ is the target set,
- $\mathcal{A}$ is the anomaly score function with the restart rate parameter $\alpha = \frac{1}{|Q|}$,
- $\Theta = n - |U|$ is the safety threshold,
- $K = k$ is the budget,
- $\widehat{A} = \bigcup_{Q_i \in Q}\{(q_i, Q_i)\}$, i.e., only edges from q_i to corresponding Q_i can be added,
- $\widehat{R} = \emptyset$, i.e., none of the edges can be removed.

Since $\widehat{R} = \emptyset$, for any solution to the constructed instance of the **PA-RWAD** problem, we must have $R^* = \emptyset$. Hence, we will omit the mentions of R^* in the remainder of the proof, and we will assume that a solution consists just of A^*. We next prove a useful lemma.

Lemma 3. *Let $A \subseteq \bigcup_{Q_i \in Q}\{(q_i, Q_i)\}$, and let $G_Q \cup A = (V, E \cup A)$. We have that:*

$$\forall_{u_i \in U} \forall_{v \notin U} \mathcal{A}(G_Q \cup A, v) > \mathcal{A}(G_Q \cup A, u_i)$$

if and only if $\forall_{u_i \in U} \exists_{(q_j, Q_j) \in A} u_i \in Q_j$.

Proof. From the formula of the anomaly score function, we have that $\mathcal{A}(G_Q \cup A, v_i) = 1 - \vec{s}(G_Q \cup A, v_i)$, where:

$$\vec{s}(G_Q \cup A, v_i) = \frac{\alpha}{n} + (1 - \alpha) \sum_{v_j \in V} \vec{s}(G_Q \cup A, v_j) P_{j,i}.$$

Therefore, we have that $\mathcal{A}(G_Q \cup A, v_i) > \mathcal{A}(G_Q \cup A, v_j)$ if and only if $\vec{s}(G_Q \cup A, v_i) < \vec{s}(G_Q \cup A, v_j)$. Let $A(u_i)$ be the set of Q_j containing u_i that got connected to the corresponding node q_j via the edges in A, i.e., $A(u_i) = \{Q_j \in Q : u_i \in Q_j \wedge (q_j, Q_j) \in A\}$. We now compute the values of $\vec{s}(G_Q \cup A, v_i)$ for all nodes in V:

- $\vec{s}(G_Q \cup A, q_i) = \vec{s}(G_Q \cup A, x_{i,j}) = \vec{s}(G_Q \cup A, o_i) = \frac{\alpha}{n} = \frac{1}{|Q|n}$, as nodes q_i, $x_{i,j}$, and o_i do not have any predecessors,

- $\vec{s}(G_Q \cup A, y_{i,j}) = \vec{s}(G_Q \cup A, z_{i,j}) = \frac{\alpha}{n} + (1-\alpha)\vec{s}(G_Q \cup A, x_{i,j})\frac{1}{3} = \frac{\alpha}{n} + \frac{(1-\alpha)\alpha}{3n} = \frac{(4-\alpha)\alpha}{3n} = \frac{\left(4-\frac{1}{|Q|}\right)}{3|Q|n}$, as the only predecessor of nodes $y_{i,j}$ and $z_{i,j}$ is the node $x_{i,j}$ with out-degree 3,
- $\vec{s}(G_Q \cup A, h_i) = \frac{\alpha}{n} + (1-\alpha)\sum_{o_j \in V} \vec{s}(G_Q \cup A, o_j)\frac{1}{3} = \frac{\alpha}{n} + \frac{|Q|(1-\alpha)\alpha}{3n} = \frac{((1-\alpha)|Q|+3)\alpha}{3n} = \frac{|Q|+2}{3|Q|n}$, as the predecessors of h_i are all $|Q|$ nodes o_j, each with out-degree 3,
- if $(q_i, Q_i) \notin A$ then $\vec{s}(G_Q \cup A, Q_i) = \frac{\alpha}{n} = \frac{1}{|Q|n}$, as such node Q_i has no predecessors,
- if $(q_i, Q_i) \in A$ then $\vec{s}(G_Q \cup A, Q_i) = \frac{\alpha}{n} + (1-\alpha)\vec{s}(G_Q \cup A, q_i) = \frac{\alpha}{n} + (1-\alpha)\frac{\alpha}{n} = \frac{2|Q|-1}{|Q|^2 n}$, as the only predecessor of such node Q_i is the node q_i,
- $\vec{s}(G_Q \cup A, u_i) = \frac{\alpha}{n} + (1-\alpha)\sum_{Q_j \in Q(u_i)} \vec{s}(G, Q_j)\frac{1}{3} + (1-\alpha)\sum_{x_{i,j} \in V} \vec{s}(G, x_{i,j})\frac{1}{3} = \frac{\alpha}{n} + \frac{|Q|(1-\alpha)\alpha}{3n} + |A(u_i)|\frac{(1-\alpha)^2\alpha}{3n} = \frac{((1-\alpha)|Q|+3+|A(u_i)|(1-\alpha)^2)\alpha}{3n} = \frac{|Q|+2+|A(u_i)|\left(1-\frac{1}{|Q|}\right)^2}{3|Q|n}$, as the predecessors of u_i are $|Q(u_i)|$ nodes Q_j, as well as $|Q| - |Q(u_i)|$ nodes $x_{i,j}$, each with out-degree 3.

We now prove the main equivalence of the lemma. Assume that $\forall_{u_i \in U} \forall_{v \notin U} \mathcal{A}(G_Q \cup A, v) > \mathcal{A}(G_Q \cup A, u_i)$. In particular, it implies that: $\forall_{u_i \in U}\ \vec{s}(G_Q \cup A, u_i) - \vec{s}(G_Q \cup A, h_1) > 0$. By substituting the values in the inequality, we get:

$$\forall_{u_i \in U} \frac{|A(u_i)|\left(1-\frac{1}{|Q|}\right)^2}{3|Q|n} > 0,$$

which in turn implies that $\forall_{u_i \in U} |A(u_i)| > 0$. Hence, we have that for every $u_i \in U$ there exists at least one Q_j such that $u_i \in Q_j$ and $(q_j, Q_j) \in A$.

To prove the implication in the other direction, assume that $\forall_{u_i \in U} \exists_{(q_j, Q_j) \in A} u_i \in Q_j$. Hence, we get that $\forall_{u_i \in U} |A(u_i)| > 0$, which implies that: $\forall_{u_i \in U}\ \vec{s}(G_Q \cup A, u_i) \geq \frac{|Q|+2+\left(1-\frac{1}{|Q|}\right)^2}{3|Q|n}$. By comparing this value to the values computed above, we have that $\forall_{u_i \in U}, \forall_{v \notin U}$:

$$\vec{s}(G_Q \cup A, v) < \frac{|Q| + 2 + \left(1-\frac{1}{|Q|}\right)^2}{3|Q|n} \leq \vec{s}(G_Q \cup A, u_i),$$

which in turn implies that:

$$\forall_{u_i \in U} \forall_{v \notin U}\ \mathcal{A}(G_Q \cup A, v) > \mathcal{A}(G_Q \cup A, u_i).$$

This concludes the proof of the lemma. □

Let $Q^* \subseteq Q$ be a solution to the given instance of the 3-Set Cover problem, i.e., $|Q^*| \leq k$ and $\forall_{u_i \in U} \exists_{Q_j \in Q^*} u_i \in Q_j$. From Lemma 3 we have that $\mathcal{A}(G_Q \cup A^*, v) > \mathcal{A}(G_Q \cup A^*, u_i)$ where $A^* = \{(q_i, Q_i) : Q_i \in Q^*\}$. Hence, in network $G_Q \cup A^*$ all $\Theta = n - |U|$ nodes other than the nodes in U have greater anomaly scores than all the nodes in U, and $|A^*| \leq k = K$. Therefore, A^* is a solution to the constructed instance of the **PA-RWAD** problem. To prove the implication in the other direction, assume that A^* is a solution to the constructed instance of the **PA-RWAD** problem. In particular, it implies that $|A^*| \leq K = k$ and $\forall_{u_i \in U} \forall_{v \notin U} \mathcal{A}(G_Q \cup A, v) > \mathcal{A}(G_Q \cup A, u_i)$. From Lemma 3 we have that $\forall_{u_i \in U} \exists_{(q_j, Q_j) \in A} u_i \in Q_j$. Therefore, $\{Q_i \in Q : (q_i, Q_i) \in A^*\}$ is a solution to the given instance of the 3-Set Cover problem.

We have shown that the constructed instance of the **PA-RWAD** problem has a solution if and only if the given instance of the 3-Set Cover problem has a solution, which concludes the proof of NP-hardness. □

Theorem 4. *The **PA-RWAD** problem is NP-hard given an undirected graph.*

Proof. We will prove that the problem is NP-hard by showing a reduction from the NP-complete *Finding k-Clique* problem. An instance of this problem is defined by a network $G' = (V', E')$, and a number $k \in \mathbb{N}$. The goal is to determine whether there exist k nodes that induce a clique in G'.

Let (G', k) be a given instance of the Finding k-Clique problem. We will now construct an instance of the **PA-RWAD** problem. Let $n' = |V'|$, and let $d(G, v)$ be the degree of v in network G, i.e., $d(G, v) = |\{w \in V : (v, w) \in E\}|$. First, we construct a undirected network $G = (V, E)$, where:

- $V = V' \cup \{t\} \cup \bigcup_{v'_i \in V'} \bigcup_{j=1}^{n'+k-d(G',v')-3} \{x_{i,j}\}$,
- $E = E' \cup \bigcup_{v'_i \in V'} \{(t, v'_i)\} \cup \bigcup_{x_{i,j} \in V} \{(v'_i, x_{i,j})\}$.

An example of this (e.g., $|V'| = 4, k = 3$) construction is presented in Fig. 3.3. Now, consider the instance $(G, \mathcal{T}, \mathcal{A}, \Theta, K, \widehat{A}, \widehat{R})$ of the **PA-RWAD** problem, where:

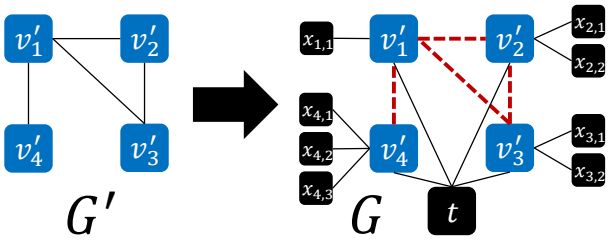


Figure 3.3: An example of the construction used in the proof of Theorem 4. The red dashed lines represent edges that can be removed.

- G is the network we just constructed,
- $\mathcal{T} = \{t\}$ is the target set,
- $\mathcal{A}$ is the anomaly score function with the restart rate parameter $\alpha = 0$,
- $\Theta = n - (n' - k + 1)$ is the safety threshold,
- $K = \frac{k(k-1)}{2}$ is the budget,
- $\widehat{A} = \emptyset$, i.e., none of the edges can be added,
- $\widehat{R} = E'$, i.e., only edges existing in G' can be removed from G.

Since $\widehat{A} = \emptyset$, for any solution to the constructed instance of the **PA-RWAD** problem, we must have $A^* = \emptyset$. Hence, we will omit the mentions of A^* in the remainder of the proof, and we will assume that a solution consists just of R^*.

From the formula of the anomaly score function with $\alpha = 0$ we have that $\mathcal{A}(G, v_i) = 1 - \vec{s}(G, v_i)$, where:

$$\vec{s}(G, v_i) = \sum_{v_j \in V} \vec{s}(G, v_j) P_{j,i}.$$

Therefore, we have that $\mathcal{A}(G, v_i) > \mathcal{A}(G, v_j)$ if and only if $\vec{s}(G, v_i) < \vec{s}(G, v_j)$.

Moreover, from Perra and Fortunato [141], we have that for the stationary distribution $\vec{s}$ of this form (i.e., for $\alpha = 0$) in an undirected network G we have that $\vec{s}(G, v_i) \sim d(G, v_i)$, i.e., the value of the entry in $\vec{s}$ for a given node is proportional to its degree. Therefore, we have that

$\mathcal{A}(G, v_i) > \mathcal{A}(G, v_j)$ if and only if $d(G, v_i) < d(G, v_j)$. Let us now compute the values of $d(G, v_i)$ for all nodes in G:

- $d(G, t) = n'$, as the node t is connected with all n' nodes v'_i,
- $d(G, x_{i,j}) = 1 < d(G, t)$, as each node $x_{i,j}$ is only connected with the node v'_i,
- $d(G, v'_i) = 1 + d(G', v'_i) + n' + k - d(G', v'_i) - 3 = n' + k - 2 \geq d(G, t)$, as each node v'_i is connected with the node t, $d(G', v'_i)$ nodes from V', as well as $n' + k - d(G', v'_i) - 3$ nodes $x_{i,j}$.

Since $\Theta = n - (n' - k + 1)$, all nodes $x_{i,j}$ have a smaller degree than t, and the total number of $x_{i,j}$ is $n - n' - 1$, we need at least k out of n' nodes in V' to have a smaller degree than t in order for the safety threshold to be satisfied. However, they all have equal or greater degrees than t. Hence, the safety threshold is not satisfied in G.

Since the removal of edges from $\widehat{R}$ can only change the degrees of nodes in V', we need to decrease the degree of k of these nodes to a value smaller than that of t. For each of these k nodes we have to remove at least Δ edges incident with it, where:

$$\Delta = d(G', t) - d(G', v'_i) + 1 = n' + k - 2 - n' + 1 = k - 1.$$

Let $V^* \subseteq V'$ be a solution to the given instance of the Finding k-Clique problem, i.e., a set of k nodes forming a clique in G'. Since $\widehat{R} = E'$ and the degree of each node in k-clique is $k - 1$, we have that $V^* \times V^* \subseteq \widehat{R}$, and removing $V^* \times V^*$ from G decreases the degree of k nodes from V' by $\Delta = k - 1$ each. Therefore, $V^* \times V^*$ is a solution to the constructed instance of the **PA-RWAD** problem.

To prove the implication in the other direction, assume that R^* is a solution to the constructed instance of the **PA-RWAD** problem. At least $\frac{k\Delta}{2} = \frac{k(k-1)}{2}$ of the removed edges have to be incident with the k nodes from V' contributing to the safety threshold. However, since the total budget is $K = \frac{k(k-1)}{2}$, all of the removed edges have to be incident with the k nodes from V' contributing to the safety threshold, and $\frac{k(k-1)}{2}$ edges incident with k nodes constitute

a clique. Since we have that $\widehat{R} = E'$, the same edges constitute a k-clique in G'. Therefore, $\bigcup_{(v'_i,v'_j)\in R^*}\{v'_i, v'_j\}$ is a solution to the given instance of the Finding k-Clique problem.

We have shown that the constructed instance of the **PA-RWAD** problem has a solution if and only if the given instance of the Finding k-Clique problem has a solution, which concludes the proof of NP-hardness. □

3.1.3 Practical Graph-Space Attacks

In this section, we investigate practical attacks in the graph space. We note that the graph-space attack itself is important in the case where the graph is directly accessible. Moreover, as we will show later, the results of graph-space attacks provide insightful guidance for feature-space attacks.

3.1.3.1 Attack Formulation

We begin by formulating the decision problem **PA-RWAD** as an optimization problem. We use $G = (V, E)$ with its corresponding adjacency matrix W to represent the original clean graph. We assume that the anomaly detection system predicts node v as an anomaly if the anomaly score $\mathcal{A}(v)$ is greater than a threshold θ. The attacker aims to decrease the number of nodes in a given target set $\mathcal{T} \subset V$ that are identified as anomalies by modifying at most K edges in the graph. To represent the edge manipulations, we denote the modification by a binary matrix $B = (b_{uv})_{(|V|\times|V|)}$, where the element $b_{uv} \in \{0, 1\}$. If $b_{uv} = 0$, the edge $\langle u, v\rangle$ remains unchanged, and $b_{uv} = 1$ lead to add/delete of edge $\langle u, v\rangle$. Then the attack graph can be represented by $|W - B|$. In this section, we consider undirected graphs where the adjacency matrix is always symmetric, and the budget constraint can be represented as $\sum_{u>v} b_{uv} \leq K$. Then the graph-space attack problem can be formulated as follows:

$$\begin{aligned} \min_{B} \quad & \textstyle\sum_{v\in\mathcal{T}} \mathbb{I}(\mathcal{A}(v) > \theta), \\ \text{s.t.} \quad & b_{uv} \in \{0, 1\}, \textstyle\sum_{u>v} b_{uv} \leq K, \end{aligned} \tag{3.4}$$

where $\mathbb{I}(\cdot)$ is a indicator function, $\mathbb{I}(\mathcal{A}(v) > \theta) = 1$ if the anomaly scores of node v is greater than θ.

3.1.3.2 Attack Method

To address the non-differentiable issue of the binary values in B, we adopt a relaxation strategy by representing b_{uv} in a continuous space that ranges from 0 to 1. This is denoted as $\tilde{B}$, which is subsequently converted back to binary form $\bar{B}$ after solving the optimization problem. To handle the discrete objective function in Eq. (3.4), we replace it with the sum of anomaly scores among target nodes, $\mathcal{L}_a(\tilde{B}) = \sum_{v\in\mathcal{T}} \mathcal{A}(v)$, then we can re-formulate the attack problem as:

$$\begin{aligned} \min_{\tilde{B}} \quad & \mathcal{L}_a(\tilde{B}) = \textstyle\sum_{v\in\mathcal{T}} \mathcal{A}(v), \\ \text{s.t.} \quad & \tilde{b}_{uv} \in [0,1],\ \textstyle\sum_{u>v} \bar{b}_{uv} \leq K, \end{aligned} \tag{3.5}$$

where $\tilde{B}$ is the relaxed and continuous adjacency matrix, $\bar{B} = (\bar{b}_{ij})$ is the discrete version of $\tilde{B}$.

To solve the challenging bi-level optimization problem, we propose two strategies: alternative iteration attack (**alterI**-attack) and closed-form attack (**cf**-attack). In brief, the **alterI**-attack iterates the inner RW model and the attack optimization alternatively to approximate the bi-level optimization, while the **cf**-attack transforms the bi-level optimization into a single-level problem. We first introduce the **alterI**-attack and then highlight the difference in the **cf**-attack.

1) alterI-attack. The optimization of problem (3.5) remains a challenging task due to the need to reverse the continuous variable $\bar{B}$ to binary $\bar{B}$ while satisfying the budget constraint. To overcome this difficulty, we first use projected gradient descent (PGD) to efficiently optimize $\tilde{B}$ without considering the budget constraint $\sum_{u>v} \bar{b}_{uv} \leq K$. Instead, we add l_2-norm regularization on the variable $\tilde{B}$: $\mathcal{L}_a(\tilde{B}) = \sum_{v\in\mathcal{T}} \mathcal{A}(v) + \lambda||\tilde{B}||_2$, where the λ is regularization coefficient. Then, we obtain the binary matrix $\bar{B}$ by selecting the top-K elements from $\tilde{B}$. This approach allows us to efficiently approximate the constrained optimization problem while ensuring that the attack budget is satisfied. The advantage of our optimization strategy is that the continuous solution $\tilde{B}$ we obtained does not depend on the attack budget K. This implies that we can reuse the same $\tilde{B}$ for various K, eliminating the need for recalculations.

However, optimizing the relaxed optimization problem is still challenging because the anomaly score $\mathcal{A}(v)$ in the loss function $\mathcal{L}_a(\tilde{B})$ depends on the variable $\tilde{B}$ in a complex way. After updating $\tilde{B}$, obtaining $\mathcal{A}(v)$ requires iterating over Eq. (2.1) dozens of times to get the converged node similarity vector $\vec{s}$, and the gradient needs to be traced back to each iteration. To address this issue, we only iterate over Eq. (2.1) once instead of multiple times. The detailed procedures are summarized in Alg. 1 and Fig. 3.4 (top). Firstly, we update the adjacency matrix with $\tilde{W} = |W - \tilde{B}|$ (line:5), and then we update the similarity score $\mathcal{A}(v)$ based on $\tilde{W}$ for one step using Eq. (2.1) and then obtain the anomaly score with Eq. (3.2) or 3.3 (line:6-9). Next, we update attack loss $\mathcal{L}_a(\tilde{B})$ based on $\mathcal{A}(v)$ (line:10), and calculate the projected gradient to optimize $\tilde{B}$ for one step (line: 11-15). Repeating the alternative iteration leads to the convergence of the inner model $\vec{s}$ and also the continuous attack variable $\tilde{B}$. After the iterations, we keep the top-K elements in $\tilde{B}$ to obtain $\bar{B}$ and the others are set to zeros (line:17-18). Finally, the attacked graph is obtained by $\hat{W} = |W - \bar{B}|$ (line:19). This algorithm is also suitable for weighted graphs in which the weights on edges are in $[0, 1]$, and the final solution is to modify K edge weights while the other weights remain unchanged.

2) cf-attack. While the **alterI**-attack approach is feasible, the one-step update of the inner model is a simple estimation that may not provide accurate attack loss during the iteration. To address this issue and obtain accurate attack loss, we employ the closed-form solution of the inner model to transform the bi-level optimization problem into a single-level problem. According to [142, 143], the inner model (Eq. (2.1)) has closed-form solution as follows:

$$\vec{s} = \alpha(I - (1 - \alpha)P)^{-1}\vec{r}, \tag{3.6}$$

where I is an identity matrix. With the closed-form solution, we can directly obtain the accurate anomaly scores after the update of $\tilde{B}$. In contrast to the **alterI**-attack, which iterates the inner model once after updating $\tilde{B}$, our innovative **cf**-attack approach substitutes the Eq. (2.1) (line:7) with Eq. (3.6) to obtain the accurate connectivity scores $\vec{s}$ for current $\tilde{B}$, and others remain the same.

Algorithm 1 Graph-space attack.

Input: Graph with adjacency matrix W, attack budget K, attack iteration T, learning rate η.
Output: Attacked graph with adjacency matrix $\hat{W}$.
function AlterI-attack(W, K, T, η)
    for $t = 1$ to T **do**
        Update adjacency matrix: $\tilde{W} = |W - \tilde{B}|$.
        for each node v in target set $\mathcal{T}$ **do**
            Update similarity scores $\vec{s}$ with Eq. (2.1).
            Update anomaly score $\mathcal{A}(v)$ based on $\vec{s}$.
        end for
        Update objective function $\mathcal{L}_a(\tilde{B})$ with $\mathcal{A}(v)$.
        for each edge $\tilde{b}_{uv}$ in $\tilde{B}$ **do**
            Calculate gradient $g_{uv} = \tilde{b}_{uv} - \eta \frac{\partial \mathcal{L}(\tilde{B})}{\partial \tilde{b}_{uv}}$
            Project g_{uv} into $[0, 1]$
            Update $\tilde{b}_{uv}$ in $\tilde{B}$
        end for
    end for
    Choose top-K edges in $\tilde{B}$ to obtain $\bar{B}$:

$$\bar{b}_{uv} = \begin{cases} \tilde{b}_{uv} & \text{if } \tilde{b}_{uv} \in top_K(\tilde{B}), \\ 0 & \text{otherwise.} \end{cases}$$

    Obtain attacked graph $\hat{W} = |W - \bar{B}|$.
    return $\hat{W}$.
end function

3.1.3.3 Complexity Analysis

While **cf**-attack offers a more accurate formulation than **alterI**-attack, it comes with the cost of potential time consumption when calculating the matrix inverse, particularly for graphs with a

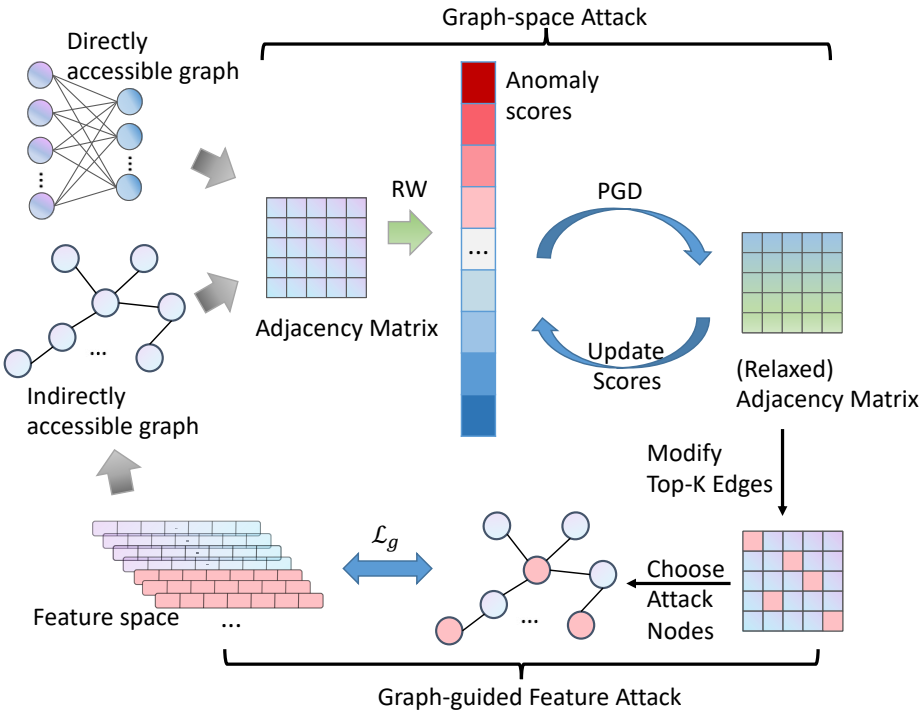


Figure 3.4: Illustration of proposed attacks.

large number of nodes or edges. In contrast, **alterI**-attack does not encounter such a problem, making it a more efficient option for such scenarios. Both **cf**-attack and **alterI**-attack have their own unique advantages.

Our **alterI**-attack has the complexity of $O(T(|E|+2|V|^2))$. First, we need to update the anomaly scores by Eq. (2.1), in which the time complexity is $O(|E|)$ because it transits through all edges in the graph. Our loss function includes a l_2-norm on the variable $\tilde{B}$, which requires $O(|V|^2)$ computation. Then, we take the gradient for each element in $\tilde{B}$, whose complexity is $O(|V|^2)$. We repeat the process for T steps, then the total complexity for **alterI**-attack is $O(T(|E| + 2|V|^2))$. For **cf**-attack, we update the anomaly score by the Eq. (3.6), which takes the complexity of $O(|V|^{2.21})$ [144] for sparse matrix inverse. Hence, the total complexity is $O(T(|V|^{2.21}+|E|))$.

3.1.4 Graph-Guided Feature-Space Attacks

3.1.4.1 Motivation for Feature-space Attacks

Previously, we presented effective graph-space attacks against Di-RWAD. However, for InDi-RWAD, where the graphs are not directly accessible, the *realizability* of the attacks becomes a serious concern: the attacker cannot directly modify the edges in a virtual graph space. Instead, in many practical application scenarios, what the attacker can modify are the attributes associated with the entities in their control. For example, when it comes to network intrusion detection, each TCP connection represents an entity or node, and attackers can manipulate certain TCP connections by altering attributes such as connection duration, protocol type, and the number of urgent packets. Such manipulations will change the structure of the proximity graph in the *ProxGraphRW* model to become perturbed, which can help shield the targeted anomaly TCP connection from being detected.

Thus, investigating feature-space attacks against InDi-RWAD is of significant practical importance. In particular, we consider the scenario where an attacker can manipulate a set of entities (corresponding to nodes in the constructed proximity graph) and modify their features to assist a group of target nodes in avoiding detection. We explore the connection between graph-space and feature-space attacks and demonstrate how guidance from graph-space attacks can be leveraged to construct effective feature-space attacks.

3.1.4.2 Attack Formulation

Consider a set of entities with features $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$, where $\mathbf{x}_i \in \mathcal{X}$ denotes the feature vector associated with entity i. As introduced in Section 3.1.1.1, a proximity graph can be constructed from $\mathbf{X}$, where the nodes represent those entities and edges indicate similar node pairs. An attacker aims to allow a set of target entities (nodes) $\mathcal{T}$ to evade detection. We assume that the attacker has control of a set of *attack nodes* $\mathcal{Z}$ such that the features of the nodes in $\mathcal{Z}$ can be arbitrarily modified in a certain domain $\mathcal{X}$. To limit the attacker's ability, we make the restriction that $\mathcal{Z} \cap \mathcal{T} = \emptyset$ and $|\mathcal{Z}| \leq K'$. For an attack node $i \in \mathcal{Z}$, we denote the modified

feature vector as $\hat{\mathbf{x}}_i$. The manipulated feature matrix is $\hat{\mathbf{X}}$. We note that since the manipulation of the features leads to the change of graph structure, the anomaly score function $\mathcal{A}(v; \hat{\mathbf{X}})$ depends on the features $\hat{\mathbf{X}}$. Then, we can formulate the feature-space attack as follows:

$$\begin{aligned} \min_{\hat{\mathbf{x}}_i, i \notin \mathcal{T}} \quad & \mathcal{L}(\hat{\mathbf{X}}) = \textstyle\sum_{v \in \mathcal{T}} \mathbb{I}(\mathcal{A}(v; \hat{\mathbf{X}}) > \theta), \\ \text{s.t.} \quad & \hat{\mathbf{x}}_v = \mathbf{x}_v, \forall v \in \mathcal{T},\ \hat{\mathbf{x}}_i \in \mathcal{X}, \\ & \mathcal{Z} = \{i | \hat{\mathbf{x}}_i \neq \mathbf{x}_i\},\ |\mathcal{Z}| \leq K'. \end{aligned} \tag{3.7}$$

3.1.4.3 Two Levels of Guidance from Graph-Space Attacks

Applying the gradient-descent method to solve problem (3.7) faces a crucial challenge: while gradient descent can be used to optimize the node features, it is hard to decide which nodes are to be manipulated. In other words, it is nontrivial to guarantee the constraint $|\mathcal{Z}| \leq K'$ while preserving optimization performance. We adopt a divide-and-conquer strategy to tackle this problem: we first select up to K' nodes as the attack nodes and then utilize gradient descent to optimize node features. In particular, we show that the results from graph-space attacks can be innovatively utilized to guide both the selection of attack nodes and feature optimization.

Specifically, given a proximity graph $\mathcal{G}$, we can leverage the attacks in Section 3.1.3 to produce a poisoned graph $\mathcal{G}'$. Even though $\mathcal{G}'$ might not be directly realized, it represents an excellent candidate in the graph space with which the target nodes $\mathcal{T}$ could evade detection with high probability. Thus, our intuition is to manipulate features so that the resulting proximity graph would approximate $\mathcal{G}'$. To this end, we utilize the guidance from the following two aspects.

Guidance on attack node selection. In the graph-space attack, the nodes involved in the structure modification might have a more significant impact on the attack goal. We denote the set of edges/non-edges modified by the attacker as $\mathcal{E}_a$. Intuitively, the modification of $\mathcal{E}_a$ will influence the anomaly scores of the targets most. To preserve such an influence, we set the attack nodes $\mathcal{Z}$ as those ones incident to the edges/non-edges in $\mathcal{E}_a$. Note that we can always easily adjust the budget in the graph-space attack such that the constraint $|\mathcal{Z}| \leq K'$ is satisfied.

After fixing the attack nodes $\mathcal{Z}$, we can follow a similar approach in the graph-space attack to optimize the features. Specifically, we replace the indicator function in (3.7) with the sum of anomaly scores of target nodes. For discrete features, we relaxed their discrete feature domain to the continuous space denoted by $\tilde{\mathcal{X}}$. Then, let $\tilde{\mathbf{x}}_i \in \tilde{\mathcal{X}}$ denote the relaxed feature, and $\tilde{\mathbf{X}} = \{\tilde{\mathbf{x}}_i | i \in V\}$, the feature-space attack can be formulated as the following optimization problem:

$$\min_{\hat{\mathbf{x}}_i \in \tilde{\mathcal{X}}, i \in \mathcal{Z}} \mathcal{L}_a(\tilde{\mathbf{X}}) = \sum_{v \in \mathcal{T}} \mathcal{A}(v; \tilde{\mathbf{X}}). \tag{3.8}$$

We term this type (with objective function $\mathcal{L}_a$) of feature-space attacks as **G-Guided**. We can straightforwardly adopt the two algorithms **alterI-attack** and **cf-attack** to solve the optimization problem (3.8), resulting in two variants named **G-Guided-alterI** and **G-Guided-cf**.

Guidance on reformulation of attack objective. Beyond the selection of attack nodes, the poisoned graph $\mathcal{G}'$ obtained from the graph-space attack can provide vital information for optimizing the features. Specifically, we aim to optimize the features such that the proximity graph constructed from the modified features would approximate $\mathcal{G}'$ as much as possible. To this end, we reformulate the attack objective function as follows:

$$\mathcal{L}_g(\tilde{\mathbf{X}}) = \sum_{\substack{\{(i,j)|\hat{b}_{ij}>0\} \\ i/j \in \mathcal{Z}}} |sim(\mathbf{x}_i, \mathbf{x}_j) - \hat{w}_{i,j}|, \tag{3.9}$$

where $\hat{w}_{ij}$ is the element in the attacked adjacency matrix $\hat{W}$. This objective function aims to push the similarity between control nodes $\mathbf{x}_i$ and other nodes $\mathbf{x}_j$ (denoted by $sim(\mathbf{x}_i, \mathbf{x}_j)$) close to the manipulated edges $\hat{w}_{i,j}$ in the poisoned graph $\mathcal{G}'$. Intuitively, minimizing $\mathcal{L}_g$ allows us to approximate an inverse problem: given $\mathcal{G}'$, find the node features from which $\mathcal{G}'$ can be constructed. Since (3.9) is a single-level function, we can directly adopt PGD (similar to the graph-space attack) to solve the optimization problem. We term this type (with objective function $\mathcal{L}_g$) of feature-space attacks as **G-Guided-plus**. The attack algorithm in the feature space is summarized in Alg. 2 and Fig. 3.4 (bottom).

The perturbed graph obtained from the graph-space attack serves as a valuable source of information. It not only highlights the crucial nodes that should be targeted by the feature-space

Algorithm 2 Feature-space attack.

Input: Feature matrix $\tilde{\mathbf{X}}$, attack nodes $\mathcal{Z}$, attack iteration T, learning rate η.
Output: Attacked feature matrix $\hat{\mathbf{X}}$.
function FeatureAttack($\tilde{\mathbf{X}}$, $\mathcal{Z}$, T, η)
    for $t = 1$ to T **do**
        Construct graph based on $\tilde{\mathbf{X}}$ (Section 3.1.1.1).
        Update similarity scores $\vec{s}$ with Eq. (2.1).
        Update the anomaly scores based on $\vec{s}$ (Eq. (3.3)).
        Update objective function $\mathcal{L}(\tilde{\mathbf{X}})$.
        for each attack nodes $\tilde{\mathbf{x}}_i, i \in \mathcal{Z}$ **do**
            Calculate gradient $g_i = \tilde{\mathbf{x}}_i - \eta \frac{\partial \mathcal{L}(\tilde{\mathbf{X}})}{\partial \tilde{\mathbf{x}}_i}$.
            Project $g_{i,j}$ into the feasible set $\tilde{\mathcal{X}}$.
            Update $\tilde{\mathbf{x}}_{i,j}$ in $\tilde{\mathbf{X}}$.
        end for
    end for
    Rounding the attacked feature:

$$\hat{\mathbf{x}}_i = \begin{cases} round(\tilde{\mathbf{x}}_{ij}) & \text{if feature } j \text{ is discrete,} \\ \tilde{\mathbf{x}}_{ij} & \text{otherwise.} \end{cases}$$

    return Attacked feature matrix $\hat{\mathbf{X}}$.
end function

attack but also suggests how the node features should be modified to maximize the impact on the anomaly score calculation.

3.1.5 Experiments

In this section, we evaluate the performances of our proposed attacks by answering these four major questions: 1) Are our proposed graph-space attacks effective? 2) What are the preferences

of the proposed graph attack? 3) How effective are the graph-guided feature-space attacks? 4) How is the transferability of the graph-guided feature-space attacks?

3.1.5.1 Datasets and Experiment Settings

We consider four datasets that are commonly used for graph-based anomaly detection: Paper-Author, Magazine, KDD-99, and MINIST (outlier). Among them, the first two are bipartite graphs while the latter two datasets are feature data. Below is the detailed description. **All datasets, source code for our proposed attacks, and evaluated baselines are in our *GitHub* link.** [1]

- Paper-Author [38]: This dataset contains papers crawled from the arXiv preprint database. Nodes U represent papers, while nodes V represent authors. An edge $\langle u, v\rangle$ indicates that the author v is shown in the paper u. We randomly sample $10,000$ records and delete nodes with degrees lower than 5, resulting in $|U|, |V| = 2311, 405$. We manually inject 10% of anomaly nodes following [66].

- Magazine: This dataset contains Amazon Reviews Data [2] under the category of Magazine Subscriptions. We randomly sample $100,000$ records and removed nodes with degrees lower than 3, resulting in $|U|, |V| = 1079, 1180$ nodes. We also inject 10% of anomaly nodes manually following [66].

- KDD-99 [43]: The dataset contains network intrusion data with 41 features and 4 types of attacks. We randomly sample $10,000$ benign data and 100 anomaly data for the experiment.

- MINIST (outlier): This is a subset of the MINST handwritten digits dataset, created for the outlier detection task in Outlier Detection DataSets [3]. It contains a total of 7603 images,

[1] https://github.com/Yuni-Lai/CoupledAttackRW.

[2] https://nijianmo.github.io/amazon/, accessed May 2023.

[3] http://odds.cs.stonybrook.edu/, accessed May 2023.

with 6903 images of digit-0 regarded as normal points and 700 images of digit-6 regarded as outliers. Each sample has 100 features.

3.1.5.2 Experimental Settings

We conduct our experiments on Ubuntu 20.04 system with an NVIDIA GeForce RTX 3090 GPU, Python 3.7, and PyTorch $1.10.0$. All the experiments are repeated 10 times with different random seeds, and different target nodes are sampled.

1) Target nodes and budgets. For attacking *BiGraphRW* model, we sample 5 target nodes from the top 100 anomaly nodes, while in *ProxGraphRW* model, we sample 20 target nodes from the top 100 anomaly nodes. We set the attack edge budget proportion to the sum of *target node degrees* (e.g., budget 10% : $K = 0.1 \times \sum_{v \in \mathcal{T}} d(v)$, where $d(v)$ is the degree of node v). Setting the budget associated with node degree is commonly adopted in targeted attacks such as Nettack [19, 75]. In feature-space attacks, we set the number of attack nodes $\mathcal{Z}$ as the number of nodes involved in the **alterI** graph-space attack. Then the attack intensity is associated with the attack budget in the graph space attack.

2) Evaluation metrics. Our main focus is to evaluate the effectiveness of our proposed method facilitating target nodes to evade detection under different detection thresholds. Usually, the detection threshold θ is set to the proportion of data size, and we evaluate the level of detect ratio as the top 5% and 10%. We then use the *evasion rate* ER of target nodes under these detection thresholds as the main metric. Specifically, the evasion rate is computed as $\mathsf{ER} = n_0/|\mathcal{T}|$, where n_0 is the number of target nodes not shown in the top 5% or 10% anomaly scores (i.e., evaded successfully). Besides, we also evaluate the average anomaly scores of target nodes.

3) Baselines. We evaluate the effectiveness of our proposed attacks against several baselines for both graph-space attacks and feature-space attacks.

Graph-space attack. The most relevant prior work is [75]. Although this work also proposes a targeted attack for the RW model, it is specific to the DeepWalk model and cannot be directly applied to our RWAD systems. Therefore, we transfer its targeted attack to our model. Besides, we also adopt two common baselines RndAdd and DegAdd following [75].

- **RndAdd**: This baseline randomly adds candidate edges, where the candidate edges are the edges incident to target nodes.
- **DegAdd**: This baseline adds candidate edges with the top-K highest degrees, where the candidate edges are also the edges incident to target nodes.
- **DeepWalk**[75]: In this baseline, we transfer the attack designed for DeepWalk to RWAD models.
- Our methods: **alterI** and **cf** are our proposed attacks with alternative iteration and closed-form solution, respectively.

Feature-space attack. To evaluate the effectiveness of our graph-guided attack in node selection, we include random selection as a baseline for comparison.

- **VanillaOpt**: This baseline randomly selects attack nodes from candidates and optimizes node features with the objective function $\mathcal{L}_a(\tilde{\mathbf{X}})$ in (3.8) with strategy **alterI**.
- Our methods: We use the graph-space attacks to guide the selection of attack nodes and choose $\mathcal{L}_a(\tilde{\mathbf{X}})$ as the attack objective function, resulting in two attack methods **G-guided-alterI** and **G-guided-cf**, which adopt **alterI** and **cf** to optimize node features respectively. In addition, when the objective function $\mathcal{L}_g(\tilde{\mathbf{X}})$ (3.9) is selected, the attack method is **G-guided-plus**.

4) Hyper-parameters. Grid search is employed to find the optimal hyper-parameters in all the attack methods over different datasets. For *BiGraphRW* model, the regularization parameter $\lambda = 1 \times 10^{-6}$, learning rate $lr = 1.0$, 60 epochs with SGD optimizer. For *ProxGraphRW*

model, we evaluate proximity graphs constructed with cosine similarity ($Cos(\mathbf{x}_i, \mathbf{x}_j) = \frac{<\mathbf{x}_i, \mathbf{x}_j>}{||\mathbf{x}_i|| \, ||\mathbf{x}_j||}$) and correlation similarity ($Corr(\mathbf{x}_i, \mathbf{x}_j) = \frac{<\mathbf{x}_i - \bar{\mathbf{x}}_i, \mathbf{x}_j - \bar{\mathbf{x}}_j>}{||\mathbf{x}_i - \bar{\mathbf{x}}_i|| \, ||\mathbf{x}_j - \bar{\mathbf{x}}_j||}$). The similarity threshold ϵ for constructing the graph is 0.8 for the KDD-99 dataset and 0.5 for the MNIST dataset; We employed the regularization parameter $\lambda = 1 \times 10^{-4}$, learning rate $lr = 1.0$, 35 epochs for the KDD-99 dataset and 100 for the MNIST dataset with Adam optimizer in the graph-space attack. In feature-space attack, learning rate $lr = 1.0$, 500 epochs with Adam optimizer.

3.1.5.3 Performance of Graph-space Attacks

To begin with, we evaluate the performance of the target RWAD models over corresponding datasets. As shown in Tab. 3.2, both models achieved an AUC (area under reception curve) of at least 0.89, demonstrating a strong ability to identify anomalies.

Table 3.2: AUC of RWAD.

Models	*BiGraphRW*		*ProxGraphRW*	
Dataset	Author-Paper	Magazine	KDD-99	MNIST
AUC	1.00	0.89	0.98	0.90

1) Effectiveness of attacks. We present the evasion rates ER of those attack methods under different detection levels (top-$5\%/10\%$) in Tab. 3.3 and 3.4. We observe that our proposed graph attack methods, **alterI** and **cf**, significantly outperform other baselines on all datasets. For instance, at the detection level of top-5%, our results indicate that our proposed attack on *BiGraphRW* model is highly effective, achieving an evasion rate of over 85% with a budget of 40.0%. Similarly, for *ProxGraphRW* model, with a budget of 60.0%, the evasion rate (under detection threshold top-10%) is over 80% on the MNIST dataset. Since the MNIST dataset is relatively easier to attack, we report the attack performance at a higher detection threshold. The reason why the **DeepWalk** method does not exhibit a strong attack effect could be attributed to its transferability across different types of random walk models.

Table 3.3: Graph attack results on *BiGraphRW* model.

Dataset	Metrics	budget	RndAdd	DegAdd	DeepWalk	alterI	cf
Author-Paper	ER (5%)	0%	0.560	0.560	0.560	0.560	0.560
		20%	0.560	0.560	0.578	0.720	**0.760**
		40%	0.560	0.560	0.578	0.880	**0.940**
		60%	0.580	0.560	0.578	0.920	**0.960**
		80%	0.580	0.560	0.600	0.980	**1.000**
		100%	0.580	0.560	0.600	**1.000**	**1.000**
	ER (10%)	0%	0.000	0.000	0.000	0.000	0.000
		20%	0.000	0.000	0.000	0.060	**0.280**
		40%	0.000	0.000	0.000	0.260	**0.360**
		60%	0.000	0.000	0.000	**0.460**	0.360
		80%	0.000	0.000	0.000	**0.660**	0.600
		100%	0.000	0.000	0.000	**0.820**	0.740
Magzine	ER (5%)	0%	0.740	0.740	0.740	0.740	0.740
		20%	0.760	0.740	0.760	0.760	**0.780**
		40%	0.760	0.760	0.760	**0.880**	0.860
		60%	0.760	0.760	0.760	**0.920**	0.880
		80%	0.760	0.760	0.760	**0.960**	0.880
		100%	0.780	0.760	0.760	**0.980**	0.880
	ER (10%)	0%	0.380	0.380	0.380	0.380	0.380
		20%	0.380	0.380	0.380	0.500	**0.600**
		40%	0.400	0.380	0.380	0.560	**0.740**
		60%	0.400	0.380	0.400	0.620	**0.760**
		80%	0.400	0.380	0.400	0.760	**0.820**
		100%	0.400	0.400	0.400	0.840	**0.860**

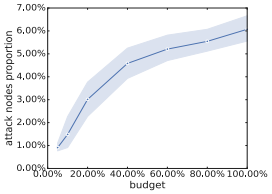
Comparing **alterI** and **cf** attack, it was observed that **cf** attack slightly outperforms **alterI** in most cases. In our experiments, we observe that **cf** can achieve significantly lower attack loss

Table 3.4: Graph attack results on *ProxGraphRW* model.

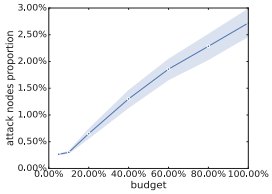
Dataset	Similarity	budget	RndAdd	DegAdd	DeepWalk	alterI	cf
KDD99 ER (5%)	cosine	0%	0.045	0.045	0.045	0.045	0.045
		10%	0.045	0.045	0.045	0.050	**0.055**
		20%	0.045	0.045	0.045	0.155	**0.245**
		40%	0.045	0.045	0.045	0.605	**0.620**
		60%	0.045	0.045	0.045	0.745	**0.825**
		80%	0.055	0.045	0.050	0.775	**0.865**
		100%	0.085	0.045	0.060	0.775	**0.875**
	correlation	0%	0.045	0.045	0.045	0.045	0.045
		10%	0.045	0.045	0.045	0.055	**0.060**
		20%	0.045	0.045	0.045	0.110	**0.150**
		40%	0.045	0.045	0.045	0.315	**0.405**
		60%	0.045	0.045	0.045	0.575	**0.690**
		80%	0.050	0.045	0.050	0.670	**0.735**
		100%	0.060	0.045	0.055	0.695	**0.845**
MNIST ER (10%)	cosine	0%	0.000	0.000	0.000	0.000	0.000
		10%	0.000	0.000	0.000	**0.060**	0.045
		20%	0.000	0.000	0.000	**0.210**	0.135
		40%	0.000	0.000	0.000	**0.585**	0.515
		60%	0.000	0.000	0.020	0.800	**0.860**
		80%	0.005	0.000	0.030	0.940	**0.975**
		100%	0.050	0.000	0.070	0.985	**0.995**
	correlation	0%	0.000	0.000	0.000	0.000	0.000
		10%	0.000	0.000	0.000	0.045	**0.060**
		20%	0.000	0.000	0.000	**0.205**	0.185
		40%	0.000	0.000	0.000	**0.555**	0.550
		60%	0.010	0.000	0.010	0.770	**0.825**
		80%	0.040	0.000	0.045	**0.940**	**0.940**
		100%	0.095	0.005	0.080	**0.995**	**0.995**

in the continuous domain (i.e., $\tilde{B}$). However, when discretizing the optimization results, the attack performance is not guaranteed to be preserved. While **cf** is generally more effective (also observed for feature-space attacks in Section 3.1.5.4), **alterI** is more efficient on larger graphs

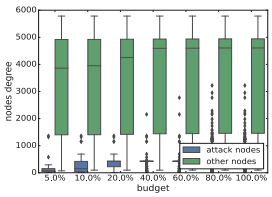
such as KDD-99 and MNIST (see Tab. 3.5).



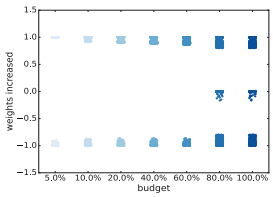
(a) Attack nodes (KDD-99).



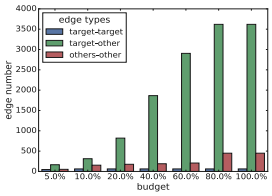
(b) Attack nodes (MNIST).



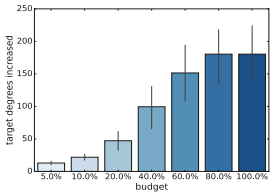
(c) The attacker prefers nodes with lower degrees.



(d) The weights changed by attacker.



(e) The attacker prefers adding edges between target nodes and the others.

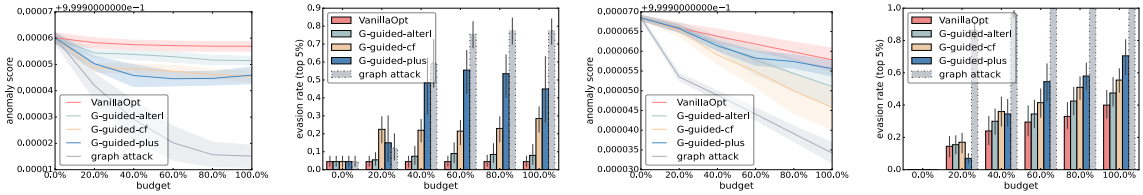


(f) The attacker increases the degree of target nodes.

Figure 3.5: Graph-space attack (**alterI**) result analysis on KDD-99.

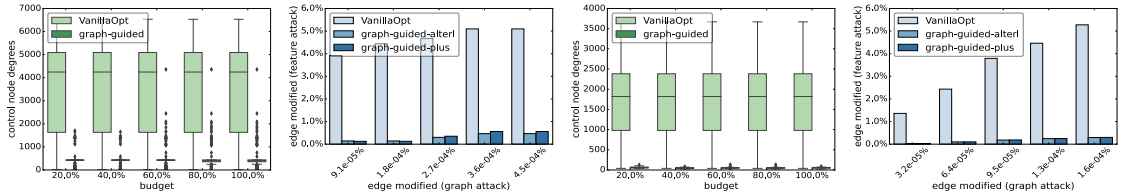
2) Preferences of graph attack. We further present a more detailed analysis of the graph attack results in Fig. 3.5, in which Fig. 3.5(a) and 3.5(b) show the proportion of the attacked nodes (to the total number of nodes) corresponding to different budgets. On average, only about $1\% - 6\%$ (KDD-99) and $0.3\% - 2.7\%$ (MNIST) of nodes are involved in the edge modification under various budgets (Fig. 3.5(b)). In Fig. 3.5(c), we present the node degrees of attack nodes and others, and we observe that the attacker prefers nodes with lower degrees as attack nodes. Fig. 3.5(d) presents the weights changed in the attack. We observe that the attacker tends to

make larger weight changes in the K attack edges. This is because we choose the top-K edges in priority of the values in $\tilde{B}$, and a higher value of $\tilde{b}_{ij}$ leads to larger weight change. The attacker mainly adds/deletes edges between target nodes and other nodes (Fig. 3.5(e)), and the target-other edge modification tends to increase the degree of target nodes (Fig. 3.5(f)). These actually provide convenience for our graph-guided feature attack with attack loss $\mathcal{L}_g(\tilde{\mathbf{X}})$, where the target node features are fixed (the edges between target-target are fixed) and the attack nodes can be optimized to be close to the desired edge weights (the edges between target nodes and control nodes). We observe similar phenomena in the MNIST dataset.



(a) KDD-99 (b) KDD-99 (c) MNIST (d) MNIST

Figure 3.6: Feature-space attack results.



(a) Control node degrees (KDD-99) (b) Edge modified (KDD-99) (c) Control node degrees (MNIST) (d) Edge modified (MNIST)

Figure 3.7: Result analysis of feature-space attacks.

3.1.5.4 Performances of Graph-guided Feature-space Attacks

First of all, to limit the attack intensity, we set the number of control nodes $|\mathcal{Z}|$ as the number of nodes involved in the graph-space attack. As mentioned before in Section 3.1.5.3, the number of nodes involved in the graph-space attack ranges from $1\% - 6\%$ in the KDD-99 dataset and $0.3\% - 2.7\%$ in the MNIST dataset with various edge manipulation budgets.

1) Effectiveness of attacks. We compare the performances of these feature-space attacks in Fig. 3.6. Our analysis shows that **G-guided-alterI** outperforms the **VanillaOpt** method, achieving much lower anomaly scores and higher evasion rates. These two models are only different in the selection of attack nodes, which indicates the effectiveness of using guidance from graph-space attacks in node selection. Comparing the performance of the **alterI** and **cf** attack strategies under $\mathcal{L}_a$, we observe that **cf**-attack also improves the performance, although the side effect is that **cf**-attack takes about 7 times longer than **alterI** in our experiments (Tab. 3.5). Additionally, **G-guided-plus** has a higher evasion rate than **G-guided-alterI** and **G-guided-cf** in most cases, indicating the advantage of using the attack loss $\mathcal{L}_g$ as further guidance for feature attack.

2) Unnoticeability of attack. In Fig. 3.7, we provide an analysis of the feature attack highlighting its advantage of unnoticeability. Specifically, we visualize the control nodes' original degree in Fig. 3.7(a) and Fig. 3.7(c). The manipulation in the feature space will then lead to the perturbation in the graph space. In Fig. 3.7(b) and Fig. 3.7(d), in order to quantify the perturbation volume, we present the ratio of edges modified by the feature-space attack on the y-axis and the ratio of edges modified by the graph-space attack x-axis. These ratios are proportionate to the original graph's total number of edges. As mentioned earlier, the graph attack prefers the attack nodes with lower degrees. As a result, our graph-guided attack nodes have lower node degrees compared to **VanillaOpt** (Fig. 3.7(a) and Fig. 3.7(c)). This leads to significantly fewer edge modifications in graph-guided attacks compared to **VanillaOpt** (Fig. 3.7(b) and Fig. 3.7(d)), which enhances the unnoticeability of the attack. In particular, both of our graph-guided attacks only lead to less than 0.5% edge modification in the graph space in both datasets.

Table 3.5: Runtime comparison of **alterI** and **cf**-attack.

	Attacks	Author-Paper	Magazine	KDD-99	MNIST
Graph	alterI	00:00:07	00:00:04	00:00:10	00:00:16
attack	cf	00:00:02	00:00:02	00:00:23	00:00:35
Feature	alterI	-	-	00:00:27	00:00:18
attack	cf	-	-	00:03:26	00:01:49

3.1.5.5 Transferability of graph-guided attack

We transfer our feature-space attacks to several unsupervised anomaly detection models, including Beta-VAE [145], IForest [146], and ECOD [147]. Tab. 3.6 shows the anomaly scores of target nodes before and after the transfer attack based on our **G-guided-alterI** and **G-guided-plus** feature attack on the KDD-99 dataset. The results indicate that the graph-guided attack with graph attack loss significantly decreases the anomaly scores of the target nodes across different models. This suggests that the graph-guided attack on RWAD has the potential to be used as a surrogate model for black-box attacks. The graph-guided attack could be a useful tool for attackers to evade detection and deceive anomaly detection systems in real-world scenarios.

Table 3.6: Transferability: The change in anomaly score (%) compared to the clean data. Lower is better.

Detect Methods	Attack Methods	20%	40%	60%	80%	100%
Beta-VAE	VanillaOpt	-11.56	-13.97	-14.70	-15.18	-15.68
	G-guided-alterI	-4.15	-5.65	-6.13	-7.14	-9.711
	G-guided-plus	**-25.26**	**-31.79**	**-33.25**	**-33.94**	**-33.99**
IForest	VanillaOpt	-10.63	-0.06	-8.41	-0.82	-11.93
	G-guided-alterI	9.94	10.44	-0.18	0.39	-2.31
	G-guided-plus	**-25.03**	**-44.21**	**-40.27**	**-47.58**	**-47.26**
ECOD	VanillaOpt	-2.20	-2.72	-2.90	-2.988	-3.099
	G-guided-alterI	-0.29	-0.64	-0.66	-0.90	-1.28
	G-guided-plus	**-3.70**	**-5.32**	**-5.81**	**-6.01**	**-6.00**

3.1.6 Limitation and future work

Although our thesis introduces coupled-space attacks for RWAD and demonstrates the superior performance of our proposed methods compared to the baselines, we did identify certain limitations. Specifically, in Fig. 6, we observed that our feature-space attacks were not as effective

as our graph-space attacks. Additionally, the unnoticeability of feature attacks was found to be comparatively weaker than graph attacks. Fig. 7 indicates that the feature space attack resulted in a higher proportion of graph structure perturbation.

These limitations highlight areas for further investigation and improvement in future research. While our proposed coupled-space attacks offer significant advancements, addressing these limitations could potentially enhance the effectiveness and stealthiness of feature-space attacks in RWAD.

Generalization of coupled-space attack: In this study, we introduce coupled-space attacks against RWAD, where the interdependency between the graph space and the feature space is exploited to enhance the effectiveness of attacks. Besides RWAD, there are many other feature-derived graph models where the graph and feature are interdependent [148, 149]. For example, graph structure can be constructed based on traffic sensor data [150], earthquake sensor data [150], image data [151], video data [152], and genomics data [153]. Future work can generalize our proposed strategies to more feature-derived graph-based models in which the graph is constructed on raw features. Because the model directly relies on the graph, as long as the graph constructed on the perturbed feature is close to the perturbed graph, the attack is expected to be effective.

3.1.7 Conclusion

In conclusion, this section has shed light on the vulnerabilities of Random-Walk-based Anomaly Detection (RWAD), a classical and important anomaly detection tool. Specifically, we introduce a novel study of adversarial poisoning attacks on RWAD, where the graph is constructed on top of the feature space. We provide a theoretical understanding of these attacks, including proof of NP-hardness. Our approach involves proposing graph-space attacks and using the graph attack to guide the feature-space attack, which bridges the gap between these two attacks. Our experiments on four datasets, encompassing both directly and indirectly accessible graphs, demonstrate the effectiveness of our proposed graph-space attack and its ability to guide the selection of attack nodes and optimization of the attack loss for feature-space attacks. By taking RWAD as an example, our study provides valuable insights into the effectiveness of graph-space attacks

and feature-space attacks. Future research can extend this work to apply RWAD for black-box attacks on other deep learning-based anomaly detection systems, without relying on labeled data or inner models.

3.2 Adversarial Attack on Recommender System

Recommender systems (RS) are now considered an essential component of online shopping platforms like Amazon, Taobao, and eBay. By analyzing customers' historical shopping behaviors, including the items they have browsed, reviewed, or rated, RS can provide personalized product recommendations to potential customers who may be interested in them. A typical RS is built around a machine-learning algorithm that operates on a bipartite graph. Specifically, the graph comprises two sets of nodes representing users and items, and the edges between them indicate the ratings that users have given to items. To generate personalized recommendations, various graph analytic techniques, such as Matrix Factorization (MF) [55, 56] and Graph Neural Networks (GNNs) [154, 155], have been utilized to predict the missing ratings. Based on these predicted ratings, the RS recommends items to users that are likely to be of interest to them with higher predicted ratings resulting in higher recommendation priority.

Similar to other machine learning-based systems [156, 157], the adversarial robustness of RS has been a topic of significant research interest. One reason is that the predictions of RS are crucial for sellers to generate profits and for users to make informed decisions, making RS a tempting target for attackers. Additionally, it is relatively easy for attackers to manipulate the graph data that RS operates on in real-world scenarios. For example, attackers can inject fake user accounts with manipulated ratings to deceive the prediction results of an RS. In fact, node injection attacks [22, 23, 24] have emerged as the primary form of attacks, where fake nodes, called fraudsters, are injected with carefully crafted ratings to intentionally alter the recommendation outcomes.

To defend against the attacks, *GraphRfi* [54] was proposed to jointly train a fraudster detection and robust recommender. At a high level, *GraphRfi* innovatively attaches the GNN-based rec-

ommendation component with a fraudster detection component, which can produce the anomaly probability of each user. This probability is further used as the weight for that user in the training objective function for recommendation, such that users with a higher anomaly probability would have a lower contribution. *GraphRfi* trains both components jointly in an end-to-end manner, resulting in state-of-the-art performance for robust recommendation under node injection attacks. Overall, *GraphRfi* offers a promising way of integrating fraudster detection into recommendation to achieve robustness. However, our analysis shows that *GraphRfi* is still vulnerable to node injection attacks. The underlying reason, as we show in Section 3.2.2.3, is that the fraudster detection component relies on a supervised learning method, which in turn relies on the availability of initial user labels (i.e., fake or normal) to accurately detect anomalies. In practice, obtaining these true labels is extremely difficult, if not impossible. Even if unsupervised anomaly detection methods are used to preprocess the data and label users, the results may contain errors. This leads to noisy user labels, where some fake users are labeled as normal, causing *GraphRfi* to assign large weights to these fake users due to its supervised nature. As a result, GraphRfi may malfunction and not effectively detect fraudsters, rendering it vulnerable to node injection attacks.

Our first step is to conduct a thorough vulnerability analysis of fraudster-detection-based robust RS with *GraphRfi* as the representative. Specifically, we design a powerful node injection attack which is formulated as a bi-level combinatorial optimization problem. We utilize a gradient-based method to solve the problem, where one of the main challenges is to compute the required gradients. We adopt the idea of meta-gradients proposed by [19] that is designed to attack Graph Convolutional Networks (GCNs) [4] via manipulating the graph structure (i.e., add/delete edges). Different from the attack proposed in [19] that is designed for unweighted graphs, we not only need to decide the optimal injected edges between fake users and items but also the optimal rating associated with each edge. Our solution is to use a **continuous** *rating probability tensor* to encode all discrete ratings. After optimization, we use discretization techniques to recover the desired ratings. We term our attack as **metaC**. Our experiments show that **metaC** is very effective in promoting targeted items even with small budgets against the robust model *GraphRfi* as well as the MF-based model.

In summary, we propose a new attack method **metaC** that is tested effectively against both GNN-based and MF-based recommender systems. In particular, we conduct a detailed analysis of the causes of the vulnerability of a representative robust RS *GraphRfi*, providing insights for properly integrating fraudster detection into RS.

The rest of the section is organized as follows. We describe the problem statements in Section 3.2.1. In Section 3.2.2, we present our proposed attack method **MetaC**. We investigate how to generalize **MetaC** to the MF-based model in Section 3.2.3. We conduct extensive experiments in Section 3.2.4 to show the effectiveness of our proposed attack. Finally, we conclude our findings in Section 3.2.5.

3.2.1 Problem Statements

In this section, we introduce the adversarial environment in which a recommender system operates. Below, we specify the goal, knowledge, and ability of both the attacker and defender.

Threat Model. We consider an attacker whose goal is to promote a set of target items $\mathcal{T} \subset \mathcal{V}$. More specifically, the attacker aims to increase the probability that a target item $v_t \in \mathcal{T}$ appears in the top-k recommendation lists of target users. Based on Kerckhoffs's principle [158], we assume a worst-case scenario where the attacker has full knowledge of the target RS, including the data (i.e., the clean graph $\mathcal{G}$) and the recommendation algorithm. To achieve the malicious goal, the attacker is able to inject a set of fake users $\mathcal{U}'$ as well as some ratings (i.e., edges $\mathcal{E}'$ between $\mathcal{U}'$ and $\mathcal{V}$), resulting in a manipulated graph $\mathcal{G}' = (\mathcal{U} \cup \mathcal{U}' \cup \mathcal{V}, \mathcal{E} \cup \mathcal{E}')$. To constrain the attacker's ability, we assume that there are at most H fake users (i.e., $|\mathcal{U}'| \leq H$), and each fake user can give at most B ratings. After the attack, the defender observes the manipulated graph $\mathcal{G}'$, from which the RS is trained and tested; this attack falls into the category of data poisoning attacks.

3.2.2 Attacks against Existing RS

In this section, we use the GNN-based robust recommender system *GraphRfi* as the example to illustrate our attack. We show that our attack can be extended to MF-based RS in Section 3.2.3.

3.2.2.1 Attack Formulation

We begin by quantifying the attacker's malicious goal. Recall that the attacker aims to promote a set of target items $\mathcal{T}$, a commonly used metric to measure the effectiveness of attack for an item is the *hit ratio*. Specifically, a hit ratio for an item v with parameter k (denoted as $HR@k(v, \mathcal{G}, \theta)$) is the percentage of users whose top-k recommendation list includes that item. Note that we make explicit the dependency of the hit ratio of v on the graph $\mathcal{G}$ and the trained model parameter θ. Thus, we can use an adversarial objective function $F_{adv}(\mathcal{G}, \mathcal{T}, \theta) = \frac{1}{\mathcal{T}} \sum_{v \in \mathcal{T}} HR@k(v, \mathcal{G}, \theta)$, the average hit ratios of those target items, to quantify the attacker's goal.

Poisoning attacks against recommendation then amounts to finding the optimal poisoned graph $\mathcal{G}'$ to maximize the adversarial objective function. It can be formulated as a *bi-level* optimization problem, where in the outer level the attacker optimizes the objective over the graph $\mathcal{G}'$ while in the inner level, the model parameter θ is optimized though minimizing the training loss, also depending on $\mathcal{G}'$. Mathematically, a poisoning attack is formulated as:

$$\begin{aligned} \max_{\mathcal{G}'} \quad & F_{adv}(\mathcal{G}, \mathcal{T}, \theta^*) = \frac{1}{\mathcal{T}} \sum_{v \in \mathcal{T}} HR@k(v, \mathcal{G}', \theta^*) \\ \text{s.t.} \quad & \theta^* = \arg\min_{\theta} \mathcal{L}(\theta, \mathcal{G}'), \quad \mathcal{G}' = \mathcal{G} \cup \mathcal{U}' \cup \mathcal{E}', \\ & |\mathcal{U}'| \leq H, \quad d(u') \leq B, \quad \forall u' \in \mathcal{U}', \end{aligned} \tag{3.10}$$

where we use $\mathcal{G}' = \mathcal{G} \cup \mathcal{U}' \cup \mathcal{E}'$ to denote that $\mathcal{G}'$ is obtained by injecting a set of fake users $\mathcal{U}'$ and edges $\mathcal{E}'$ into $\mathcal{G}$, $|\mathcal{U}'| \leq H$ requires that at most H fake users are injected, and the degree constraint of fake user $d(u') \leq B$ requires that each fake user can give at most B ratings.

3.2.2.2 Attack Method

1) Reformulation of attack. The major challenges in solving the above optimization problem are the discrete search space and the exponential growth of candidate edges in $\mathcal{G}'$: the attacker needs to determine which items to rate as well as the specific discrete ratings (e.g., scale 1 to 5). We thus use a series of techniques to approximate this discrete optimization problem. First, we use a continuous probability vector $\hat{\mathbf{r}} = (p_1, p_2, \cdots, p_{r_{max}})$ (e.g., $r_{max} = 5$) to encode a discrete rating, denoting a user will give a rating $l \in \{1, 2, \cdots, r_{max}\}$ with probability p_l. Then, we assume that the injected users will initially connect to *all* items. Thus, the attacker's behavior is now fully captured by a continuous rating tensor $\hat{\mathbf{R}} \in [0, 1]^{|\mathcal{U}'|\times|\mathcal{V}|\times r_{max}}$. We denote the manipulated graph as $\hat{\mathcal{G}} = \mathcal{G} \cup \mathcal{U}' \cup \hat{\mathbf{R}}$. Another difficulty comes from the non-differentiability of the objective function, in particular, the hit ratios. Thus, we use a sum of $softmax$ function ratios [159] to approximate $F_{adv}(\mathcal{G}, \mathcal{T}, \theta^*)$ as below:

$$\mathcal{L}_{adv}(\hat{\mathcal{G}}, \mathcal{T}, \theta^*) = -\sum_{t\in\mathcal{T}}\sum_{u\in\mathcal{U}} \log(\frac{\exp(r'_{ut})}{\sum_{v\in\mathcal{V}} \exp(r'_{uv})}), \tag{3.11}$$

where r'_{uv} denotes the predicted rating from user u to v. Basically, this function measures the fraction of the ratings for targeted items over the ratings for all items. Now, the optimization problem defined in (3.10) is recast as:

$$\begin{aligned} \min_{\hat{\mathbf{R}}} \quad & \mathcal{L}_{adv}(\hat{\mathcal{G}}, \mathcal{T}, \theta^*) \\ s.t. \quad & \theta^* = \arg\min_{\theta} \mathcal{L}(\theta, \hat{\mathcal{G}}),\ \hat{\mathcal{G}} = \mathcal{G} \cup \mathcal{U}' \cup \hat{\mathbf{R}}, \\ & \sum_{l} \hat{\mathbf{R}}_{i,j,l} = 1(\forall i, j),\ \hat{\mathbf{R}} \in [0, 1]^{|\mathcal{U}'|\times|\mathcal{V}|\times r_{max}}. \end{aligned} \tag{3.12}$$

To utilize the continuous rating probability tensor $\hat{R}$, we design the loss function during attack optimization as below:

$$\mathcal{L}(\theta, \hat{\mathcal{G}}) = \sum_{\forall(u,v)\in\mathcal{E}} \max_{l} \hat{\mathbf{R}}_{uvl} \cdot (r'_{uv} - \arg\max_{l} \hat{\mathbf{R}}_{uvl})^2, \tag{3.13}$$

where the term $\arg\max_l \hat{\mathbf{R}}_{uvl}$ is used to find the maximum index of the probability vector as the ground truth, and $\max_l \hat{\mathbf{R}}_{uvl}$ is the associated maximum probability value. Rating vectors with higher maximum probability will have a higher contribution to the RS training loss during the attack optimization.

2) Optimization method. Now, we describe the method to solve problem (3.12) to obtain the (sub-)optimal continuous rating tensor $\hat{\mathbf{R}}$, from which we derive the discrete ratings that satisfy all the constraints. We alternately update the inner objective function $\mathcal{L}(\cdot)$ with respect to θ for K steps and update the outer objective function $\mathcal{L}_{adv}(\cdot)$ with respect to $\hat{\mathbf{R}}$ for one step, where K is a hyper-parameter. However, the central challenge lies in computing the gradients of $\mathcal{L}_{adv}(\cdot)$ with respect to $\hat{\mathbf{R}}$ because θ^* itself is obtained through an optimization process depending on $\hat{\mathbf{R}}$. We adapt the idea of *approximating meta gradients* [19] to compute the required gradients. In detail, we sum the gradients $\nabla_{\hat{\mathbf{R}}}\mathcal{L}_{adv}(\hat{\mathcal{G}}, \mathcal{T}, \theta^t)$ during the K steps of inner model updates as the approximation, termed as meta-gradients:

$$\nabla^{meta}_{\hat{\mathbf{R}}}\mathcal{L}_{adv} \approx \sum_{t=1}^{K} \nabla_{\hat{\mathbf{R}}}\mathcal{L}_{adv}(\hat{\mathcal{G}}, \mathcal{T}, \theta^t). \tag{3.14}$$

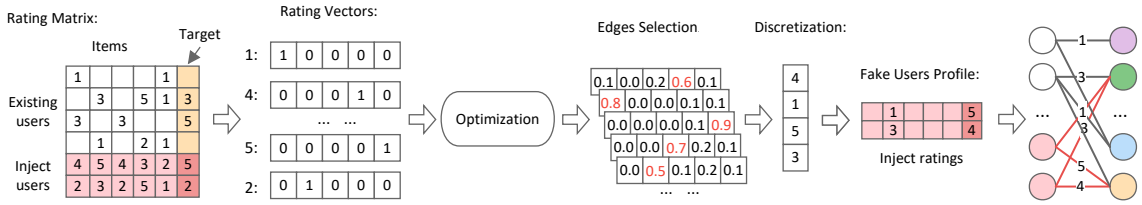


Figure 3.8: Illustration of poisoning attack **MetaC**. Fake users are pre-injected into a user-item graph (represented by a rating matrix). The discrete ratings are encoded by continuous rating vectors, optimized via gradient descent. Finally, a subset of these ratings is selected, and the discrete rating is determined by discretizing the corresponding rating vector.

As a result, we can update $\mathcal{L}_{adv}$ for one single step in the outer layer based on meta-gradients:

$$\hat{\mathbf{R}}^{t+1} = \hat{\mathbf{R}}^{t} - \eta_2 \cdot \nabla^{meta}_{\hat{\mathbf{R}}}\mathcal{L}_{adv}, \tag{3.15}$$

where η_2 is the learning rate. After each update of $\hat{\mathbf{R}}$, we conduct zero-one scaling and normalization to ensure that the entries of $\hat{\mathbf{R}}$ always stay within the range of $[0, 1]$ and sum to 1 while gradient updates might break these constraints. Specifically, we employ the commonly used 0-1 normalization method on all entries (for $\forall i, j$):

$$\hat{\mathbf{R}}_{i,j,:} = \frac{\hat{\mathbf{R}}_{i,j,:} - \min(\hat{\mathbf{R}}_{i,j,:})}{\max(\hat{\mathbf{R}}_{i,j,:}) - \min(\hat{\mathbf{R}}_{i,j,:})}, \hat{\mathbf{R}}_{i,j,:} = \frac{\hat{\mathbf{R}}_{i,j,:}}{\sum_{l=1}^{L}\hat{\mathbf{R}}_{i,j,:}}.$$

These operations ensure that the entries of $\hat{\mathbf{R}}$, which is $\hat{\mathbf{R}}_{i,j,:} \in \mathbb{R}^{r_{max}}$, can be interpreted as the probabilities of adding specific ratings during optimization. We can then iteratively update $\hat{\mathbf{R}}$ and θ until the loss $\mathcal{L}_{adv}$ diminishes to an acceptable level.

After the optimization, we will need to discretize the ratings and simultaneously pick B (budget) ratings for each fake user. Since the highest value of probabilities $\max_l \hat{\mathbf{R}}_{uvl}$ are served as weights in the training loss in Eq. (3.13), we discretize each rating vector $\hat{\mathbf{R}}_{ij}$ as (l, p_l) and pick top-B ratings by ranking the p_l in descending order as the injected ratings, where $p_l = \max_l \hat{\mathbf{R}}_{ij}$ and the discrete rating l is obtained by the corresponding index of p_l. In doing so, we have tackled the problem of simultaneously determining the optimal edges and ratings.

We further summarize the process of obtaining continuous rating tensor $\hat{\mathbf{R}}$ through alternate iteration in Algorithm 3, and Fig. 3.8 shows the framework of **MetaC**. Firstly, we initialize the $\hat{\mathbf{R}}$ by sampling ratings from the normal distribution $N(\mu, \sigma^2)$, where μ and σ^2 are the mean and variance of existing history ratings. Secondly, we do inner training (the RS model) for K steps with fixed $\hat{\mathbf{R}}$. Thirdly, we update $\hat{\mathbf{R}}$ by approximated meta-gradient $\nabla^{meta}_{\hat{\mathbf{R}}} \mathcal{L}_{adv}$ aiming to optimize adversarial attack loss $\mathcal{L}_{adv}$, and scale the vectors in $\hat{\mathbf{R}}$ to satisfy the probability constraints. We update the θ and $\hat{\mathbf{R}}$ alternately for T_{train} epochs.

Algorithm 3 MetaC Poisoning Attack.

Input: Initiated rating tensor $\hat{R}$; Total training epochs T_{train}; Inner training steps K.

for $t \leftarrow 0, \cdots, T_{train}$ **do**
  for $k \leftarrow 0, \cdots, K$ **do**
    $\theta^{(tK+k+1)} = \theta^{(tK+k)} - \eta_1 \nabla_{\theta^{(tK+k)}} \mathcal{L}(\theta^{(tK+k)}, \hat{\mathcal{G}})$.
  end for
  $\hat{R}^{t+1} = \hat{R}^t - \eta_2 \cdot \nabla^{meta}_{\hat{\mathbf{R}}} \mathcal{L}_{adv}$.
  $\hat{R}_{i,j,:} = \frac{\hat{R}_{i,j,:} - min(\hat{R}_{i,j,:})}{max(\hat{R}_{i,j,:}) - min(\hat{R}_{i,j,:})}$.
  $\hat{R}_{i,j,:} = \frac{\hat{R}_{i,j,:}}{\sum_{l=1}^{L} \hat{R}_{i,j,:}}$.
end forreturn The optimized rating tensor $\hat{R}$.

In summary, we formulate the node injection poisoning attack as a bi-level optimization problem

that is solved by an alternate iteration method in the continuous domain. Through discretization, we finally obtain the injected users associated with ratings, i.e., a poisoned graph $\mathcal{G}'$. We term this attack method as **MetaC**. We note that, by Kerckhoffs's principle in security, we consider a worst-case scenario when designing **MetaC**, where the attacker has full knowledge of the RS and some additional attack design goals, such as unnoticeability, are not considered. We then aim to design robust RS against this strong attack in the worst case.

3.2.2.3 Vulnerability analysis

The underlying reason that *GraphRfi* fails against our proposed poisoning attack is that its anomaly detection component adopts a *supervised* learning approach. As a result, if a user is labeled as normal (even if it is actually fake), supervised learning will eventually assign a small anomaly score to it as the training process continues. That is, fake users that are labeled as normal would still have strong malicious effects on the prediction.

We conduct comprehensive experiments to demonstrate this phenomenon. Specifically, we classify the users into four types. Type I and Type II users are normal and anomalous users inherently existing in the graph, respectively, and the defender knows their labels reliably. Among the injected fake users, a fraction of τ users (denoted as Type III) are determined as anomalous with high confidence by the defender and thus are labeled as abnormal. The rest of the fake users (denoted as Type IV) are labeled as normal. We emphasize that the parameter τ is a variable that reflects the defender's ability to identify abnormal users from the collected data, and in practice, it is not uncommon that τ is low especially when there are several effective stealthy node injection attacks against recommender systems. We observed the anomaly scores for those four types of users during the whole training process of *GraphRfi*, shown in Fig. 4.3(a). We can observe that Type II and Type III users (labeled as fake) have very high anomaly scores during training. In comparison, the anomaly scores of Type IV users (fake but labeled as normal) keep decreasing as training continues and eventually approach to those of Type I users (normal).

In summary, the insufficient ability of a defender to filter out fake users (which is common) resulted in highly noisy user labels and further caused the supervised anomaly detection com-

ponent to assign low anomaly scores (i.e., large weights of contribution) to *evaded* fake users, which finally left *GraphRfi* vulnerable to poisoning attacks. This crucial observation also guides us in designing robust RS in the next chapter.

3.2.3 Generalization to MF-based RS

In this section, we demonstrate that our attack and defense approaches can be applied to MF-based RS with minor modifications.

Different from the GNNs-based model, where $\mathcal{L}_{adv}(\hat{\mathcal{G}}, \mathcal{T}, \theta^t)$ is dependent on the input $\hat{\mathbf{R}}$ when given θ^t, the prediction of MF-based model only rely on $\theta^t = \{U, V\}$. This actually makes the computation of the gradients easier for MF-based RS due to its simplicity. Specifically, we can apply the meta-gradient (*instead of approximating*) to directly compute $\nabla^{meta}_{\hat{\mathbf{R}}}\mathcal{L}_{adv}$. Briefly, $\mathcal{L}_{adv}$ is depended on θ^K (obtain by K steps of inner model iteration), and each θ^{t+1} depends on $\hat{\mathbf{R}}$ and θ^t during the training, so the gradient $\nabla^{meta}_{\hat{\mathbf{R}}}\mathcal{L}_{adv}$ can be traced back to the each iteration of θ^t as follow,

$$\nabla^{meta}_{\hat{\mathbf{R}}}\mathcal{L}_{adv} = \nabla_{r'_{\theta^K}}\mathcal{L}_{adv}(\mathcal{T}, \theta^K) \cdot \nabla_{\theta^t} r'_{\theta^t}(\hat{\mathcal{G}}) \cdot \nabla_{\hat{\mathbf{R}}}\theta^K, \tag{3.16}$$

where $\nabla_{\hat{\mathbf{R}}}\theta^{t+1} = \nabla_{\hat{\mathbf{R}}}\theta^t - \eta_1 \nabla_{\hat{\mathbf{R}}}\nabla_{\theta^t}\mathcal{L}(\theta, \hat{\mathcal{G}}), t = 0, \cdots, K-1$, η_1 is the learning rate of the inner model, $r'_{\theta^t}(\hat{\mathcal{G}}) = \{U_u^T V_v | \forall (u, v) \in \hat{\mathcal{G}}\}$, are the rating predictions between among user-item pairs of the inner model.

The difficulty, however, lies in that it requires the loss function $\mathcal{L}(\theta, \hat{\mathcal{G}})$ in Eq. (3.13) is differentiable with respect to $\hat{\mathbf{R}}$. To address this, we smooth the arg max function via $softmax$ function. We approximate the arg max function by $softmax$ with parameter β (the larger β, the better approximation) as follow,

$$\arg\max_l \hat{\mathbf{R}}_{uvl} \approx [1, 2, 3, 4, 5]^T \times softmax(\beta \cdot \hat{\mathbf{R}}_{uv}). \tag{3.17}$$

Note that the attack optimized on the MF-based model is only different in the calculation of $\nabla^{meta}_{\hat{\mathbf{R}}}\mathcal{L}_{adv}$, and the other processes are the same.

3.2.4 Experiments

In this section, we aim to evaluate our methods by answering the following: Does the **MetaC** attack effectively compromise the security of existing robust recommendation systems?

3.2.4.1 Datasets and Experiment Settings

Datasets. We conduct experiments over two widely-used real-world datasets *YelpCHI* and *Amazon Movies&TV* (abbreviated as *Movies*) that collect user reviews from two platforms. *YelpCHI* contains approximately $60,000$ reviews/ratings regarding 201 restaurants and hotels in Chicago from $38,063$ reviewers. Each rating ranges from 1 to 5, and the corresponding review is provided with a label of *fake* or *normal*. In our setting, we treat a user giving *fake* review(s) as *fake*. The other dataset *Movies* contains reviews from Amazon under the category of Movie&TV. Each review, with a rating from 1 to 5, is voted *helpful/unhelpful* by other users, which provides the information to determine whether a user is fake or normal. Specifically, we only consider the reviews with more than 20 votes. If more than 70% of the votes of a review are *helpful*, we regard it as *normal*; otherwise, *fake*. Similarly to *YelpCHI*, we treat the users giving *fake* review(s) as *fake* users. The statistics of the two datasets are summarized in Tab. 3.7.

Table 3.7: Statistics of *YelpCHI* and *Movies*.

	# Users	# Items	# Edges	# Fake users
YelpCHI	38063	201	67395	7739
Movies	39578	71187	232082	19909

Note that there are two types of fake users, one is the existing fake users in the dataset (we do not know their targets/goals, but these users can enlarge the number of anomaly users for detection model training), and the other is our own injected fake user (from 0.3% to 2.0% according to attack power setting). During the data preprocessing, we iteratively remove the cold items and users that are less than two records. We extract the user features used in GNNs following [54].

Environments. We conduct our experiments on Intel 10C20T Core i9-10850K CPU with GIGABYTE RTX3090 24GB GPU on development environment Ubuntu18.04, Python 3.7, PyTorch 1.10.0.

Settings. Following the typical settings in [160], we randomly sample 5 items from all items as the targets. To train the RS model, we randomly sample 20% of existing ratings labeled with normal for testing, and the remaining are the training set. The rating budget of each injected user is $B = 15$. Due to the large number of items in the *Movies* dataset, the search space for the attack optimization is extensive. As noted in [161], two-layer GCN is highly susceptible to poisoning nodes within 2-hop. Therefore, we limit the space of candidate items to 2-hop neighbors of the target items, which improves the efficiency of the training process. The experiments are repeated for 5 times with different random seeds to initialize model parameters. We use the averaged hit ratios of the target items, i.e., $HR@10$ and $HR@50$, as the metrics to evaluate how the attacks can promote target items. We test the attack performances under various attack powers (0.0%, 0.3%, 0.5%, 0.7%, 1.0%, 2.0%), where an attack power represents the fraction of the number of injected users over all users. In this section, the fraction of injected fake users with correct labels is set as $\tau = 30\%$ to reflect that a user deploying *GraphRfi* may have some prior knowledge about the data. However, later we show that *GraphRfi* can be successfully attacked regardless of the value of τ. In attacks, the number of inject user proportion is set to 1% of the original user number and detection fraction $\tau = 30\%$ if not mentioned. We set the alternate iteration steps $K = 100$, batch size 128. We totally train the model for T_{train} epochs ($T_{train} = 50$ for *GraphRfi*; $T_{train} = 300$ for MF-based model) during attack optimization and retrain the model after poisoning for T_{train} epochs. The smoothing parameter β for attacking the MF-based model is set to 10.0.

3.2.4.2 Baselines

We compare five representative attack methods: *Random*, *Average*, *Popular/Bandwagon* [88], *Poison Training* (*PoisonT*) [90], and *Trial Attack* [87]. Among these attacks, *Random*, *Average*,

and *Popular/Bandwagon* do not depend on the RS model, while *PoisonT* and *Trial Attack* are designed for MF-based RS. Each fake user gives the highest ratings to the target items and rates a set of *filler* items using the remaining budget with various strategies. In *Random Attack*, filler items are randomly selected, and the corresponding ratings are sampled from a normal distribution. $\mathcal{N}(\mu, \sigma^2)$, where μ and σ are the mean and deviation of all existing ratings. For *Average Attack*, the only difference from *Random Attack* is that the rating given to a filler item v_i is sampled from $\mathcal{N}(\mu_{v_i}, \sigma_{v_i})$, where the μ_{v_i}, σ_{v_i} are the means and deviation of existing ratings for item v_i. In *Popular Attack*, a portion (set as 30% in our experiment) of filler items are selected as popular items since they might have bigger impacts, and the ratings given to these popular items are also set as r_{max}. In *PoisonT Attack*, it adds poisoning users one by one with maximum ratings given to target items and trains a poisoned RS model to predict the ratings for filler items. It chooses filler items in descending order of the predicted rating scores and gives ratings sampled from a normal distribution. *Trial Attack* trains generator module, influence module, and discrimination module together to generate stealthy fake users that maximise the influence on attack goals while evading the detection of the discriminator.

3.2.4.3 Effectiveness of MetaC Attack

Againt MF. We start with MF-based RS, for which there exists a direct comparison. The attack results are summarized in Tab. 3.8 and Tab. 3.9. We can observe that **MetaC** achieves the best results under all attack powers, demonstrating its strength as well as the challenge in defending against this strong attack. Besides, the attack performance of **MetaC** is still better than *PoisonT* most of the time under the two defense frameworks with anomaly detection (highlighted by underlining in Tab. 4.1 and Tab. 4.2), which further demonstrates the power of **MetaC**. In addition, *PoisonT* has slightly better attack performance than that of *Trial Attack*. Also, due to its computational simplicity, we choose to adapt *Poison Training* for attacking *GraphRfi*.

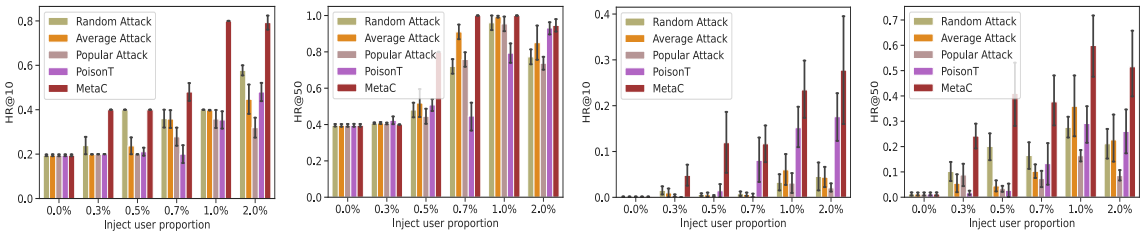
Table 3.8: Attack performances ($HR@50$) on MF-based model (*YelpCHI*).

Power	*Random*	*Average*	*Popular*	*PoisonT*	*Trial*	**MetaC**
0.0%	0.389	0.389	0.389	0.389	0.389	0.389
0.3%	0.446	0.445	0.462	0.471	0.551	**0.614**
0.5%	0.529	0.524	0.500	0.523	0.638	**0.772**
0.7%	0.584	0.587	0.540	0.661	0.694	**0.864**
1.0%	0.682	0.673	0.596	0.776	0.717	**0.929**
2.0%	0.909	0.889	0.773	0.942	0.790	**0.972**

Table 3.9: Attack performances ($HR@50$) on MF-based model (*Movies*).

Power	*Random*	*Average*	*Popular*	*PoisonT*	*Trial*	**MetaC**
0.0%	0.200	0.200	0.200	0.200	0.200	0.200
0.3%	0.376	0.371	0.258	0.386	0.398	**0.409**
0.5%	0.387	0.377	0.398	0.397	0.408	**0.522**
0.7%	0.398	0.392	0.384	0.401	0.409	**0.828**
1.0%	0.418	0.399	0.404	0.445	0.507	**0.957**
2.0%	0.787	0.550	0.534	0.929	0.620	**0.986**

Against *GraphRfi*. The hit ratios under attack (the higher, the better) are presented in Fig. 3.9. We can see that **MetaC** achieves the best attack performances in all cases, especially on *Movies*. One observation is that the gaps between **MetaC** and others are more evident for $HR@10$. Note that pushing the items to top-10 is harder than pushing to top-50, which further demonstrates the effectiveness of **MetaC**. Meanwhile, we notice that the results show a larger variance on *Movies*. A possible reason is that hit ratio is a ranking-based metric, and *Movies* has significantly more items. Thus, there are much more items around the top-10/50 threshold, making the ranking sensitive to perturbations (this large variance is also observed in related research [159, 162]).



(a) YelpCHI (b) YelpCHI (c) Movies (d) Movies

Figure 3.9: Attack performances ($HR@10$ and $HR@10$) on *GraphRfi* with different attack powers.

3.2.5 Conclusion

In this section, we demonstrated the vulnerabilities of a state-of-the-art robust recommender system called *GraphRfi* by designing an effective attack approach **MetaC**. We showed that the vulnerabilities come from the supervised nature of its fraudster detection component. In addition, we also show that our attack methods can also be applied to MF-based RS. The vulnerability of existing RS motivates as to further improve their robustness in the next chapter.

Chapter 4

Empirical Defense of Recommender Systems

Due to the prevalent of adversarial attacks in recommender systems (RS) as mentioned in Section 3.2, various defense mechanisms [57, 58] have been proposed to improve the adversarial robustness of RS against node injection attacks. These defense strategies can be broadly classified into two categories: training robust models and detecting malicious fraudsters. The first approach involves training RS models with robust parameters that can accurately predict outcomes even when the input data is poisoned. Adversarial training [163] is one of the primary techniques used to achieve this goal. The second approach focuses on developing specialized techniques to detect and identify malicious fraudsters. These techniques are designed to filter out or mitigate the impact of fraudsters on the RS. It is worth noting that the two defense approaches mentioned above are not mutually exclusive. Rather, they are complementary and can be used together to further enhance the overall adversarial robustness of RS.

However, in Section 3.2, we demonstrate that the existing defense strategy, *GraphRfi*, is still vulnerable due to its supervised nature of the fraudster detector. To address the above limitations, we propose a novel way of integrating fraudster detection into the recommendation, resulting in a robust recommendation framework that could be applicable to widely used recommender systems.

Then, based on our vulnerability analysis of *GraphRfi* (Section 3.2), we design a general robust recommendation framework termed Posterior-Detection Recommender (**PDR**) featured with an *adaptive fraudster detection* module. In particular, this new fraudster detection module will take label uncertainty into consideration and is jointly trained with existing recommender systems to enhance their robustness. Specifically, we treat the input user labels as observed but uncertain and changeable variables (*priors*). We then employ an Implicit Posterior (IP) model [164] to estimate the *posterior* probability of the true label. Furthermore, we use a strategy to dynamically adjust the prior labels based on the estimated posterior probabilities to counter the noise. The effect is that even if the input labels are noisy, they can be properly adjusted during the training process. Consequently, the fake users (even though mislabeled as normal) would have fewer contributions to the recommendation, which makes our proposed **PDR** robust against attacks.

We implement **PDR** with GNN-based and MF-based as the base RS, respectively, against the powerful attack **MetaC**. Our comprehensive experiments demonstrate that **PDR** can significantly mitigate the attack effects of **MetaC** and outperforms other defense baselines. This highlights the efficacy of our proposed adaptive fraudster detection module as a viable plug-in, which results in a general framework **PDR** to provide adversarial robustness for recommendation.

The rest of the section is organized as follows. We provide more preliminaries in Section 4.1. We describe the threat model in Section 4.2. Then, we propose, **PDR**, a robust recommender system framework in Section 4.3 using *GraphRfi* as the illustrative example. We investigate how to generalize **PDR** to the MF-based model in Section 4.3.4. We conduct extensive experiments in Section 4.4 to show the effectiveness of our proposed attack and defense. Finally, we conclude our findings in Section 4.5.

4.1 Preliminaries

In Section 2.2, we have introduced the recommender systems that are based on Matrix Factorization (MF) and Graph Neural Networks (GNNs). In this section, we introduce more preliminaries related to this section.

4.1.1 Posterior Estimation

In Bayesian statistics, Maximum A Posterior (MAP) estimate is a method for estimating an unknown quantity based on observed data and prior knowledge. It is obtained by finding the distribution that maximizes the likelihood function incorporated with a prior distribution. Suppose that there are n samples with feature $z_i, i \in \{1, \cdots, n\}$, and each sample has a corresponding unknown variable l_i. If the prior probability for each l_i is defined as $p(l_i), i \in \{1, \cdots, n\}$, and the observation is z_i, the posterior probability $q(l_i|z_i)$ based on the prior $p(l_i)$ and observation z_i can be estimated by maximizing the log-likelihood. According to the negative evidence lower bound (ELBO), the inequality of the negative log-likelihood regarding all observed data z_i is as follows [164]:

$$-\sum_i log(p(z_i)) \leq \underbrace{-\sum_{i,c} q(l_i = c|z_i) log(\frac{p(z_i|l_i)p(l_i = c)}{q(l_i = c|z_i)})}_{F}, \tag{4.1}$$

where c represents one of the all possible classes that l_i can take (e.g., *fake*, *normal*), the term F is also known as free energy, and minimizing F leads to maximization of the log-likelihood $\sum_i log(p(z_i))$.

It is common to have coarse and imprecise labels in computer vision tasks, such as segmentation, since high-resolution labels are usually hard to obtain. Implicit Posterior model (IP) [164] was first employed to resolve this label uncertainty problem, and it treats uncertain labels as priors and features z_i as observed data. To estimate the posterior probability of the uncertain label, $q(l_i|z_i)$ can be parameterized by a neural network. We denote θ as trainable parameters of neural network, and $q_\theta(l_i = c|z_i) := q(l_i = c|z_i; \theta)$. If F is minimized, the optimum is attained at [164]:

$$p(z_i|l_i) = \frac{q_\theta(l_i|z_i)}{\sum_j q_\theta(l_j|z_j)}. \tag{4.2}$$

By replacing the $p(z_i|l_i)$ in F, the free energy F is equivalent to the loss function $\mathcal{L}_{IP}$ that guide the optimization of posterior $q_\theta(l_i = c|z_i)$:

$$\mathcal{L}_{IP} = \sum_{i,c} \left(q_\theta(l_i = c|z_i) \log \frac{\sum_j q_\theta(l_j = c|z_j)}{p(l_i = c)} \right).$$

The minimization of the IP loss ($\mathcal{L}_{IP}$) leads to the maximization of log-likelihood regarding all observed data z_i.

4.2 Problem Statement

4.2.1 Threat Model

We consider an attacker whose goal is to promote a set of target items $\mathcal{T} \subset \mathcal{V}$. More specifically, the attacker aims to increase the probability that a target item $v_t \in \mathcal{T}$ appears in the top-k recommendation lists of target users. Based on Kerckhoffs's principle [158], we assume a worst-case scenario where the attacker has full knowledge of the target RS, including the data (i.e., the clean graph $\mathcal{G}$) and the recommendation algorithm. To achieve the malicious goal, the attacker is able to inject a set of fake users $\mathcal{U}'$ as well as some ratings (i.e., edges $\mathcal{E}'$ between $\mathcal{U}'$ and $\mathcal{V}$), resulting in a manipulated graph $\mathcal{G}' = (\mathcal{U} \cup \mathcal{U}' \cup \mathcal{V}, \mathcal{E} \cup \mathcal{E}')$. To constrain the attacker's ability, we assume that there are at most H fake users (i.e., $|\mathcal{U}'| \leq H$), and each fake user can give at most B ratings. After the attack, the defender observes the manipulated graph $\mathcal{G}'$, from which the RS is trained and tested; this attack falls into the category of data poisoning attacks.

4.2.2 Defender

The defender can only observe the poisoned graph $\mathcal{G}'$ instead of the clean one $\mathcal{G}$. The goal of the defender is to train a robust RS over $\mathcal{G}'$ that can mitigate the malicious effects of the injected fake users. Specifically, it is expected that with the robust RS, the target items would not be significantly promoted. We note that the defender does not know which the target items are, and we only use such information for evaluation purposes. In practice, it is common for the defender to run anomaly detection systems to filter out fraudsters *before training*. To reflect this fact, we assume that the defender can identify a fraction τ ($0\% \leq \tau \leq 100\%$) of fake users reliably. This parameter τ indicates the defender's prior knowledge about the attacks; however, we emphasize that our proposed robust RS works even when $\tau = 30\%$.

4.3 Robust Recommendation under Attack

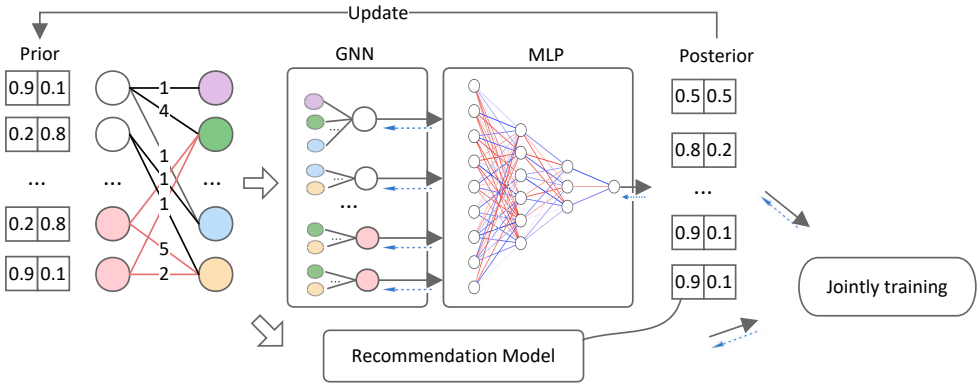


Figure 4.1: Robust recommendation framework **PDR**. Given the prior probability of each user being fake or normal, our adaptive fraudster detection module estimates the posterior probabilities based on the user behavior in the graph, using a GNN with MLP. The learned posterior probabilities are used as weights for users in the RS model, and the two models are jointly trained.

In this section, we introduce our framework **PDR** using *GraphRfi* as the illustrative example. We show that **PDR** can also be extended to MF-based RS in Section 4.3.4.

4.3.1 Framework

Our previous analysis shows that the inability to label all fake users correctly causes the failure of *GraphRfi*. In this section, we propose techniques to resolve this issue with the goal of building a robust recommender system termed Posterior-Detection Recommender system (**PDR**).

Figure 4.1 presents the framework of **PDR**, which consists of two components: a recommendation model and an adaptive fraudster detection model. The fraudster detection model starts by assigning a prior anomaly probability to users based on the given noisy label. Then, an anomaly detection module (i.e., a GNN layer and an MLP) is used to estimate the posterior probability based on the history rating graph. If the RS is a GNN-based model (e.g., *GraphRfi*), we can use the user embedding provided by the GNN in the RS directly. The detection model and rec-

ommendation model are jointly trained, and the posterior probability provided by the detection model is served as weights of users in the RS. During the training process, we adjust the prior based on the posterior. Next, we introduce the robust framework in detail.

At a high level, our anomaly detection component combines two procedures: *posterior probability estimation* and *dynamic label adjustment*. Specifically, we assume that a label associative with a user is a *variable* instead of being fixed. We turn the given noisy labels into soft labels as priors and the *unknown* true labels as latent variables. Thus, given the priors and observations (i.e., user embeddings, history rating graph), we can use a model to estimate the posterior probabilities of the true labels. Then, based on the estimated posterior probabilities, we use a strategy to dynamically adjust the soft labels (priors) during the end-to-end training process in order to estimate the posterior probabilities more accurately. Below, we articulate the details of the two procedures.

4.3.2 Posterior probability estimation

We aim to estimate the true labels based on the noisy labels and observation z_u (the user embedding learned by GNNs in RS). We define the true label of user u as a latent variable $l_u \in C = \{f, n\}$, where f and n represent fake and normal, respectively. The prior probability of this label l_u is represented as a two-dimensional vector $p(l_u) = [p(l_u = f), p(l_u = n)]$, which is user-specific prior. To take into account that the given labels are noisy, we initialize the prior probabilities as follows. For a user with a given label f, we set $p(l_u) = [1 - p_0, p_0]$ (instead of $[1, 0]$), where p_0 is the probability that a user labeled with fake is actually normal. Similarly, for a normal user, we set $p(l_u) = [p_1, 1 - p_1]$, where p_1 is the probability that a user labeled with normal is actually fake. We note that p_0 and p_1 are hyper-parameters of the system that depend on the anomaly detection system used to preprocess the data.

We further denote the posterior probability of the true label as $q(l_u|z_u) = [q(l_u = f), q(l_u = n)]$. We adopt the Implicit Posterior (IP) model [164] to estimate the posterior probability $q(l_u|z_u)$ based on the prior $p(l_u)$ and z_u. We parameterize $q(l_u = c|z_u)$ by a neural network: $q_\theta(l_u = c|z_u)$, where θ represents the trainable parameters. To obtain a more reliable posterior, we em-

Algorithm 4 PDR Defense Framework

Input: Poisoned Graph $\mathcal{G}'$; Total training epochs T_{train}; Threshold a_0; Update rate α; Update interval c_1, c_2; Observed label L, Prior parameters p_0,p_1.

function clip(x, a, b)
  if $x < a$ **then**: $x = a$
  if $x > b$ **then**: $x = b$
  return x
end function
function set_prior(L, p_0, p_1)
  for l in L **do**
    if $l = fake$ **then**: $p(l_u|z_u) = [1 - p_0, p_0]$
    if $l = normal$ **then**: $p(l_u|z_u) = [p_1, 1 - p_1]$
  end for
  return $p(l_u|z_u)$
end function
$set\_prior(L, p_0, p_1)$
for $t \leftarrow 1, \cdots, T_{train}$ **do**
  $\theta^{t+1} = \theta^t - \eta_1 \nabla_{\theta^t} \mathcal{L}(\theta^t, \mathcal{G}')$
  if AUC $> a_0$ **then**

$$p(f_u)^{t+1} = \begin{cases} \text{if } q(f_u)^t < c_1: & (1-\alpha)p(f_u)^t - \alpha(1 - q(f_u)^t) \\ \text{if } q(f_u)^t > c_2: & (1-\alpha)p(f_u)^t + \alpha q(f_u)^t \\ \text{otherwise}: & p(f_u)^t \end{cases}$$

    $q(f_u)^t = clip(q(f_u)^t, 0, 1)$
    $q(n_u)^t = 1 - q(f_u)^t$
  end if
end for
return Learned RS model parameter $\theta^* = \theta^{t+1}$.

ploy the $sofmax$ function with temperature scaling. This function produces a smoother probability when temperature $T > 1$, which prevents the model from becoming overconfident [165]. The formula is as follows,

$$softmax_T(x_i) = \frac{exp(\frac{x_i}{T})}{\sum_j exp(\frac{x_j}{T})}. \tag{4.3}$$

We use this loss $\mathcal{L}_{IP}$ to train our fraudster detection component which will output the estimated posterior probability $q_\theta(l_u|z_u)$ for each user u, and it serves as the weight in $\mathcal{L}_{\text{rating}}$. To integrate the IP model in the training of RS, we substitute the $\mathcal{L}_{\text{fraudster}}$ in Eq. (2.3) with the IP loss:

$$\begin{aligned} &\mathcal{L}(\theta, \mathcal{G}') = \mathcal{L}_{\text{rating}} + \lambda \cdot \mathcal{L}_{IP}, \\ &\mathcal{L}_{\text{rating}} = \frac{1}{|\mathcal{E}|} \sum_{\forall (u,v) \in \mathcal{E}} q_\theta(l_u = n|z_u) \cdot (r'_{uv} - r_{uv})^2 . \end{aligned} \tag{4.4}$$

4.3.3 Dynamic label adjustment

We use another technique to estimate $q(l_u|z_u)$ more accurately. We observed in our experiments that as the training continues, the posterior probabilities learned by neural networks will eventually approach to the priors, probably due to the over-fitting of neural networks. To address this, we will use the highly confident posteriors to correct the errors (noise) in the priors. In other words, we will update a soft label (prior) if the corresponding posterior is of high confidence.

Specifically, we will update the soft labels in iterations along with the training process. For ease of presentation, we use $p(f_u)^t$, $q(f_u)^t$, $p(n_u)^t$, and $q(n_u)^t$ as the simplicity of $p(l_u = f)$, $q(l_u = f)$, $p(l_u = n)$, and $q(l_u = n)$ in the t-th iteration, respectively. We update $p(f_u)$ according to the following strategy:

$$p(f_u)^{t+1} = \begin{cases} (1-\alpha)p(f_u)^t - \alpha(1 - q(f_u)^t), & q(f_u)^t < c_1 \\ (1-\alpha)p(f_u)^t + \alpha q(f_u)^t, & q(f_u)^t > c_2 \\ p(f_u)^t, & \text{otherwise.} \end{cases}$$

Basically, we use intervals $[0, c_1]$ and $[c_2, 1]$ to determine whether the estimation of $q(f_u)^t$ is confident or not. In particular, if $q(f_u)^t$ is higher than an upper-threshold c_2, we increase its prior probability $p(f_u)^{t+1}$ to $(1-\alpha)p(f_u)^t + \alpha q(f_u)^t$, where $0 < \alpha < 1$ is an update rate that

controls the adjustment speed (i.e., the effect of $p(f_u)^t$ is discounted by α). Similarly, if $q(f_u)^t$ is smaller than a lower-threshold c_1, we decrease $p(f_u)^{t+1}$ to $(1-\alpha)p(f_u)^t - \alpha(1-q(f_u)^t)$. We clip the $p(f_u)^{t+1}$ to [0,1] if it exceeds 0 or 1, and we set $p(n_u)^{t+1} = 1 - p(f_u)^{t+1}$. We apply this dynamic label adjustment after the detection AUC (Area Under Curve) on the training set first reaches a_0 that the model has a good performance but before over-fitting, where $0.5 < a_0 < 1$ is a hyper-parameter. We further summarize the whole training process of **PDR** in Algorithm 4.

4.3.4 Generalization to MF-based RS

In this section, we demonstrate that our attack and defense approaches can be applied to MF-based RS with minor modifications.

To adapt our approach to MF-based RS, we substitute the loss function Eq. (4.4) with the following:

$$\mathcal{L}_{\text{rating}} = \frac{1}{|\mathcal{E}|} \sum_{\forall (u,v) \in \mathcal{E}} q_\theta(l_u = n | z_u) \cdot (r_{uv} - U_u^T V_v)^2.$$

Without the GNNs module that can provide user embedding, we estimate the posterior probability using a single-layer GNN and an MLP: $q(l_u|z_u) = softmax_T(\text{MLP}(z_u))$, where $z_u = GNN(\mathcal{G}')$, the input is poisoned history rating graph $\mathcal{G}'$. Specifically, we normalize the ratings as the weights on edges, and employ single layer GraphSAGE [5] with mean aggregator:

$$z_u = ReLU(W \cdot (\sum_{v \in \mathcal{N}(u)} \frac{h_v \cdot \omega_{u,v}}{\sum_{i \in \mathcal{N}(u)} \omega_{u,i}} \oplus h_u), \tag{4.5}$$

where W is learnable weight matrix, and h_v is initial embedding of user or item, $ReLU(\cdot)$ is activation function, $\oplus$ is concatenating function, $\omega_{u,v} \in [0, 1]$ is the normalized rating between user u and item v.

4.4 Experiments

In section, we aim to evaluate our methods by answering the following key questions:

- How resilient is our proposed **PDR** framework against attacks (4.4.3)?
- What are the underlying mechanisms that enable **PDR** to achieve adversarial robustness (4.4.4)?
- How does the level of prior knowledge τ impact the performance of **PDR** (4.4.5)?

4.4.1 Datasets and Experiment Settings

In this chapter, we employ exactly the same dataset and experiment setting as in Section. 3.2.4.1.

In the PDR framework, we set the temperature of the $softmax$ function as $T = 2.0$. We set the probability that a user labeled with fake is actually normal as $p_0 = 0.01$, and the probability that a user labeled with normal is actually fake as $p_1 = 0.2$. In the label adjustment strategy, we update the labels when the AUC of the detection model reaches a_0 ($a_0 = 0.8$ for *GraphRfi*; $a_0 = 0.7$ for MF-based) on the training set, and we set update rate $\alpha = 0.05$ to adjust priors. At the beginning, we set the adjusting interval parameters $c_1 = 0.4$, $c_2 = 0.85$, and decreasing c_1 while increasing c_2 to decay the range of adjusting interval by $c_1^{t+1} = \min\{c_1^t - 0.025, 0.2\}$, and $c_2^{t+1} = \max\{c_2^t + 0.025, 1.0\}$. We set a larger adjust interval for the normal user side ($q(f_u)^t < c_1$) since there are more normal users than fake users. Embedding dimension in *GraphRfi* is set as 50 for *YelpCHI* and 100 for *Movies*; 128 in MF. The hidden layer number of MLP is 2. The regularization coefficient is set as 0.01 in *GraphRfi* and 1×10^{-5} in *MF*.

4.4.2 Baselines

Since there are no prior defense strategies for *GraphRfi*, we adapt representative defense approaches from both categories. First, for the adversarial training approach, we adopt a representative work proposed by [166]. Specifically, it adds *perturbation noise* to the model parameters when training the model. Second, we explore the idea of using the result of anomaly detection for defense. The natural idea is to remove the detected fake users from the system. Note that we do not constrain a specific method here for anomaly detection; instead, we assume that a fraction

τ of the injected fake users can be detected due to the fact that any anomaly detection method might be employed in practice, and the detection performance varies. In our experiment, this fraction τ of fake users are removed; we thus term this approach as *Remove Anomaly*.

4.4.3 Robustness Evaluation of PDR

The primary defense goal is to retain the hit ratios of the target items under attack. We evaluate the defense effectiveness of proposed **PDR** on GNN-based model (against **MetaC** attack) and on MF-based model (against **MetaC** and *PoisonT* attack). Fig. 4.2 presents the performances of different defense approaches applied to *GraphRfi*. We can see that **PDR** achieves the best defense performance, especially when the attack power is higher (it is also when defense is harder). We note that *Remove Anomaly* may or may not be better than *GraphRfi* (i.e., without defense). The reason is that the anomaly detection component within *GraphRfi* is supervised. Thus, removing the correctly labeled fake users, as *Remove Anomaly* did, reduces the supervision, which might harm the performance. This actually demonstrates the significance of our proposed way of dealing with those detected fake users. Similarly, the advantages of **PDR** on MF-based RS can also be observed. Tab. 4.1 and Tab. 4.2 shows the defense performance on MF-based model under 2 attacks (*PoisonT* and **MetaC**), where *No defense* is the original MF-based model, *Hard label* adds the same GNN detection model in **PDR**, but it uses the common cross-entropy loss with hard labels (similar to *GraphRfi*), and **PDR** is our robust model that uses soft label posterior detection. The results demonstrate that the MF-based model trained with **PDR** has the closest $HR@10$ to the original one with 0% attack power.

4.4.4 Why PDR is Robust

The adversarial robustness of **PDR** comes from the fact that it can detect and dynamically adjust the contributions of fake users in the recommender system.

To illustrate this point, we visualize the trajectories of anomaly scores (inversely proportional to contribution) of different types of users during the training of two systems *GraphRfi* and **PDR**

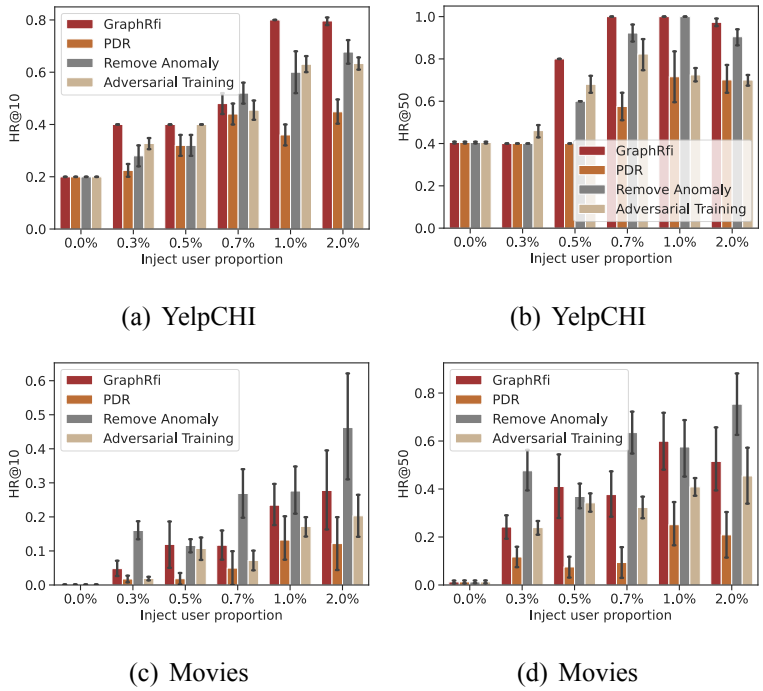
Table 4.1: Defense performances ($HR@10$) on MF-based model under different attack power (*YelpCHI*).

Attack Method	*PoisonT*			**MetaC**		
Attack Power	No defense	Hard label	**PDR**	No defense	Hard label	**PDR**
0.0%	0.214	0.214	0.214	0.214	0.214	0.214
0.3%	0.245	0.235	**0.227**	0.335	0.261	**0.240**
0.5%	0.265	0.243	**0.238**	0.426	0.281	**0.263**
0.7%	0.279	0.241	**0.231**	0.537	0.299	**0.273**
1.0%	0.302	0.265	**0.252**	0.701	0.339	**0.309**
2.0%	0.467	0.304	**0.267**	0.880	0.401	**0.364**

Table 4.2: Defense performance ($HR@10$) on MF-based model under different attack power (*Movies*).

Attack Method	*PoisonT*			**MetaC**		
Attack Power	No defense	Hard label	**PDR**	No defense	Hard label	**PDR**
0.0%	0.183	0.183	0.183	0.183	0.183	0.183
0.3%	0.201	0.274	**0.200**	0.350	0.264	**0.199**
0.5%	0.302	0.271	**0.199**	0.371	0.334	**0.322**
0.7%	0.359	0.272	**0.199**	0.424	0.456	**0.362**
1.0%	0.352	0.282	**0.200**	0.606	0.563	**0.539**
2.0%	0.407	0.382	**0.220**	0.923	0.624	**0.557**

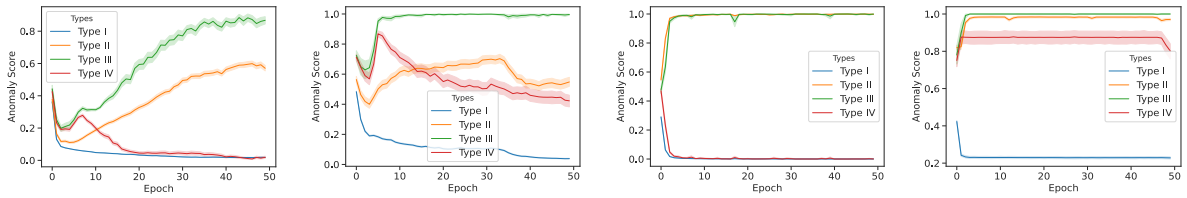
in Fig. 4.3(a) and Fig. 4.3(b), respectively. What we should focus on is the Type IV users (i.e., fake users but labeled as normal), the anomaly scores of which are shown in red over *YelpCHI*. Compared to *GraphRfi*, **PDR** can assign large anomaly scores even for Type IV users, which is the reason for its adversarial robustness. The results over *Movies* are similar.



(a) YelpCHI (b) YelpCHI

(c) Movies (d) Movies

Figure 4.2: Defense performances on GNN-based model under different attack powers.



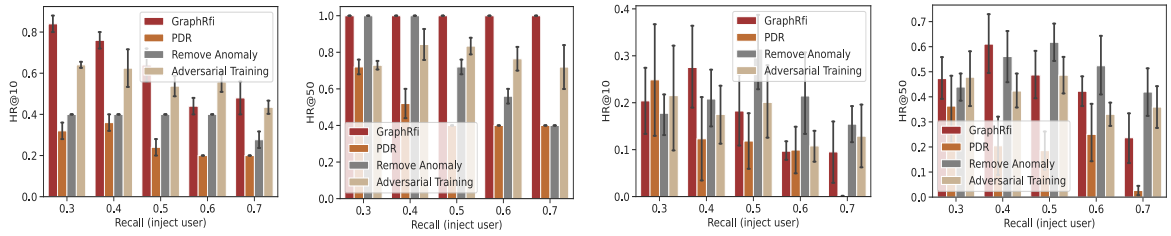
(a) without defense (*YelpCHI*) (b) with defense (*YelpCHI*) (c) without defense (*Movies*) (d) with defense (*Movies*)

Figure 4.3: Anomaly scores for different types of users.

4.4.5 Influence of Prior Knowledge

In the experiments, we use a parameter τ to control the defender's prior knowledge (possibly obtained from using some anomaly detection methods to preprocess the data) regarding the injected fake users. Specifically, τ is the *recall* over injected users defined as $\tau = \frac{|\{u \in \mathcal{U}' | \text{labeled as fake}\}|}{|\mathcal{U}'|}$, representing the fraction of fake users that are correctly labeled. We thus evaluate the two different ways (i.e., *Remove Anomaly* and **PDR**) of dealing with detected fake users under different

levels of τ. Fig. 4.4 shows that **PDR** achieves the best performance over *YelpCHI* and the performance becomes better as τ increases as it receives more supervision. Again, *Remove Anomaly* is not quite effective in some cases as removing correctly labeled fake users also decreases the supervision.



(a) YelpCHI (b) YelpCHI (c) Movies (d) Movies

Figure 4.4: Defense performance with various τ on *YelpCHI*.

4.4.6 Running Time and Complexity

The running time of **PDR** is similar to *GraphRfi*, since *GraphRfi* already includes the detection model, which takes approximately 3 hours on *YelpCHI* and 17 hours on *Movies*. When applied to the MF-based model, **PDR** takes around 2 minutes on *YelpCHI* and 6 minutes with **PDR**. The main additional computation workload in our proposed **PDR** framework is the GNN-based anomaly detection module, which has a polynomial time complexity of $O(L|\mathcal{E}|F+LNF^2)$ [103], where L is the number of layers, $|\mathcal{E}|$ is the number of ratings, N is the number of nodes, and F is the embedding dimension of nodes. It is worth noting that real-world user-item rating graphs are highly sparse [167], resulting in $|\mathcal{E}| << N^2$. This observation demonstrates the potential of employing PDR on large graphs in practical scenarios.

4.5 Conclusion

In this section, we demonstrated the vulnerabilities of a state-of-the-art robust recommender system called *GraphRfi* by designing an effective attack approach **MetaC**. We re-designed the

detection component which is equipped with the ability to dynamically adjust the importance of *newly* injected fake users, resulting in a robust RS termed **PDR**. In addition, we also show that our attack and defense methods can also be applied to MF-based RS. This research demonstrated the effectiveness of a framework for integrating anomaly detection into learning systems to improve their adversarial robustness. In our future work, we expect to see the successful application of this framework on more learning systems.

Chapter 5

Certified Robustness against Graph Injection Attacks

Deep Graph Learning (DGL), particularly Graph Neural Networks (GNNs), has established itself as the dominant approach for graph learning tasks. DGL has consistently demonstrated outstanding performance across various applications, such as recommender systems, community detection, link prediction in social networks, network intrusion detection, and anomaly detection in financial networks [6]. Many of these applications are critical for ensuring system security, such as node classification in anomaly detection, which helps prevent money laundering [168] and financial fraud [9]. Consequently, ensuring the trustworthiness of those DGL models is of paramount importance.

Indeed, extensive research has been dedicated to studying the adversarial robustness of DGL against attacks. Specifically, various graph adversarial attacks [19, 20, 21] have been proposed to assess the vulnerability of DGL models. In response, different defense mechanisms are explored, resulting in robust DGL models such as Pro-GNN [26], RobustGCN[27], and GCNGuard [28]. However, despite the effectiveness of defense models, their robustness is often compromised by the relentless development of new attack techniques [29]. The ongoing research on attacks and defense for DGL has resulted in a highly competitive status.

To end such an arms race between attack and defense, there is a growing interest in developing provable defense methods that offer certified robustness [25]. Specifically, certifiably robust models [59, 34, 33, 35] can provide the theoretical guarantee that their predictions would stay unchanged as long as the amount of input perturbations is within a certain range. For instance, a smoothed GNN-based classifier [59] can achieve a *certified accuracy* of 60% (meaning that 60% of the test nodes are *guaranteed* to be correctly classified) when faced with an attack involving the *arbitrary* deletion of up to 10 edges from the graph. Overall, certified robustness can significantly enhance the trustworthiness of DGL models in deployment.

Despite the significant progress in achieving certified robustness of models in computer vision [119, 120, 121, 129, 62, 122] and graph learning [59, 34, 33, 35], *there is a notable gap regarding the certified robustness of DGL against a novel and significant form of attack known as the Graph Injection Attack (GIA)*. Unlike the commonly investigated Graph Modification Attack (GMA), which allows the attacker to modify the existing structure of the graph, GIA involves the injection of new nodes (along with associated edges) into the original graph. *Exploring the certified robustness of DGL against GIA is of great importance for several reasons.*

Firstly, unlike in GMA where the attacker requires control over the entire graph, in GIA, the attacker only needs to control the newly injected malicious nodes. Consequently, GIA presents a less demanding threat model, making it a more realistic threat. Secondly, it is shown that GIA is a more powerful and stealthy attack compared to GMA [108, 109, 115, 3, 60, 61]. Notably, black-box GIAs such as G^2A2C [61] have successfully doubled the misclassification rate with only one node injected and one edge inserted. Lastly, GIA is a type of attack that is particularly prevalent in recommender systems [111, 112, 113, 114], where adversaries can easily create fake accounts to engage with items, deliberately damaging recommendations intended for genuine users. *The practical deployment, high stealthiness, and power of GIA* underscore the urgent need to investigate the provable robustness of DGL under such attacks.

In this chapter, we first propose a sample-wise certificate to defend against GIA, and then we further improve the certified performance by developing a collective certificate.

5.1 Sample-wise Certificate

To develop certified robustness for node classification models, the most usual solution is to certify testing nodes one by one. These are also termed sample-wise certificates.

Is adapting from existing methods sufficient? Our initial attempt is to adapt two existing and prevalent certifying frameworks to tackle the task of certifying DGL against GIA. We emphasize that these two certifying schemes have achieved state-of-the-art performance in their respective tasks, which include image classification and node classification in graphs. Specifically:

1) Bagging-based certifying scheme [1]. Such a scheme is designed to certify image classifiers against sample insertion or deletion attacks. We can extend it to our task by regarding each node as an independent sample without accounting for the graph structure.

2) Randomized-smoothing certifying scheme [59]. It is designed for DGL models against GMA. To extend it for GIA, we can pre-inject several isolated nodes in the clean graph and then certify how many edges can be injected from these nodes.

Both schemes operate by adding carefully crafted random noise to an input graph, resulting in a collection of randomized graphs, which are subsequently classified by a base classifier. The final classification result for the graph is then determined by a *majority vote* among the classifications obtained from the randomized graphs. Leveraging the Neyman-Pearson lemma [169], these schemes offer verifiable classification margins when dealing with perturbed data. This is achieved by assessing the *probability of overlap*, or likelihood ratio, between the randomized graphs originating from the clean graph and the perturbed graph. The underlying intuition is that after the randomization, if the perturbed graph and clean graph are identical, they should yield the same prediction. Consequently, a higher overlap probability corresponds to a wider certifiable radius, indicating a greater tolerance for perturbation levels.

Nevertheless, the above two adapted approaches have their own limitations, leading to poor certification performances (as shown in Tab. 5.2 and Fig. 5.5). The adapted bagging-based certifying scheme *did not properly define the perturbation space* (limitation ①), resulting in an *ex-*

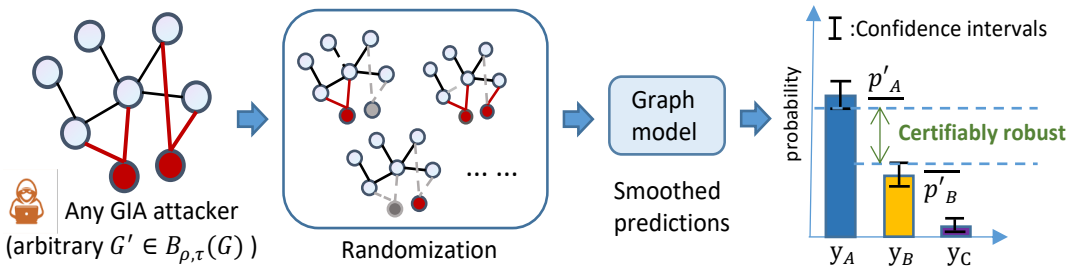


Figure 5.1: Certified Robustness via Node-aware Smoothing.

aggerated threat model where the attacker is unnecessarily too strong. In particular, the scheme completely neglects the graph structure information, resulting in no constraints on the number of connected edges for each injected node (in practice, this will make GIA easily detectable). Consequently, the certification performance against GIA is significantly compromised. Although the randomized-smoothing certifying scheme can explicitly consider graph structure and restrict the added number of edges per node, *it suffers from an extremely low probability of random sample overlap under GIAs* (limitation ②), resulting in an inadequate certification performance. These limitations underscore the necessity and the challenges of developing novel certifying approaches that can effectively leverage the graph structure while increasing the overlap probability in order to provide a more effective certificate against GIA.

Our solutions. We propose a novel **node-aware bi-smoothing** scheme to explicitly address the above limitations. Specifically, to address limitation ①, we fully consider the practical constraint for GIA that each injected node can only connect to a few edges to ensure attack unnoticeability, leading to more accurate perturbation space and improved certification performances. Our solution to this is a nontrivial generalization of the sparsity-aware certificate [59] to certify against node injection perturbation. Furthermore, to increase the sample overlap probability under GIA (limitation ②), our bi-smoothing scheme will *randomly delete nodes and edges simultaneously*. More specifically, we show that increasing the probability of deleting all inserted edges is essential for improving the overlap probability. Considering that the potential perturbed edges are concentrated around the injected nodes, node deletion enables a significantly higher probability of removing all perturbed edges originating from the injected nodes. Overall, by introducing node-aware bi-smoothing, we can model a more realistic and restricted attacker, and

increase the chance of deleting all the perturbed edges from an injected node.

We offer a rigorous theoretical analysis to establish the validity of our robustness certificate. Additionally, we show the versatility of our framework by demonstrating its effectiveness not only against evasion attacks but also against poisoning attacks. Nevertheless, to enhance the certification performance specifically for poisoning attacks, we introduce a variant called **node-aware-exclude**. This variant excludes isolated nodes from the prediction process after randomization, thereby improving the overall prediction quality.

Our comprehensive evaluation shows that the proposed node-aware bi-smoothing framework can significantly improve the certification performances of the baselines (i.e., direct adaptations). For instance, in certain cases, our scheme has shown improvements of 760% and 530% in terms of the average certifiable radius (ACR). When arbitrarily injecting 10 nodes with a maximum of 5 edges per node, our node-aware and node-aware-exclude approaches achieve certified accuracies of 35% and 55% respectively. In contrast, the two direct adapted baselines yield 0% certified accuracy.

Practical Implications. We further investigate the *practical implications* of our proposed node-aware bi-smoothing schemes from two perspectives. Firstly, we explore the application of node-aware bi-smoothing schemes as an empirical defense approach to protect against an actual GIA. We compare our smoothed classifier with other state-of-the-art robust GNN models. Remarkably, the experimental results demonstrate that our model not only achieves competitive empirical accuracy but also provides certified accuracy, which is a distinctive advantage over other empirically robust models. Secondly, we examine the potential of applying the smoothing schemes to recommendation systems, where GIAs are commonly encountered. Specifically, we treat the recommender system as a multi-label node classification task, where it predicts K items for each user (node). To assess the effectiveness of our model, we evaluate the certified number of overlap items between the predicted and the ground truth items. Notably, our model demonstrates superior certified performance compared to the baseline method [2] specifically designed for recommender systems.

We summarize the main contributions as follows:

- We address the challenging task of achieving certified robustness against the *graph injection attack*, which is a highly powerful and stealthy form of attack compared to the graph modification attack. Our primary technical advancement is a novel *node-aware bi-smoothing* scheme, which is essential to achieve enhanced certified robustness.
- Our node-aware bi-smoothing scheme is highly versatile. It is model-agnostic and is applicable to both evasion and poisoning attacks. Furthermore, with minimum modification, our scheme can also provide certification for recommender systems, where graph injection attacks are commonly observed.
- In addition, we demonstrate that our node-aware bi-smoothing scheme can be used as a practical defense strategy. Notably, our defense method achieves comparable empirical robustness to state-of-the-art robust models under actual graph injection attacks, while offering theoretical robustness guarantees.
- We conduct extensive experiments to validate the effectiveness of our schemes. The results show that our schemes can significantly improve the certified robustness against graph injection attacks compared to strong baseline methods.

Organization. We first state our problem in Section 5.1.1. In Section 5.1.2, we propose our smoothing scheme and the theoretical guarantees. Then, we illustrate the practical implementation in Section 5.1.3, and the experimental results in Section 5.1.4. Finally, we discuss the limitation in Section 5.1.5 and conclude in Section 5.1.6.

5.1.1 Problem Statement

In this section, we present a formal definition of the threat model and outline our defense goal.

5.1.1.1 Threat model

We consider an attacker whose goal is to degrade the performance of the node classification performance of a classifier. To achieve this, the attacker is allowed to inject ρ nodes $\tilde{\mathcal{V}} =$

$\{\tilde{v}_1, \cdots, \tilde{v}_\rho\}$ with arbitrary node features $\tilde{X} \in \mathbb{R}^{\rho \times d}$ into the graph. Let $\tilde{\mathcal{E}}$ denote the inserted edges from $\tilde{\mathcal{V}}$. To ensure stealthiness of attacks and constrain the attacker's ability, we assume that each injected node $\tilde{v}$ can connect at most τ edges. That is the degree of node $\tilde{v}$, denoted as $\delta(\tilde{v})$, is less than τ.

The attack causes a perturbation to the original graph G. Specifically, we define the node injection perturbation set as $B_{\rho,\tau}(G)$:

$$B_{\rho,\tau}(G) := \{G'(\mathcal{V}', \mathcal{E}', X') | \mathcal{V}' = \mathcal{V} \cup \tilde{\mathcal{V}}, \mathcal{E}' = \mathcal{E} \cup \tilde{\mathcal{E}}, \\ X' = X \cup \tilde{X}, |\tilde{\mathcal{V}}| \leq \rho, \delta(\tilde{v}) \leq \tau, \forall \tilde{v} \in \tilde{\mathcal{V}}\} \tag{5.1}$$

The perturbation set $B_{\rho,\tau}(G)$ means that there are at most ρ injected nodes, and at most $\rho \cdot \tau$ perturbed edges. This perturbation set belongs to a l_0-norm ball.

Furthermore, the perturbation can occur before or after the model training, which is defined as *evasion* attack and *poisoning* attack. We show that our proposed scheme is applicable for both the evasion and poisoning attacks.

5.1.1.2 Goal of Provable Defense

Our goal is to build up a smoothed classifier that can provide certified robustness against GIA. We emphasize that our method is model-agnostic in that it does not require to know model details. Specifically, we denote the perturbed graph with malicious nodes injected as G', and its corresponding adjacency matrix as A'. For any graph classifier $f(\cdot)$, we create its smoothed classifier $g(\cdot)$. Our goal is to verify whether the classification result for a given node v remains unchanged: $g_v(G) \stackrel{?}{=} g_v(G')$, for all adversarial example $G' \in B_{\rho,\tau}(G)$ in a predefined perturbation set.

5.1.2 Certified Robustness against GIA

In this section, we first introduce sparsity-aware smoothing [59] that serves as the basis of our scheme. Then, we propose our node-aware bi-smoothing scheme, and provide the general the-

oretical condition for provable robustness.

5.1.2.1 Preliminary: Sparsity-aware Smoothing

Our node-aware bi-smoothing scheme is a nontrivial modification from the sparsity-aware smoothing [59]. As mentioned above, sparsity-aware randomized smoothing is capable of providing l_0-ball guarantee for graph modification attack (GMA), in which the perturbation set denoted as B_{r_a,r_d} is at most r_a edges can be added and r_d edges be deleted among existing nodes. Specifically, it first specifies a randomization scheme ϕ that randomly adds or deletes edges: $\mathbb{P}(\phi(A)_{ij} \neq A_{ij}) = p_-^{A_{ij}} p_+^{(1-A_{ij})}$, where $p_-, p_+ \in [0,1]$ are the probability of adding edges or deleting edges (set $p_+ = 0$ in our adaptation). Based on the randomization, it constructs a smoothed classifier g defined in Eq. (2.4). The model verifies whether $g(G) = g(G')$ for any adversarial example $G' \in B_{r_a,r_d}(G)$ in the given graph modification perturbation set. With this existing certifying framework proposed for l_0-ball graph modification attack (GMA), next, we illustrate how to generalize it to graph injection attack (GIA).

A Direct Adaptation as Baseline We can use this model to certify the node injection perturbation set $B_{\rho,\tau}(G)$ defined in (5.1), by pre-injecting ρ isolated nodes in the clean graph, and then applying this model to certify if the model can tolerate adding arbitrary $\rho \cdot \tau$ edges (i.e., $r_a = \rho\tau$). This is based on the assumption that the isolated/singleton nodes will not impact the classification results of other nodes. Note that this assumption holds for almost all graph models, such as all message-passing GNNs [32] and common recommendation models. For the poisoning setting, we can merge the training phase and testing phase of the base classifier as a whole classifier $F(G)$ that takes G as input for both training and then making predictions. To avoid the effect from isolated nodes to other nodes, we can let $F(G)$ bypass all isolated nodes in the training phase. (For the detailed adaptation with theoretical proof, please refer to Section 5.1.2, Theorem 5 with $p_n = 0, p_- = p_e > 0$.)

Despite the applicable adaptation, we point out that, given a perturbed graph $G' \in B_{\rho,\tau}(G)$, its likelihood ratio (sample overlap) $\frac{\mathbb{P}(\phi(G')=Z)}{\mathbb{P}(\phi(G)=Z)} > 0$ only if all inserted edges are deleted (see

Appendix A.1), and the probability is $(p_-)^{\rho\tau}$, which diminishes significantly as the number of injected nodes and allowable edges grow. To enlarge the probability within the positive likelihood region, we subsequently introduce our node-aware bi-smoothing scheme.

5.1.2.2 Node-aware Bi-Smoothing

The main idea of our certificates against node injection is to design suitable smoothing distributions: (1) deleting edges $\phi_e(G)$, and (2) deleting nodes $\phi_n(G)$. Specifically, $\phi_e(G)$ randomly deletes edges in G with probability p_e, and $\phi_n(G)$ randomly deletes nodes (all its incident edges) with probability p_n. We combine edge-level and node-level smoothing distributions to form $\phi(G) = (\phi_e(G), \phi_n(G))$, which we termed **node-aware bi-smoothing**, to generate the randomized smoothing samples and then classify all of the graphs to obtain the "majority vote". We formally represent our smoothed classifier g as follows:

$$g_v(G) := \underset{y\in\{1,\cdots,C\}}{\arg\max}\ p_{v,y}(G), \tag{5.2}$$
$$p_{v,y}(G) := \mathbb{P}(f_v(\phi(G)) = y),$$

where $p_{v,y}(G)$ represents the probability that the base graph classifier f returned the class y for node v given a randomized graph $\phi(G)$, and smoothed classifier $g(\cdot)$ returns the "majority votes" of $f(\cdot)$.

We assume that for any graph model, the classification result of a query node $v \in G'$ is the same as $v \in G$ if the injected nodes are isolated from all existing nodes in the graph G. Next, we briefly illustrate how node-aware bi-smoothing can enlarge the probability of isolating all injected nodes. All the perturbed edges from an injected node are removed if the injected node is deleted in node deletion. Moreover, since the attacker is restricted to injecting only a few edges per node, both node deletion and edge deletion contribute to the probability of individually deleting perturbed edges. A perturbed edge can be deleted either through edge deletion or by deleting the other node that the edge connects to (See Fig. 5.2 for examples, and proof of Theorem 5 for more details). Since there are ρ injected nodes, and at τ injected edges per injected nodes, the probability of deleting all the perturbed edges is $\tilde{p} := (p_n + (1-p_n)(p_e + p_n - p_e p_n)^\tau)^\rho$ under

our proposed node-aware bi-smoothing (see Appendix A.1). However, when adapting sparsity-aware smoothing [59] ($p_n = 0$), the probability is $(p_e)^{\rho\tau}$, which is much smaller. Next, we formulate the certified robustness verification problem and show that such probability is the key to yielding the robustness guarantee (Theorem 5).

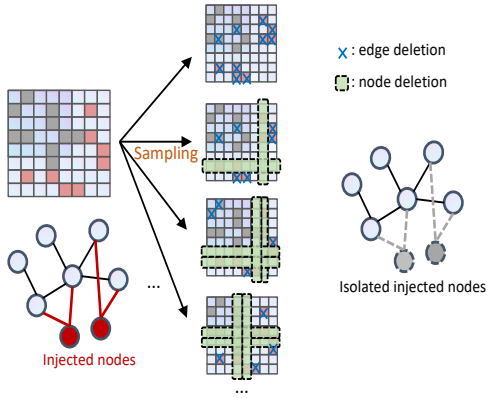


Figure 5.2: Illustration of **node-aware** bi-smoothing.

5.1.2.3 Condition for Certified Robustness under Node-aware Bi-Smoothing

We first formulate the certifying problem as a linear program following the idea of [59, 118] and then derive the condition for certified robustness.

Problem Formulation. Let us begin with defining the necessary notations. For a given graph G and $\forall v \in \mathcal{V}$, we assume that the top-class of smoothed model $g_v(G)$ is $y_A = \arg\max_{y \in \mathcal{Y}} p_{v,y}(G)$ and the running-up class is $y_B = \arg\max_{y \neq y_A} p_{v,y}(G)$. Let $p_A := p_{v,y_A}(G)$, and $p_B := p_{v,y_B}(G)$, if the node v is correctly classified by g under clean graph with certificate, we must have $p_A \geq \underline{p_A} > \overline{p_B} \geq p_B$, where the $\underline{p_A}$ is the lower bound of p_A, and $\overline{p_B}$ is the upper bound of p_B. The prediction can be further certified under perturbed graph if $p'_A > p'_B, \forall G' \in B_{r_a,r_d}(G)$, where $p'_A := p_{v,y_A}(G')$ and $p'_B := p_{v,y_B}(G')$ are the classification probabilities under perturbed graph.

The p'_A and p'_B can be obtained based on the fact that the randomized sample $\phi(G)$ and $\phi(G')$ have a probability of being overlapped, and the likelihood ratio is the same within some regions.

We can divide sample space into disjoint regions $\mathbb{G} = \bigcup_i^I \mathcal{R}_i$, where $\mathcal{R}_i$ denote the consent likelihood region that $\frac{\mathbb{P}(\phi(G)=Z)}{\mathbb{P}(\phi(G')=Z)} = c_i$ for some constant c_i. Let $r_i = \mathbb{P}(\phi(G) \in \mathcal{R}_i)$, $r'_i = \mathbb{P}(\phi(G') \in \mathcal{R}_i)$ denote the probability that the random sample fall in the partitioned region $\mathcal{R}_i$. By the law of total probability, we have $p_{v,y}(G) = \sum_i \mathbb{P}(f(Z) = y | Z \in \mathcal{R}_i)\mathbb{P}(\phi(G) = Z \in \mathcal{R}_i)$. Let $h_i := \mathbb{P}(f(Z) = y_A | Z \in \mathcal{R}_i)$ and $t_i := \mathbb{P}(f(Z) = y_B | Z \in \mathcal{R}_i)$, we have, $p'_A = p_{v,y_A}(G') = h^T r'$, and $p'_B = p_{v,y_B}(G') = t^T r'$. Then, the verification problem can be defined as a Linear Programming (LP) problem [59]:

$$\begin{aligned} \min_{h,t} \quad & \mu := p'_A - p'_B = h^T r' - t^T r', \\ \text{s.t.} \quad & h^T r = \underline{p_A},\ t^T r = \overline{p_B}, \\ & 0 \le h \le 1,\ 0 \le t \le 1, \end{aligned} \tag{5.3}$$

where the $\underline{p_A}$ is the lower bound of p_A, and $\overline{p_B}$ is the upper bound of p_B; the vectors $h \in [0,1]^I$ and $t \in [0,1]^I$ determine the worse-case classifier that assigns class y_A and class y_B among the regions such that the $\mu := p'_A - p'_B$ under perturbed graph is minimized. Hence, the optimal $\mu^* > 0$ indicates that the prediction is certified to be consistent for $\forall G' \in B_{r_a,r_d}(G)$.

Solution & Condition. The LP problem (5.3) can be solved directly according to sorted constant likelihood ratio regions ([59], Appendix B). The worst-case classifier h will assign class y_A in decreasing order of regions by their constant likelihood ratios ($c_1 \ge c_2 \ge \cdots \ge c_I$) until $\mathbb{P}(f_v(\phi(A)) = y_A) = h^T r = \underline{p_A}$, and t will assign class y_B in increasing order of the constant likelihood regions ($c_I \le \cdots \le c_1$) until $\mathbb{P}(f_v(\phi(A)) = y_B) = t^T r = \overline{p_B}$. Subsequently, leveraging this solution, we establish the theoretical condition of the certificate under our node-aware bi-smoothing scheme:

Theorem 5. *Let $f : \mathbb{G} \longrightarrow \{1, \cdots, C\}^n$ be any graph classifier, g be its smoothed classifier defined in* (5.2) *with $\phi(G) = (\phi_e(G), \phi_n(G))$, $v \in G$ be any query node, $B_{\rho,\tau}(G)$ be the node injection perturbation set defined in* (5.1)*. Suppose $y_A, y_B \in \{1, \cdots, C\}$ and $\underline{p_A}, \overline{p_B} \in [0,1]$. Then we have $g_v(G') = g_v(G)$, $\forall G' \in B_{\rho,\tau}(G)$, if:*

$$\mu_{\rho,\tau} := \tilde{p}\,(\underline{p_A} - \overline{p_B} + 1) - 1 > 0, \tag{5.4}$$

where $\tilde{p} := (p_n + (1 - p_n)(p_e + p_n - p_e p_n)^\tau)^\rho$.

Proof. (Sketch) There are two constant likelihood ratios: $c_1 = 1/\tilde{p}$ when all inserted edges are removed and $c_2 = 0$ when they are not. The worst-case classifier with condition $\mathbb{P}(f_v(\phi(A)) = y_A) = \underline{p_A}$ and $\mathbb{P}(f_v(\phi(A)) = y_B) = \overline{p_B}$ will assign class y_A in the low likelihood ratio region in priority in order to make the p'_A (classification probability under perturb graph) as small as possible. On the other hand, it will assign class y_B in the high likelihood ratio region in priority in order to make the p'_B as large as possible. The $\mu_{\rho,\tau}$ is calculated from $p'_A - p'_B$ under such a worst-case classifier. Please refer to Appendix A.1 for detailed proof. □

Based on the Theorem. 5, we have the following corollary that further highlights the important role of the probability $\tilde{p}$ in certifiable condition:

Corollary 1. *A node is only certifiable with the necessary conditions:* $\tilde{p} > \frac{1}{2}$.

Proof. With the definition of $\underline{p_A}$ and $\overline{p_B}$, we have $\underline{p_A} - \overline{p_B} \leq 1$. Then, $\mu_{\rho,\tau} = \tilde{p}(\underline{p_A} - \overline{p_B} + 1) - 1 \leq \tilde{p}(1 + 1) - 1 \leq 2 \cdot \tilde{p} - 1$. According to Theorem 5, we can certify a node if $\mu_{\rho,\tau} > 0$. We have $\mu_{\rho,\tau} > 0$ only if $2 \cdot \tilde{p} - 1 > 0$, which means $\tilde{p} > \frac{1}{2}$. □

With the Theorem. 5, we can now give black-box certified robustness for graph models against graph injection evasion attacks. Next, we show that this is also applicable to poisoning attacks with small changes.

5.1.2.4 Improving Certificate against Poisoning Attack

In this subsection, we show that our certifying scheme can also work for the poisoning attack threat model with minor adaptation. First, the certifying condition defined in Theorem 5 does not rely on the structure of the target model. That is, the Theorem 5 is suitable for any graph model as long as the isolated nodes do not impact the model predictions on other nodes. The graph model training process, however, can be viewed as an end-to-end function that takes in a graph for training and outputs a model parameter. With a fixed model parameter, it takes the same graph

as input and outputs the predictions for each node. We can combine these two processes as a complex function $F : \mathbb{G} \longrightarrow \{1, \cdots, C\}^n$, so that the previous certifying scheme is applicable to the poisoning threat model. The only difference is that the classifier itself depends on the data.

To avoid the impact of isolated nodes on the model parameter, we propose two different strategies termed **node-aware-include** and **node-aware-exclude**. By excluding isolating nodes from training while including them in the testing phase, node-aware-include strategy has the same certifying scheme as an evasion attack, because the data sampling is totally the same as in Theorem 5. Nevertheless, the graph models trained without isolated nodes might have poor generalization on isolated nodes in the testing. Furthermore, some graph models, such as graph-based recommender system models, cannot make predictions for the nodes (i.e., users and items) that are not involved in the training phase. To deal with these problems, we propose a variant termed node-aware-exclude that excludes isolated nodes totally from training and testing. However, the sample space is slightly different from that in the Theorem 5 because the base model does not vote for the isolated nodes. Next, we formally define the smoothed classier with node-aware-exclude and provide the corresponding certifying condition.

We let the smoothed classifier g under node-aware-exclude abstain from voting for all isolated nodes:

$$
\begin{aligned}
& g_v(G) := \underset{y \in \{1, \cdots, C\}}{\arg\max}\ p_{v,y}(G), \\
& p_{v,y}(G) := \mathbb{P}(F_v(\phi(G)) = y)), \\
& F_v(\phi(G)) = \text{ABSTAIN, if } v \in \phi(G) \text{ is isolated,}
\end{aligned}
\tag{5.5}
$$

where $p_{v,y}(G)$ represents the probability that the base GNN classifier F returned the class y for node v under the smoothing distribution $\phi(G)$. Note that the base classifier $F(\phi(G))$ here is first trained on $\phi(G)$ and then makes predictions. To derive the certified condition under such a smoothed classifier, we add a common assumption on the attacker that the attack edges added to a node v should not exceed its original degree $d(v)$ (which is widely adopted in almost all attackers to ensure their stealthiness). The certified condition is given as the following theorem:

Theorem 6. *Let $f : \mathbb{G} \longrightarrow \{1, \cdots, C\}^n$ be any graph classifier, g be its smoothed classifier defined in* (5.5) *with $\phi(G) = (\phi_e(G), \phi_n(G))$, $v \in G$ be any query node, $B_{\rho,\tau}(G)$ be the node injection perturbation set defined in* (5.1)*, and the attack edges added to a node v should not exceed its original degree $d(v)$. Suppose $y_A, y_B \in \{1, \cdots, C\}$ and $\underline{p_A}, \overline{p_B} \in [0, 1]$. Then we have $g_v(G') = g_v(G)$, $\forall G' \in B_{\rho,\tau}(G)$, if:*

$$\mu_{\rho,\tau} := \tilde{p}(\underline{p_A} - \frac{(1 - \underline{p'_0})\overline{p_B}}{(1 - p_0)} + 1 - \underline{p'_0}) - (1 - \underline{p'_0}) > 0, \tag{5.6}$$

where $\tilde{p} := (p_n + (1 - p_n)(p_e + p_n - p_e p_n)^\tau)^\rho$, $d(v)$ denotes the degree of node v, and $p_0 := p_n + (1 - p_n)(p_e + p_n - p_e p_n)^{d(v)}$ is the probability that the node v is deleted by the smoothing $\phi(G)$, $\underline{p'_0} := p_n + (1 - p_n)(p_e + p_n - p_e p_n)^{2d(v)}$.

Proof. (Sketch) Given a node v, the classifier does not vote for it if the node v is isolated in the smoothing. We need to calculate the likelihood ratio of regions that intersect with the region where v is not excluded. For the $\phi(G)$, it has a probability of $1 - p_0$ that the node v is included in the voting, while for $\phi(G')$, the probability is upper bound by $1 - \underline{p'_0}$ and lower bound by $1 - p_0$ because the attacker inserts new edges to node v, and the number of new edges on a single node should not exceed the original degree by assumption. The likelihood ratio has two possible values: $c_1 = \frac{1-p_0}{\tilde{p}(1-p'_0)}$ when all inserted edges are removed and $c_2 = 0$ when they are not. See Appendix A.1 for the complete proof. □

With Theorem. 6, we next illustrate that it can be further extended to provide provable robust recommendations.

5.1.2.5 Certificate for Recommender System

In particular, a recommender system can be regarded as a K-label classifier that predicts K items for each user (node). To generalize our certifying scheme to the recommender system, we adopt the framework of PORE [2] proposed by Jia et al., which defined the certified robustness problem as how many malicious users (nodes) can be injected while the recommendation for a

user u is maintained to a certain extent, i.e., *at least r recommended items are overlapped with ground truth items I_u.*

A graph-based recommender system is trained on the user-item interaction graph G. Unlike PORE, which randomly selects s users from the graph for aggregation (similar to [1]), our approach involves applying node-aware bi-smoothing to generate random graphs. Specifically, in our method, $\phi_n(G)$ removes all ratings of a user with probability p_n, while $\phi_e(G)$ removes items with probability p_e. Assuming that a base recommender system trained on $\phi(G)$ predicts K' items for a user u, we denote these predictions as $F_u(\phi(G))$. Our smoothed recommender system predicts the top-K items based on the item probabilities obtained from the base recommender system. The probabilities, denoted as $p_{u,i} = \mathbb{P}(i \in F_u(\phi(G)))$, represent the probability of item i being included in $F_u(\phi(G))$. We define the smoothed recommender system as $g_u(G)$:

$$\begin{aligned} &g_u(G) := \{i | i \in \text{top-}K(p_{u,:})\}, \\ &p_{u,i} := \mathbb{P}(i \in F_u(\phi(G))), \\ &F_u(\phi(G)) = \text{ABSTAIN, if } u \in \phi(G) \text{ is isolated,} \end{aligned} \tag{5.7}$$

where top-$K(p_{u,:})$ gives the top-K items with the largest recommendation probability $p_{u,i}$.

According to [2], we can get at least r recommended items that match with ground truth items I_u if the rth highest item probability among items I_u is higher than the $(K - r + 1)$th highest item probability among $I \setminus I_u$ under the poisoned graph, where I is all the items in the training set. We next provide the condition for $|g_u(G') \cap I_u| \geq r, \forall G' \in B_{\rho,\tau}(G)$ in the following theorem:

Theorem 7. *Let $F_u(G)$ be any base recommender system trained on G and recommend K' items to the user u, $g_u(G)$ be its smoothed recommender defined in* (5.7)*, $u \in G$ be any query user, $B_{\rho,\tau}(G)$ be the node injection perturbation set defined in* (5.1)*. Then, we have at least r recommended items after poisoning are overlapped with ground truth items I_u: $|g_u(G') \cap I_u| \geq r, \forall G' \in B_{\rho,\tau}(G)$ if:*

$$\hat{p}\,\underline{p_r} - \min_{H_c}(\overline{p}_{H_c} + K'(1-\hat{p})(1-p_0))/c > 0, \tag{5.8}$$

where $\hat{p} := (p_n + (1-p_n)p_e^\tau)^\rho$, $p_0 := p_n + (1-p_n)(p_e)^{d(u)}$ is the probability that the user u is deleted by the smoothing $\phi(G)$, $d(u)$ is the number of user ratings in training set, $\underline{p_r}$ is the

lower bound of the rth largest item probability among $\{p_{u,i}|i \in I_u\}$, H_c denote any subset of the top-$(K-r+1)$ largest items among $I \setminus I_u$ with size c, $\overline{p}_{H_c} := \sum_{j \in H_c} \overline{p_{u,j}}$ is the sum of probability upper bounds for c items in H_c.

Proof. (sketch) All malicious users are removed in the randomization with probability $\hat{p} := (p_n + (1-p_n)p_e^{\tau})^{\rho}$. See detailed proof in Appendix A.1. □

5.1.3 Implementation in Practice

With the certifying conditions from Theorem 5, Theorem 6, and Theorem 7, we aim to demonstrate how to instantiate them to train a defense model and obtain its certified robustness in both *evasion* and *poisoning* settings.

5.1.3.1 Certified Robustness Against Evasion Attack

Following [30, 59], we train a graph model with noise augmentation to enhance the model's generalization on smoothed samples. In each epoch of training, we apply $\phi(G)$ to add noise to the graph (Algorithm 5). After a graph model is trained, we sample N random graphs $G_1, G_2, \cdots, G_N$ from the smoothing distribution $\phi(G)$ to process Monte Carlo: $p_{v,y}(G) \approx \sum_{i=1}^{N} \mathbb{I}(f_v(G_i) = y))/N$. Based on this frequency, we can obtain the top two predictions y_A and y_B. Nevertheless, the prediction might not always be consistent due to the randomness. There are two levels of randomness we need to deal with: the prediction of y_A and its probability p_A. To guarantee that the model predicts y_A with probability at least $1-\alpha$, following [30], we employ a two-sided hypothesis test on the count of y_A prediction $n_A \sim Binomial(n_A + n_B, \frac{1}{2})$, where $n_A := \sum_{i=1}^{N} \mathbb{I}(f_v(G_i) = y_A)$ and $n_B := \sum_{i=1}^{N} \mathbb{I}(f_v(G_i) = y_B)$. The model returns ABSTAIN during certifying if the p-value is greater than α. To bound the probability p_A and p_B, similar to [30, 59], we compute a lower bound of $\underline{p_A}$ and upper bound of $\overline{p_B}$ based on the Clopper-Pearson Bernoulli confidence interval with confidence α/C, where C is the class number of the classifier. These lead to a lower bound of $\mu_{\rho,\tau}$, and it entails a valid certificate simultaneously

with confidence level probability α. The detailed practical certifying process is further outlined in Algorithm 6.

Algorithm 5 Training with noise (evasion).

Input: Clean graph G, smoothing distribution $\phi(G)$ with smoothing parameters p_e and p_n, training epoch E.

for $e = 1, \cdots, E$ **do**
    Draw a random graph $G_e \sim \phi(G)$.
    $f = train\_model(f(G_e))$ on training nodes.
end for
return A base classifier $f(\cdot)$.

Algorithm 6 Certified robustness with Monte Carlo sampling (evasion).

Input: Clean graph G, smoothing distribution $\phi(G)$ with smoothing parameters p_e and p_n, trained base classifier $f(\cdot)$, sample number N, confidence level α, perturbation budget ρ and τ.

Draw N random graphs $\{G_i| \sim G_i \sim \phi(G)\}_{i=1}^{N}$.
$counts = |\{i : f(G_i) = y\}|$, for $y = 1, \cdots, C$.
$y_A, y_B =$ top two indices in $counts$.
$n_A, n_B = counts[y_A], counts[y_B]$.
$\underline{p_A}, \overline{p_B} =$ CP_Bernolli(n_A, n_B, N, α).
if Binomial$(n_A + n_B, \frac{1}{2}) > \alpha$ **then**
    return ABSTAIN
    if $\mu_{\rho,\tau} > 0$ **then**
        return Certified prediction y_A.
    end if
end if
return ABSTAIN.

5.1.3.2 Certified Robustness Against Poisoning Attack

In poisoning attacks, because the perturbation is before the training phase, we take the training and inference of a model as an end-to-end function $F(\cdot)$ to substitute the base classifier $f(\cdot)$ in the evasion attack. That is, for each randomized graph $G_i \sim \phi(G)$, we first train a model based on G_i and then make predictions. We further summarize the certifying process in Algorithm 7. Note that the node-aware-include and node-aware-exclude primarily differ in the calculation of $\mu_{\rho,\tau}$. The former utilizes Eq. (5.4), while the latter employs Eq. (5.6). Regarding the implementation of the recommender system, it shares a similar training process with node-aware-exclude. The key distinction is that the prediction is obtained by the classifier defined in (5.7), and $\mu_{\rho,\tau}$ should be computed with (5.8).

Algorithm 7 Certified robustness with Monte Carlo sampling (poisoning).

Input: Clean graph G, smoothing distribution $\phi(G)$ with smoothing parameters p_e and p_n, sample number N, confidence level α, perturbation budget ρ and τ.

Draw N random graphs $\{G_i| \sim G_i \sim \phi(G)\}_{i=1}^{N}$.
$F(G_i)$: model trained on G_i and makes predictions.
$counts = |\{i : F(G_i) = y\}|$, for $y = 1, \cdots, C$.
$y_A, y_B =$ top two indices in $counts$.
$n_A, n_B = counts[y_A], counts[y_B]$.
$\underline{p_A}, \overline{p_B} = CP\_Bernolli(n_A, n_B, N, \alpha)$.
if Binomial($n_A + n_B, \frac{1}{2}$) $> \alpha$ **then**
    return ABSTAIN
    if $\mu_{\rho,\tau} > 0$ **then**
        return Certified prediction y_A.
    end if
end if
return ABSTAIN.

5.1.3.3 Empirical Robustness

Our proposed node-aware bi-smoothing can not only provide certified robustness but also serve as a general defense strategy that alleviates the threat of graph injection attack (GIA). We can follow exactly the same process used to train a smoothed model to achieve empirical robustness in the presence of a perturbed graph. Importantly, our approach is compatible with a wide range of base classifiers. This property further offers the practical values of our model.

5.1.4 Evaluation

We conduct extensive experiments on three datasets to evaluate our proposed certifiably robust framework for *node classification* and *recommender system*. We assess the certifiable robustness of the smoothed classifiers against *evasion* attack and *poisoning* attack. In summary, our experiments show the following findings:

- Our proposed node-aware bi-smoothing scheme significantly enhances the certified accuracy and average certifiable radius under various realistic graph injection attack (GIA) scenarios.
- The variant node-aware-exclude method we propose for poisoning attacks further improves the certification performance in both the node classification task and recommendation task.
- Our node-aware bi-smoothing scheme has shown competitive empirical defense performance when compared to existing baselines.
- Ablation studies demonstrate the crucial role of node-aware bi-smoothing and node-aware-exclude in achieving successful certification against GIA.

5.1.4.1 Experiment Setting

In general, we follow the settings in [30, 59]. Next, we will explain the detailed settings, including datasets and models, certificate parameters, baselines, and evaluation metrics.

Datasets and Models. We take Graph Convolution Neural Network (GCN) [4], one of the most representative GNNs, as the base classifier in node classification on Cora-ML and Citeseer datasets. For evaluating the recommender system, we take an item-based recommender system named SAR [170] on MovieLens-100k dataset [171] following [2]. Specifically, the Cora-ML dataset contains $2,995$ nodes, $8,416$ edges, and 7 classes, with an average node degree of 5.68. The Citeseer dataset contains $3,327$ nodes, $4,732$ edges, and 6 classes, with an average node degree of 3.48. For each class, we sample 50 nodes as the training set, 50 for the validation set, and the remaining for the testing set. The MovieLens-100k dataset contains about $100,000$ rating records involving 943 users and $1,682$ items. For each user, we take 85% of its history ratings as training data and the remaining for testing data.

Certificate Parameters. By default, we set the number of smoothing samples as $N = 100,000$, for certifying GCN node classification against evasion attack, and $N = 1,000$ for poisoning attack. For certifying the recommender system, we set the $N = 100,000$. For all the experiments, we see the confidence level as $\alpha = 0.01$. For the node injection perturbation set, we evaluate a range of node injection numbers ρ and various edge budgets $\tau = 5, 10$ ($\tau = 5$ if not mentioned).

Adapted Baselines. Given the absence of previous work on certifying general node classification tasks against GIA, we adapt existing certificates designed for other tasks. There are three certifying schemes adaptable to our task:

- Bagging-cert [1]: As mentioned in Section 5.1, the bagging-cert was originally designed for certifying inserting or deleting training samples in image classification tasks. To extend it for node infection perturbation, we can view each node with its incident edges as

an independent training sample, such that, the problem is the same as image classification. Note that it can only certify against poisoning attacks.

- Sparsity-aware [59]: Another way is to use sparsity-aware by adding ρ singleton nodes to the clean graph, and then certifying how many edge insertions it can withstand. When we set $p_n = 0$ in our node-aware bi-smoothing, it becomes sparsity-aware smoothing (our p_e is analogous to p_- in [59]).

- PORE [2]: It extends the bagging-cert [1] to provide a provable recommender system scheme under node injection attack, and we employ it as our baseline when certifying the recommender system.

Evaluation Metrics. A common metric to measure the robustness of a model with guarantee is ***certified accuracy***: it is the ratio of samples that the prediction is both *correct* and *certified* to be consistent under the defined perturbation set. We formally define the certified accuracy with the given attack budget ρ and τ as follows: $\xi(\rho, \tau) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(g_{v_i}(G) = g_{v_i}(G') = y_i), \forall G' \in B_{\rho,\tau}(G)$, where y_i is the ground truth of node v_i. In this chapter, we evaluate the certified accuracy of the set of testing nodes. However, evaluating the certificate strength is insufficient due to the inherent trade-off between prediction quality (i.e., clean accuracy) and certified accuracy. In general, higher smoothing variance improves the certified accuracy, but it reduces the prediction confidence. For this reason, following [130, 35], we also quantify ***average certifiable radius*** (ACR) with given degree budget τ_0: $ACR = \sum_{\rho=1}^{+\infty} \rho \cdot (\xi(\rho, \tau_0) - \xi(\rho+1, \tau_0))$. Intuitively, it is the discrete integral (area) under the certified accuracy curve (Fig. 5.3).

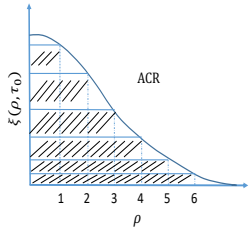


Figure 5.3: Illustration of ACR, where $\xi(\rho, \tau_0)$ denotes the certified accuracy under a fixed degree budget τ_0.

When evaluating the robustness of the recommender system, we adopt ***certified precision*** and ***certified recall*** following [2]. These metrics measure the certified overlapping of recommended items and ground truth items:

$$\text{certified precision} = \min_{G' \in B_{\rho,\tau}(G)} \frac{|I_u \cap g_u(G')|}{K}, \tag{5.9}$$

$$\text{certified recall} = \min_{G' \in B_{\rho,\tau}(G)} \frac{|I_u \cap g_u(G')|}{I_u}, \tag{5.10}$$

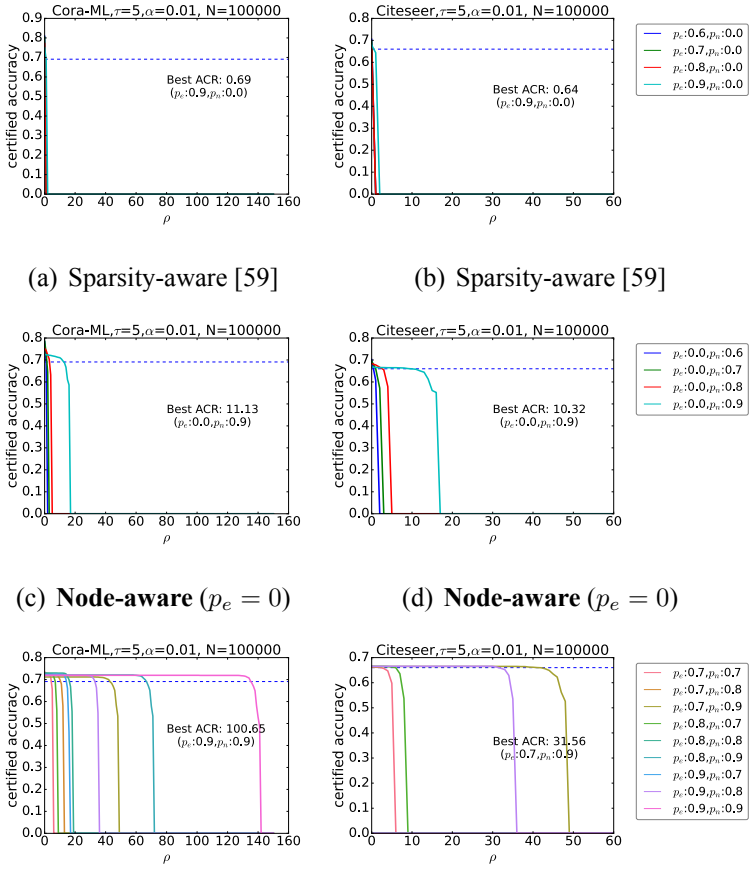
where K is the number of recommended items, and I_u is the ground truth recommendations. Similarly, *we can calculate **ACR** by substituting certified accuracy with certified precision or recall*.

5.1.4.2 Certified Robustness for Node Classification

Table 5.1: Certified accuracy comparison under *evasion* perturbation. For each method, we report the best results under different smoothing parameters, while the α and N are the same.

certified accuracy			ρ				ACR
dataset	τ	methods	0	3	5	10	
Cora-ML	5	sparse-aware [59]	**0.809**	0.000	0.000	0.000	0.691
		node-aware	0.735	**0.730**	**0.730**	**0.729**	**100.648**
	10	sparse-aware [59]	**0.809**	0.000	0.000	0.000	0.000
		node-aware	0.735	**0.730**	**0.729**	**0.721**	**51.390**
Citeseer	5	sparse-aware [59]	**0.705**	0.000	0.000	0.000	0.644
		node-aware	0.674	**0.666**	**0.666**	**0.666**	**31.558**
	10	sparse-aware [59]	**0.705**	0.000	0.000	0.000	0.000
		node-aware	0.674	**0.666**	**0.666**	**0.666**	**16.979**

Against Graph Injection Evasion Attack. For evasion attacks, we adapt sparsity smoothing [59] as a baseline for comparison (since bagging-cert [1] is designed for poisoning attacks only). In Tab. 5.1, we report the certified accuracy under the attack budgets $\rho = 3, 5, 10$, and also the clean accuracy ($\rho = 0$) of the smoothed classifier. The average certifiable radius (ACR)



(a) Sparsity-aware [59] (b) Sparsity-aware [59]

(c) **Node-aware** ($p_e = 0$) (d) **Node-aware** ($p_e = 0$)

(e) **Node-aware** ($p_e, p_n > 0$) (f) **Node-aware** ($p_e, p_n > 0$)

Figure 5.4: Certified accuracy under *evasion* perturbation with $\tau = 5$. ρ: the number of injected nodes, τ: the edge budget per injected node. The blue dotted line represents the accuracy of Multilayer Perceptron (MLP).

comparison is also shown in Tab. 5.1. For all the metrics, we report the best result for each method under various smoothing hyper-parameters. Compared to the baseline, we observe that *our proposed node-aware bi-smoothing leads to overwhelmingly higher certified accuracy* under the same budgets. For example, on the Cora-ML dataset, when there are at most 10 injected nodes with a budget degree of 5, we achieve a certified accuracy of 72.9% while the baseline is 0.0%.

In Fig. 5.4, we report the certified accuracy curve under a range of attack budget ρ using various smoothing parameters (p_e and p_n). We find that sparse-smoothing is not able to certify attack

budget $\rho > 2$ with $\tau = 5$ (Fig. 5.4a,5.4b), this highlights that *the direct adaptation from graph modification attack (GMA) to graph injection attack (GIA) is not effective although it is possible*. In contrast, *our node-aware bi-smoothing has non-trivial ACR under various parameters ($p_e > 0.7$ and $p_n > 0.7$)*.

Due to the trade-off between certified accuracy and clean accuracy, we also report the clean accuracy under various smoothing parameters in Fig. A.2a,A.2b. Since the MLP does not rely on graph structure, node injection attacks under an evasion scenario are ineffective. As a result, a robust graph classifier is meaningful only if its clean accuracy is higher than the MLP. We only report the effective results (the parameters with clean accuracy higher than the MLP).

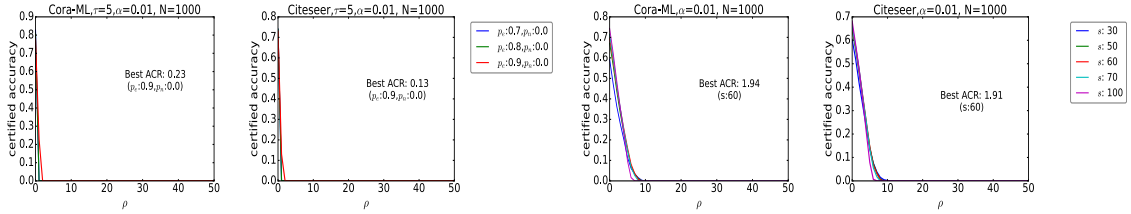
Against Graph Injection Poisoning Attack. Similar to evasion attacks, we also observed significantly higher certifiable performance under our node-aware bi-smoothing compared to baselines sparsity-aware [59] and bagging-cert [1] (Tab. 5.2 and Fig. 5.5). Notably, *our ensemble smoothed classifier has the potential to increase the clean accuracy*, as shown in Fig. A.2c, A.2d, A.2e, A.2f. Without sacrificing clean accuracy, our proposed node-aware-exclude has over 55% certifiable accuracy on Cora-ML, and 43% certifiable accuracy on Citeseer with 10 allowable malicious node with arbitrary features, while the two baseline methods are not able to certify any one of the nodes under the same condition as shown in Tab. 5.2 and Fig. 5.5. Moreover, our node-aware-exclude has an average certifiable radius (ACR) of 16.66 on Cora-ML and 12.05 on Citeseer, which improved baselines by 760% and 530%, respectively. *These experimental results reveal the effectiveness of our proposed method in poisoning attack scenarios.*

5.1.4.3 Certifiably Robust Recommender System

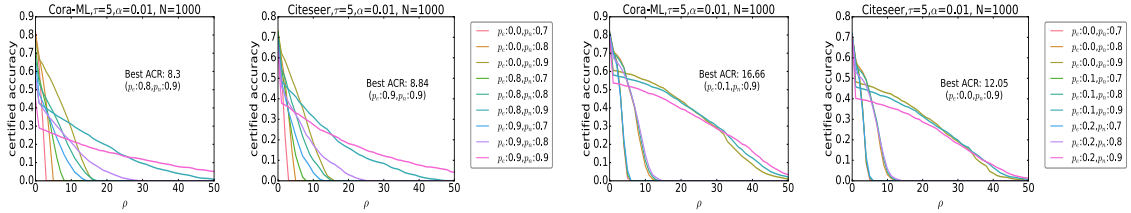
Our proposed node-aware-exclude is also applicable to provide provable recommendations. We compare our model to the provable recommender system, PORE [2]. Similar to node classification, our model is capable of considering the restricted attacker with node degree constraint, while PORE can only certify malicious nodes with unlimited degree budgets. Tab. 5.3 and Fig. 5.6 show the certified precision and recall under PORE and our proposed smoothing with

Table 5.2: Certified accuracy comparison under *poisoning* perturbation ($\tau = 5$).

certified accuracy		ρ				ACR
dataset	methods	0	3	5	10	
Cora-ML	sparse-aware[59]	**0.832**	0.000	0.000	0.000	0.229
	bagging[1]	0.744	0.354	0.163	0.000	1.931
	node-aware-include	0.770	0.480	0.428	0.351	8.297
	node-aware-exclude	0.819	**0.666**	**0.587**	**0.554**	**16.658**
Citeseer	sparse-aware[59]	**0.740**	0.000	0.000	0.000	0.130
	bagging[1]	0.681	0.362	0.146	0.001	1.905
	node-aware-include	0.734	0.493	0.443	0.321	8.835
	node-aware-exclude	0.717	**0.530**	**0.462**	**0.428**	**12.048**



(a) Sparsity-aware [59] (b) Sparsity-aware [59] (c) Bagging-cert[1] (d) Bagging-cert[1]

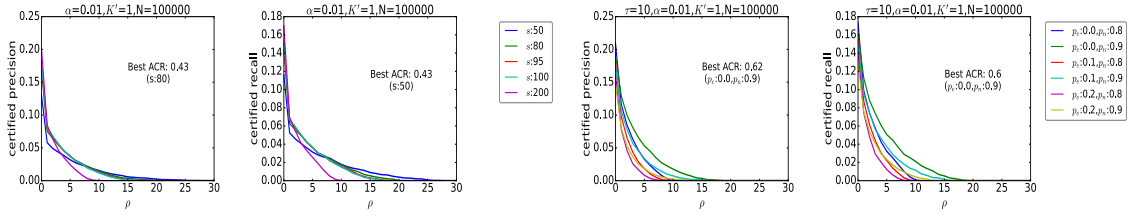


(e) **Node-aware**-include (f) **Node-aware**-include (g) **Node-aware**-exclude (h) **Node-aware**-exclude

Figure 5.5: Certified accuracy under *poisoning* perturbation with $\tau = 5$. s is the bagging size of the model [1]. The sharp decrease in certified accuracy at the beginning is due to the ABSTAIN for less confident y_A.

Table 5.3: Certified precision and recall comparison on recommender system ($\tau = 10$).

MovieLens-100k		ρ				ACR
metrics	methods	0	3	5	10	
certified precision	PORE[2]	0.203	0.056	0.038	0.015	0.427
	node-aware-exclude	**0.209**	**0.081**	**0.053**	**0.018**	**0.617**
certified recall	PORE[2]	0.170	0.050	0.037	0.018	0.424
	node-aware-exclude	**0.174**	**0.074**	**0.051**	**0.021**	**0.601**



(a) PORE [2] (b) PORE [2] (c) **Node-aware**-exclude (d) **Node-aware**-exclude

Figure 5.6: Certified precision and recall on SAR Recommender system (MovieLens-100K dataset, 85% training) under poisoning perturbation, where s is the bagging size of the model PORE [2], K' is the number of items recommended by the base recommender, and we set $K = 10$ as the number of items recommended by the smoothed recommender.

degree budget $\tau = 10$. Notably, our smoothed classifier has observed better performance.

5.1.4.4 Empirical Robustness for Node Classification

In this section, we take a state-of-the-art GIA attacker HAOGIA [3] as an example to study the empirical robustness of our model. For comparison, we take *four* widely-used GNN defense models, GCN [4], ProGNN [26], RobustGCN[27], and GNNGuard [28] as baselines, and evaluate their accuracy under attacks with budgets $\rho = \{10, 20, 30, 40, 50\}$ and $\tau = 5$. The results are presented in Tab. 5.4.

Although our model is primarily designed for certified robustness, it achieves a competitive em-

Table 5.4: Empirical robust accuracy of different defense models under HAOGIA [3] attack. The baseline models can only provide empirical robustness, while our method offers both empirical and certified robustness. We show the parameters achieve better certified accuracy (the last 3^{th} and 4^{th} columns) and better empirical accuracy (the last two columns).

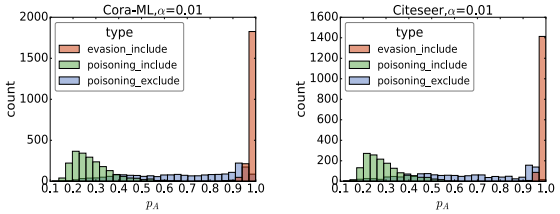
defense models		GCN	ProGNN	RobustGCN	GNNGuard	**node-aware-include**	**node-aware-exclude**	**node-aware-include**	**node-aware-exclude**
	attack	empirical robust accuracy				$(p_e: 0.8, p_n: 0.9)$	$(p_e: 0.1, p_n: 0.9)$	$(p_e: 0.1, p_n: 0.7)$	$(p_e: 0.1, p_n: 0.7)$
dataset	ρ					empirical (certified)	empirical (certified)	empirical (certified)	empirical (certified)
Cora-ML	clean	0.816	**0.832**	0.800	0.792	0.571	0.784	0.814	0.807
	10	0.815	**0.831**	0.800	0.788	0.560 (0.311)	0.778 (**0.533**)	0.814 (0.000)	0.811 (0.000)
	20	0.815	**0.830**	0.802	0.790	0.551 (0.194)	0.776 (**0.429**)	0.814 (0.000)	0.809 (0.000)
	30	0.815	**0.816**	0.791	0.782	0.550 (0.096)	0.775 (**0.300**)	0.813 (0.000)	0.785 (0.000)
	40	0.775	0.791	0.775	0.756	0.546 (0.040)	0.770 (**0.125**)	**0.813** (0.000)	0.801 (0.000)
	50	0.764	0.771	0.763	0.745	0.543 (0.008)	0.762 (**0.021**)	**0.808** (0.000)	0.793 (0.000)
	ρ	empirical robust accuracy				$(p_e: 0.8, p_n: 0.9)$	$(p_e: 0.1, p_n: 0.9)$	$(p_e: 0.6, p_n: 0.7)$	$(p_e: 0.7, p_n: 0.7)$
Citeseer	clean	0.700	0.719	0.702	0.668	0.675	0.714	**0.736**	0.732
	10	0.695	0.707	0.688	0.657	0.614 (0.312)	0.700 (**0.408**)	**0.728** (0.000)	0.724 (0.002)
	20	0.685	0.681	0.683	0.649	0.611 (0.160)	0.685 (**0.315**)	**0.732** (0.000)	0.714 (0.000)
	30	0.647	0.673	0.654	0.623	0.607 (0.085)	0.693 (**0.198**)	**0.730** (0.000)	0.711 (0.000)
	40	0.638	0.648	0.638	0.610	0.603 (0.021)	0.681 (**0.065**)	**0.730** (0.000)	0.709 (0.000)
	50	0.629	0.611	0.618	0.615	0.600 (0.001)	0.677 (**0.007**)	**0.729** (0.000)	0.706 (0.000)

pirical accuracy. Notably, the node-aware-include variant *maintains nearly unchanged empirical accuracy even as the attack budget increases*. In both datasets, we achieve the best empirical defense when the number of injected nodes $\rho \geq 40$. *These results highlight the effectiveness of our model as an empirical defense framework.* Although the node-aware-exclude variant experiences a slight decrease in empirical accuracy compared to *include*, it achieves the best certified accuracy. Notably, while our model can provide both empirical and certified robustness, other common defense baselines can only offer empirical robustness without any guarantee.

5.1.4.5 Ablation Study and Hyper-parameters

To further analyze the important factor of the effectiveness of our proposed certifying scheme, we study our node-aware bi-smoothing with a single smoothing distribution and compare the node-aware-include with node-aware-exclude.

Edge Deletion and Node Deletion smoothing. The ablation studied with the smoothing node-deletion only ($p_e = 0$) and edge-deletion only ($p_n = 0$) are shown in Fig. 5.4 and 5.5. Note that,



(a) $\underline{p_A}$ (b) $\underline{p_A}$

Figure 5.7: Histograms of $\underline{p_A}$ under different types of model.

the baseline [59] corresponds to the edge-deletion only ($p_n = 0$).

In the case of edge-deletion only ($p_n = 0$), we observe that the certifying performance improves as p_e increases, particularly in the node-aware-include strategy where the base model also votes for isolated nodes. In node-aware-*exclude* (Fig. 5.5g,5.5h), we achieve the highest Average Certifiable Radius (ACR) with small values of p_e (0.0 and 0.1). This phenomenon can be attributed to a high p_e resulting in a large number of isolated nodes that the model does not provide votes for. Consequently, the model has a less confident $\underline{p_A}$ and a higher ratio of ABSTAIN in our statistical testing for y_A due to the limited sample size. Nevertheless, this issue can be mitigated as N increases (Figure5.9), where $p_e > 0$ consistently achieves better performance. In the case of node-deletion only ($p_e = 0$), we observe an increasing performance as p_n increases (Figure5.4c,5.4d and Figure5.5e,5.5f).

These ablation studies clearly demonstrate the importance of both edge-deletion and node-deletion smoothing techniques, and the latter has a more significant impact, which is further supported by the observations in Fig. 5.8.

Comparing node-aware-include and node-aware-exclude. When comparing the *include* strategy with the *exclude* strategy in the poisoning attack scenario (Table5.2 and Figure5.5), we observe that the *exclude* strategy enhances the performance in terms of the ACR and certified accuracy. This improvement is primarily attributed to the higher confidence in p_A achieved by the *exclude* strategy (Fig. 5.7).

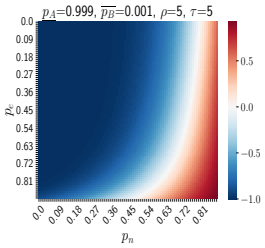
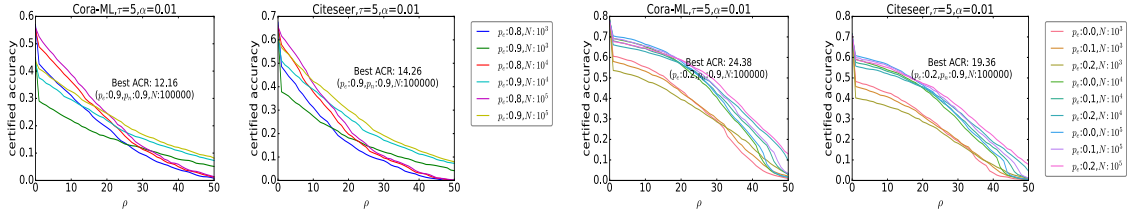


Figure 5.8: The impact of smoothing-parameters (p_e, p_n) on $\mu_{\rho,\tau}$ (*include*) under sufficiently large $\underline{p_A}$. This figure shows that both node deletion and edge deletion smoothing play an important role in the certifying condition $\mu_{\rho,\tau} > 0$ (red).



(a) **Node-aware**-include (b) **Node-aware**-include (c) **Node-aware**-exclude (d) **Node-aware**-exclude

Figure 5.9: Impact of N on certified accuracy under *poisoning* perturbation with $p_n = 0.9$, $\tau = 5$.

Hyper-parameters Analysis. Fig. 5.8 provides a visualization of the impact of p_e and p_n under the same $\underline{p_A}$ and $\overline{p_B}$. It is evident that larger values of p_e and p_n correspond to higher certifying margins $\mu_{\rho,\tau}$, highlighting the crucial role played by both edge-deletion and node-deletion smoothing techniques.

Additionally, we evaluate the effect of varying numbers of Monte-Carlo samples N in Fig. 5.9. Notably, as the value of N increases, the abstain rate decreases significantly, leading to improved certified accuracy and ACR.

5.1.5 Limitations and Future Work

This chapter focuses on a provable robust framework against graph injection attacks based on randomized smoothing. Nevertheless, the drawback of randomized smoothing is the computation overload. Future work might consider extending de-randomized smoothing [172, 173, 123] to our framework to tackle the challenge of high running time. To further improve certifiable performance of randomized smoothing, there are two common strategies: improving the training process [174, 175, 176, 177] and applying collective certification [178, 130]. The former aims to increase the intrinsic robustness of the model, while the latter further constrains the attacker to be more realistic in that it can only forge one attack sample to achieve its overall goal.

5.1.6 Conclusion

This chapter investigates the task of certifying graph-based classifiers against graph injection attacks (GIA). We propose a novel **node-aware bi-smoothing** scheme that provides certificates specifically designed to defend against GIAs under both evasion and poisoning threat models. Additionally, we propose a variant called **node-aware-exclude** to further enhance the certified performance against poisoning attacks. We evaluate the certified robustness of our model against GIAs on the GCN node classifier and SAR recommender system. While there is no previous work specifically addressing certifying general node classification against GIA, we generalize two certified robust models originally designed for other tasks and compare our model with them. Through extensive experiments on three datasets, we provide comprehensive benchmarks of our certified models against GIA. Furthermore, we evaluate the effectiveness of our model as an empirical defense method against a real GIA and compare it with four common defense models. Through extensive experiments, we demonstrate that our proposed framework not only provides significant certified robustness but also achieves competitive empirical robustness. These results demonstrate the effectiveness of our proposed model in defending against GIAs and highlight its importance in ensuring the security of graph node classification tasks.

5.2 Collective Certificate

Sample-wise v.s. Collective certification. The certification against attacks over graphs can be categorized into two types: **sample-wise** and **collective** certificates. Sample-wise certification approaches [30, 59, 131] essentially verify the prediction for a node *one by one*, assuming that the attacker can craft *a different perturbed graph each time* to attack a single node (Fig. 5.10, top). However, in reality, the attacker can only produce *a single perturbed graph* to simultaneously disrupt all predictions for a set of target nodes $\mathbb{T}$ (Fig. 5.10, bottom). Such a discrepancy makes sample-wise certificates rather pessimistic. In contrast, more recent works [130, 35] aim to certify the set of nodes at once, providing **collective certification** that can significantly improve the certifying performance.

In the domain of certifying GNNs, the majority of research works [59, 34, 33, 121, 32] focus on *sample-wise* certification against *GMA*. The only *collective* certification scheme against GMA proposed by [130], however, is not applicable to GIA. This is because the certification scheme assumes a fixed receptive field of GNNs, while GIA, which involves adding edges after injecting nodes, inevitably expands the receptive field. Although there are emerging works [131, 2] specifically designed to tackle GIA, they only offer sample-wise certificates, resulting in limited certified performance.

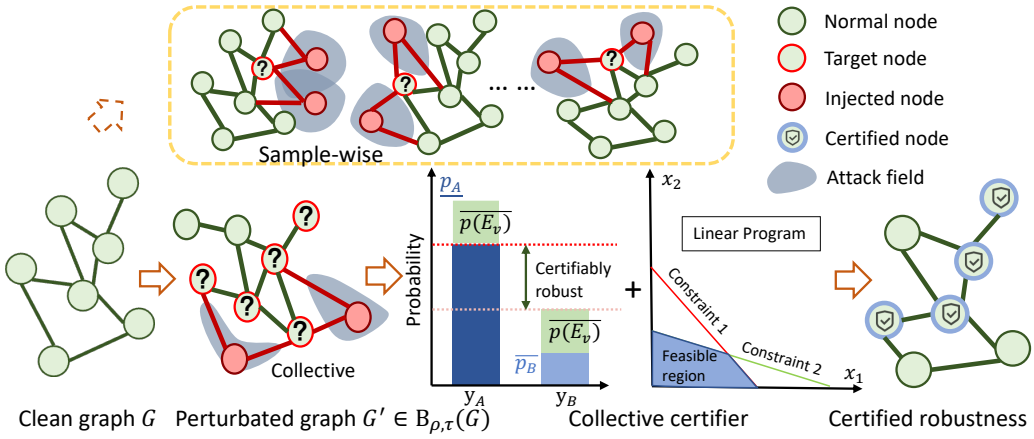


Figure 5.10: Illustration of collective certification.

We are therefore motivated to derive the **first collective certified robustness scheme** for GNNs

against GIA. To achieve collective robustness, we leverage the inherent locality property of GNNs, where the prediction of a node in a k-layer message-passing GNN is influenced solely by its k-hop neighbors. This ensures that injected edges by the attacker only impact a subset of the nodes. We address the collective certification problem by transforming it into a budget allocation problem, considering the attacker's objective of modifying the predictions of as many nodes as possible with a limited number of malicious nodes and maximum edges per node. By overestimating the number of modified nodes, we can certify the consistent classification of the remaining nodes.

However, the above problem yields a binary integer polynomial-constrained program, which is known to be NP-hard. We then propose a *customized* linearization technique to relax the original problem to a Linear Programming (LP), which can be solved efficiently. The LP relaxation provides a lower bound on the achievable certified ratio, ensuring the soundness of the verification process. We conduct comprehensive experiments to demonstrate the effectiveness as well as the computational efficiency of our collective certification scheme. For example, when the injected node number is 5% of the graph size, our collective robustness models increase the certified ratio from 0.0% to over 80.0% in both Cora-ML and Citeseer datasets, and it only takes about 1 minutes to solve the collective certifying problem.

Overall, we propose the first collective certificate for GNNs against graph injection attacks. In particular, it is computationally efficient and can significantly improve the certified ratio. Moreover, this certification scheme is *almost* model-agnostic as it is applicable for any message-passing GNNs.

5.2.1 Problem Statements

5.2.1.1 Threat Model: Graph Injection Attack

We focus on providing robustness certificates against graph injection attacks (GIAs) under the *evasion* threat model, where the attack perturbation occurs after the model training. The adversaries aim to disrupt the node classifications of a set of target nodes, denoted by $\mathbb{T}$, as many as

possible. To this end, it can inject ρ additional nodes $\tilde{\mathcal{V}} = \{\tilde{v}_1, \cdots, \tilde{v}_\rho\}$ into the graph. These injected nodes possess *arbitrary node features* represented by the matrix $\tilde{X} \in \mathbb{R}^{\rho \times d}$. Additionally, $\tilde{\mathcal{E}}$ represents the set of edges introduced by the injected nodes. To limit the power of the adversaries and avoid being detected by the defender, we assume that each injected node $\tilde{v}$ is only capable of injecting a maximum of τ edges. Thus, the degree of each injected node $\delta(\tilde{v})$ is no more than τ. Let us represent the perturbed graph as G', with its corresponding adjacency matrix denoted as A'. We formally define the potential GIA as a perturbations set associated with a given graph $G = (\mathcal{V}, \mathcal{E}, X)$:

$$B_{\rho,\tau}(G) := \{G'(\mathcal{V}', \mathcal{E}', X') | \mathcal{V}' = \mathcal{V} \cup \tilde{\mathcal{V}}, \mathcal{E}' = \mathcal{E} \cup \tilde{\mathcal{E}}, \\ X' = X \cup \tilde{X}, |\tilde{\mathcal{V}}| \leq \rho, \delta(\tilde{v}) \leq \tau, \forall \tilde{v} \in \tilde{\mathcal{V}}\}. \tag{5.11}$$

Given the absence of a collective certificate to address these types of perturbations, our first contribution is to define the problem of collective robustness.

5.2.1.2 Problem of Collective Certified Robustness

Following [32], we employ randomized smoothing to serve as the foundation of our certification. Intuitively, by adding random noise to the graph, the message from the injected node to a target node has some probability of being intercepted in the randomization, such that the GNN models will not aggregate the inserted node's feature for prediction. We adopt node-aware bi-smoothing [131], which was proposed to certify against the GIA perturbation, as our smoothed classifier. Given a graph G, random graphs are created by a randomization scheme denoted as $\phi(G) = (\phi_e(G), \phi_n(G))$. It consists of two components: edge deletion smoothing $\phi_e(G)$ and node deletion smoothing $\phi_n(G)$. Specifically, the former randomly deletes each edge with probability p_e, and the latter randomly deletes each node (together with its incident edges) with probability p_n. Based on these random graphs, a smoothed classifier $g(\cdot)$ is constructed as follows:

$$g_v(G) := \underset{y \in \{1, \cdots, K\}}{\arg\max}\ p_{v,y}(G), \tag{5.12}$$

where $p_{v,y}(G) := \mathbb{P}(f_v(\phi(G)) = y)$ represents the probability that the base GNN classifier f returned the class y for node v under the smoothing distribution $\phi(G)$, and $g(\cdot)$ returns the "majority votes" of the base classifier $f(\cdot)$.

Given a specific attack budget ρ and τ, our objective is to provide certification for the number of target nodes in $\mathbb{T}$ that are guaranteed to maintain consistent robustness against any potential attack. We assume that the attacker's objective is to maximize the disruption of predictions for the target nodes, $\sum_{v\in\mathbb{T}} \mathbb{I}\{g_v(G') \neq g_v(G)\}$, through the allocation of inserting edges. By modeling a worst-case attacker that leads to a maximum number of non-robust nodes, we can certify that the remaining number of nodes is robust. Such that the collective certification can be formulated as an optimization problem as follows:

$$\begin{aligned} \min_{G'\in B_{\rho,\tau}(G)} \quad & |\mathbb{T}| - \sum_{v\in\mathbb{T}} \mathbb{I}\{g_v(G') \neq g_v(G)\}, \\ \text{s.t.} \quad & |\tilde{\mathcal{V}}| \leq \rho,\ \delta(\tilde{v}) \leq \tau,\ \forall \tilde{v} \in \tilde{\mathcal{V}}. \end{aligned} \tag{5.13}$$

Typically, when setting the $\mathbb{T}$ as a single node, the problem degrades to a sample-wise certificate.

5.2.2 Collective Certified Robustness

In this section, we derive the collective certificate for the smoothed classifier with any message-passing GNNs as the base classifier. To ensure the clarity of the presentation, we begin by providing an overview of our approach.

5.2.2.1 Overview

The derivation of the robustness certificate relies on a *worst-case* assumption: in the message-passing process, if a node receives even a single message from any injected node, its prediction will be altered. It is important to note that this assumption exaggerates the impact of the attack, thereby validating the guarantee of the defense. Accordingly, we define **message interference** for a node v as the event E_v that the node v receives *at least* one message from injected nodes in message passing.

The achievement of collective certification then constitutes the following crucial steps. First, we derive an upper bound on the probability of the message interference event, denoted as $p(E_v)$ (Section. 5.2.2.2). Second, we establish the relation between the probability $p(E_v)$ and the prediction probability $p_{v,y}(G)$, which allows us to bound the change of $p_{v,y}(G)$ under the perturbation range $B_{\rho,\tau}(G)$ (Section. 5.2.2.2). Third, we derive the certifying condition for smoothed classifier g based on the results from the previous sections (Section. 5.2.2.2). Finally, we formulate the collective certified robustness problem as an optimization problem (Section. 5.2.2.3).

5.2.2.2 Condition for Certified Robustness

Message interference event. We begin by introducing some necessary notations. We use $P_{\tilde{v}v}^k$ to represent all the existing paths from an injected node $\tilde{v} \in \tilde{\mathcal{V}}$ to a testing node v, where the length or distance of these paths is smaller than k. Each path q in $P_{\tilde{v}v}^k$ consists of a series of linked edges. To simplify notation, we define $\phi_e(A)$ as an equivalent representation of $\phi_e(G)$, where $\phi_e(A)_{ij} = 0$ if the edge (i, j) does not exist after the sampling, and $\phi_e(A)_{ij} = 1$ if the edge (i, j) remains. Similarly, we represent $\phi_n(G)$ as $\phi_n(A)_i$, where $\phi_n(A)_i = 0$ indicates the deletion of node i, and $\phi_n(A)_i = 1$ denotes that the node remains unchanged. Then, we formally define the event E_v as:

$$\begin{aligned} \exists \tilde{v} \in \tilde{\mathcal{V}} : (\exists q \in P_{\tilde{v}v}^k : (\forall n_i \in q : \phi_n(A')_{n_i} = 1) \\ \wedge (\forall (i,j) \in q : \phi_e(A')_{ij} = 1)). \end{aligned} \tag{5.14}$$

That is at least one path from a malicious node $\tilde{v}$ to the testing node v is effective (all edges and nodes are kept in the smoothing). Below, our goal is to quantify the probability of E_v, so that we can provide an estimation of the potential impact of injected nodes on the prediction probability.

However, directly estimating the event probability $p(E_v)$ is difficult because we need to find out all the possible paths $P_{\tilde{v}v}^k$ for each node. Similar to [32], we have an upper bound for $p(E_v) \leq \overline{p(E_v)}$ by assuming the independence among the paths:

Lemma 8. *Let A be the adjacency matrix of the perturbed graph with ρ injected nodes, and the injected nodes are in the last ρ rows and columns. With smoothing $p_n > 0$ and $p_e > 0$, we have*

the upper bound of $p(E_v)$*:*

$$\begin{aligned} p(E_v) &\leq \overline{p(E_v)} \\ &= 1 - p_1^{||A_{n:(n+\rho),v}||_1} p_2^{||A^2_{n:(n+\rho),v}||_1} \cdots p_k^{||A^k_{n:(n+\rho),v}||_1}, \end{aligned} \tag{5.15}$$

where $p_i := 1 - (\bar{p}_e \bar{p}_n)^i$, $\forall i \in \{1, 2, \cdots, k\}$*, and adjacency matrix* A *contains the injected nodes encoded in the* $(n+1)^{th}$ *to* $(n+\rho)^{th}$ *row, and* $||\cdot||_1$ *is* l_1 *norm.*

Proof. (Sketch) Let $p(\bar{E}_v^{\tilde{v}})$ denote the probability that all paths are intercepted from an injected node $\tilde{v}$ to node v in the case that of considering each path independently. We have $p(\bar{E}_v^{\tilde{v}}) = \prod_{q \in P^k_{\tilde{v}v}} (1 - (\bar{p}_e \bar{p}_n)^{|q|})$, where $\bar{p}_e := 1 - p_e$, $\bar{p}_n := 1 - p_n$ and $|q| \in \{1, \cdots, k\}$ represent the length of the path $q \in P^k_{\tilde{v}v}$ from $\tilde{v}$ to v. Furthermore, $||A^k_{n:(n+\rho),v}||_1$ quantifies the number of paths with a length of k originating from any malicious node and reaching node v. Finally, by considering multiple injected nodes, we have $\overline{p(E_v)} = 1 - \prod_{\tilde{v} \in \tilde{\mathcal{V}}} p(\bar{E}_v^{\tilde{v}})$. See Appendix B.1 for complete proof. □

Bounding the change of prediction. Next, we first provide Lemma 9 to demonstrate that the occurrence of the complement event of E_v, denoted as $\bar{E}_v$, is the condition for the consistent prediction of base classifier f. Then, we prove that the change of prediction probability for the smoothed classifier g is bounded by $p(E_v)$:

Lemma 9. *Given a testing node* $v \in G$*, perturbation range* $B_{\rho,\tau}(G)$*,* $p_n > 0$ *and* $p_e > 0$*, we have* $f_v(\phi(G)) = f_v(\phi(G'))$, $\forall G' \in B_{\rho,\tau}(G)$ *if event* $\bar{E}_v$ *occurs:*

$$\begin{aligned} \forall \tilde{v} \in \tilde{\mathcal{V}} : (\forall q \in P^k_{\tilde{v}v} : (\exists n_i \in q : \phi_n(A')_{n_i} = 0) \\ \vee (\exists (i,j) \in q : \phi_e(A')_{ij} = 0)). \end{aligned} \tag{5.16}$$

Proof. For each path $q \in P^k_{\tilde{v}v}$, the message from the injected node $\tilde{v}$ to the target node v is intercepted if at least one of the edges or nodes along the path is deleted. Consequently, if all the paths are intercepted as a result of the smoothing randomization $\phi(G')$, the prediction for the target node v remains unchanged. □

Now, we can establish a bound on the change in prediction probability of the smoothed classifier g, which serves as a crucial step for deriving the certifying condition.

Theorem 10. *Given a base GNN classifier f trained on a graph G and its smoothed classifier g defined in* (5.2)*, a testing node $v \in G$ and a perturbation range $B_{\rho,\tau}(G)$, let E_v be the event defined in Eq.* (5.14)*. The absolute change in predicted probability $|p_{v,y}(G) - p_{v,y}(G')|$ for all perturbed graphs $G' \in B_{\rho,\tau}(G)$ is bounded by the probability of the event E_v: $|p_{v,y}(G) - p_{v,y}(G')| \leq p(E_v)$.*

Proof. (Sketch) $p_{v,y}(G) - p_{v,y}(G') \leq \mathbb{P}(f_v(\phi(G)) = y \wedge E_v) = p(E_v) \cdot \mathbb{P}(f_v(\phi(G)) = y | E_v) \leq p(E_v)$. See Appendix B.1 for complete proof. □

Certifying Condition. With the upper bound of the probability change $p_{v,y}(G)$ provided in Theorem 10 and upper bound of $p(E_v)$ provided in Lemma 8, we can derive the certifying condition for smoothed classifier g under a given perturbation range:

Corollary 2. *Given a base GNN classifier f trained on a graph G and its smoothed classifier g, a testing node $v \in G$ and a perturbation range $B_{\rho,\tau}(G)$, let E_v be the event defined in Eq.* (5.14)*. We have $g_v(G') = g_v(G)$ for all perturbed graphs $G' \in B_{\rho,\tau}(G)$ if:*

$$\overline{p(E_v)} < [p_{v,y^*}(G) - max_{y \neq y^*} p_{v,y}(G)]/2, \tag{5.17}$$

where $y^* \in \mathcal{Y}$ is the predicted class of $g_v(G)$.

Proof. With Theorem 10, we have $g_v(G') = g_v(G)$ if $p_{v,y^*}(G) - \overline{p(E_v)} > max_{y \neq y^*} p_{v,y}(G) + \overline{p(E_v)}$, which is equivalent to $\overline{p(E_v)} < [p_{v,y^*}(G) - max_{y \neq y^*} p_{v,y}(G)]/2$. □

Nevertheless, quantifying $\overline{p(E_v)}$ is still challenging due to the unknown paths $P^k_{\tilde{v}v}$ or the perturbed adjacency matrix. To tackle the challenge, we introduce the following collective certifying framework that models the problem of certifying node injection perturbation as an optimization problem. More importantly, we can certify a set of nodes at the same time to enhance the certifying performance.

5.2.2.3 Collective Certification as Optimization

With Corollary 2, we know that node v is not certifiably robust if

$$\overline{p(E_v)} \geq [p_{v,y^*}(G) - max_{y \neq y^*} p_{v,y}(G)]/2.$$

Under a limited attack budget, the worst-case attacker can lead to a maximum number of non-robust nodes among target nodes in $\mathbb{T}$, which can be formulated as follows:

$$\begin{aligned} \max_{G' \in B_{\rho,\tau}(G)} \quad & M = \sum_{v \in \mathbb{T}} \mathbb{I}\{\overline{p(E_v)} \geq c_v/2\}, \\ \text{s.t.} \quad & |\tilde{\mathcal{V}}| \leq \rho,\ \delta(\tilde{v}) \leq \tau,\ \forall \tilde{v} \in \tilde{\mathcal{V}}, \end{aligned} \tag{5.18}$$

where $c_v := p_{v,y^*}(G) - max_{y \neq y^*} p_{v,y}(G)$, is the classification gap of smoothed classifier. To obtain the certifiably robust node number among all testing nodes, the optimal objective value M^* of (5.18) can serve as an upper bound for non-robust nodes, and hence the remaining $|\mathbb{T}| - M^*$ nodes are certified robust. Plugging in $\overline{p(E_v)}$ with (5.15), and taking the logarithm of the $\overline{p(E_v)}$, we transformed the problem (5.18) to a binary integer *polynomial-constrained programming* (We put the problem and formulation details in Appendix B.2).

Typically, for two-layer GNNs ($k = 2$), we formulate the problem into a binary integer quadratic constrained linear programming problem (BQCLP). Let A_0 be the original adjacency matrix of the existing n nodes in the graph G, and A_1 denote the adjacency matrix from injected ρ malicious nodes to the existing nodes, and A_2 be the adjacency matrix representing the internal connection between the malicious nodes. Then the problem (B.3) becomes the BQCLP problem as follows (See Appendix B.2 for detailed formulation):

$$\begin{aligned} \max_{A_1, A_2, \mathbf{m}} \quad & M = \mathbf{t}^\top \mathbf{m}, \\ \text{s.t.} \quad & \tilde{p_1} A_1^\top \mathbf{1}_\rho + \tilde{p_2} (A_1 A_0 + A_2 A_1)^\top \mathbf{1}_\rho \leq \mathbf{C} \circ \mathbf{m}, \\ & A_1 \mathbf{1}_n + A_2 \mathbf{1}_\rho \leq \tau,\ A_2^\top = A_2, \\ & A_1 \in \{0,1\}^{\rho \times n}, \\ & A_2 \in \{0,1\}^{\rho \times \rho}, \\ & \mathbf{m} \in \{0,1\}^n, \end{aligned} \tag{5.19}$$

where **t** is a constant zero-one vector that encodes the position of the target node set $\mathbb{T}$, **m** is a vector that indicates whether the nodes are non-robust, $\tilde{p_1} = log(p_1)$ and $\tilde{p_2} = log(p_2)$ are two negative constants, $\mathbf{C} \in \mathbb{R}^n$ is a vector with negative constant elements $log(1 - \frac{c_v}{2})$, $\mathbf{1}_n$ denotes all-ones vector with length n, $\top$ represents matrix transposition, and $\circ$ denotes element-wise multiplication.

5.2.3 Effective Optimization Methods

The BQCLP problem (5.19) is non-convex and known to be NP-hard. In this section, we introduce *two* effective methods to relax problem (5.19) to a Linear Programming (LP) to solve it efficiently. The first method (termed **Collective-LP1**) relies on standard techniques to avoid quadratic terms; the second method (termed **Collective-LP2**) employs a novel customized reformulation that can significantly improve the solution quality and computational efficiency.

5.2.3.1 Standard Linear Relaxation (Collective-LP1)

To solve problem (5.19) efficiently, one common solution is to replace the quadratic terms in the constraint with linear terms by introducing extra slack variables. We adopt the standard technique [179] to address the quadratic terms in A_2A_1. Specifically, let $A_{2(ij)}$ denotes the element of i^{th} row and j^{th} column in matrix A_2 and $A_{1(jv)}$ denotes the element in matrix A_1. For each quadratic term $A_{2(ij)}A_{1(jv)}$ ($\forall i \in \{1, \cdots, \rho\}, \forall j \in \{1, \cdots, \rho\}, \forall v \in \{1, \cdots, n\}$) in A_2A_1, we can equivalently reformulate $Q_{v(ij)} := A_{2(ij)}A_{1(jv)}$ with corresponding constraints: $Q_{v(ij)} \in \mathbb{B}$, $Q_{v(ij)} \leq A_{2(ij)}$, $Q_{v(ij)} \leq A_{1(jv)}$, and $A_{2(ij)} + A_{1(jv)} - Q_{v(ij)} \leq 1$. We further relax

all the binary constraints to the box constraints $[0, 1]$, leading to an LP as follows:

$$\begin{aligned}
\max_{\substack{A_1, A_2, m, \\ Q_1, Q_2, \cdots, Q_n}} \quad & M = \mathbf{t}^\top \mathbf{m}, \\
\text{s.t.} \quad & \tilde{p_1} A_1^\top \mathbf{1}_\rho + \tilde{p_2} A_0^\top A_1^\top \mathbf{1}_\rho + \tilde{p_2} O \leq \mathbf{C} \circ \mathbf{m}, \\
& A_1 \mathbf{1}_n + A_2 \mathbf{1}_\rho \leq \tau,\ A_2^\top = A_2, \\
& Q_v = (Q_{v(ij)})_{\rho \times \rho},\ v \in \{1, 2, \cdots, n\}, \\
& O = [\mathbf{1}_\rho^\top Q_1 \mathbf{1}_\rho, \mathbf{1}_\rho^\top Q_2 \mathbf{1}_\rho, \cdots, \mathbf{1}_\rho^\top Q_n \mathbf{1}_\rho]^\top, \\
& Q_v \leq \mathbf{1}_\rho [A_{1(:,v)}]^\top,\ Q_v \leq A_2,\ Q_v \in [0, 1]^{\rho \times \rho}, \\
& \mathbf{1}_\rho [A_{1(:,v)}]^\top + A_2 - Q_v \leq 1, \\
& A_1 \in [0, 1]^{\rho \times n},\ A_2 \in [0, 1]^{\rho \times \rho},\ \mathbf{m} \in [0, 1]^n.
\end{aligned} \tag{5.20}$$

The more detailed formulation of problem (5.20) is supplied in Appendix B.2. This transformation makes our collective robustness problem solvable in polynomial time.

Validity of relaxation for certification. It is important to note that the relaxed LP problem always has a larger feasible region than the original BQCLP problem. As a result, the optimal $\bar{M}^*$ (i.e., the maximum number of non-robust nodes) of the relaxed problem is always greater than the original problem. That is, the number of robust nodes ($|\mathbb{T}| - \bar{M}^*$) certified by the relaxed problem is always smaller or equal to that obtained from the original problem, such that the relaxation always yields sound verification.

Nevertheless, this technique results in introducing $O(\rho^2|\mathbb{T}|)$ (extra) variables among the matrix O. To improve efficiency, we next design a more efficient reformulation that only requires $O(\rho|\mathbb{T}|)$ extra variables.

5.2.3.2 Customized Linear Relaxation (Collective-LP2)

To reduce the number of the extra variables, we notice that there is a vector in the quadratic term $A_1^\top A_2^\top \mathbf{1}_\rho$, and we can first combine the $A_2^\top \mathbf{1}_\rho$ to reduce the dimension. We define a vector variable $\mathbf{z} := A_2^\top \mathbf{1}_\rho$ to replace the term $A_2^\top \mathbf{1}_\rho$ in the problem (5.19). Then we can reformulate it

as:

$$\begin{aligned} \max_{A_1, z, \mathbf{m}} \quad & M = \mathbf{t}^\top \mathbf{m}, \\ \text{s.t.} \quad & \tilde{p_1} A_1^\top \mathbf{1}_\rho + \tilde{p_2} A_0^\top A_1^\top \mathbf{1}_\rho + \tilde{p_2} A_1^\top \mathbf{z} \leq \mathbf{C} \circ \mathbf{m}, \\ & A_1 \mathbf{1}_n + \mathbf{z} \leq \tau,\ A_1 \in \{0,1\}^{\rho \times n}, \\ & \mathbf{z} \in \{0, 1, \cdots, \min(\rho, \tau)\}^{\rho \times 1},\ \mathbf{m} \in \{0,1\}^n. \end{aligned} \tag{5.21}$$

To linearize the problem, we need to deal with the quadratic term $A_1^\top \mathbf{z}$. If a binary variable $x \in \mathbb{B}$, and a continuous variable $z \in [0, u]$, then $w := xz$ is equivalent to [179]: $w \leq ux, w \leq z, ux + z - w \leq u, 0 \leq w$. To apply it, we first relax the $\mathbf{z}$ to $[0, min(\tau, \rho)]$. Assuming that $\tau \leq \rho$, for each quadratic term $A_{1(ij)}^\top z_j$ ($\forall i \in \{1, \cdots, n\}, \forall j \in \{1, \cdots, \rho\}$) in $A_1^\top \mathbf{z}$, we create a substitution variable $Q_{(ij)} = A_{1(ij)}^\top z_j$ with corresponding constraints: $0 \leq Q_{(ij)}, Q_{(ij)} \leq \tau A_{1(ij)}^\top$, $Q_{(ij)} \leq z_j$, and $\tau A_{1(ij)}^\top + z_j - Q_{(ij)} \leq \tau$. We further relax all the binary constraints to $[0, 1]$ interval constraints. Then the problem (5.19) can be relaxed to an LP as follows:

$$\begin{aligned} \max_{\substack{A_1, m, \\ Q \in \mathbb{R}^{n \times \rho}}} \quad & M = \mathbf{t}^\top \mathbf{m}, \\ \text{s.t.} \quad & \tilde{p_1} A_1^\top \mathbf{1}_\rho + \tilde{p_2} A_0^\top A_1^\top \mathbf{1}_\rho + \tilde{p_2} Q \mathbf{1}_\rho \leq \mathbf{C} \circ \mathbf{m}, \\ & A_1 \mathbf{1}_n + \mathbf{z} \leq \tau,\ A_1 \in [0,1]^{\rho \times n}, \\ & Q \leq \tau A_1^\top,\ Q \leq \mathbf{1}_n \mathbf{z}^\top, \\ & \tau A_1^\top + \mathbf{1}_n \mathbf{z}^\top - Q \leq \tau, \\ & Q \in [0, \tau]^{n \times \rho},\ \mathbf{z} \in [0, \tau]^{\rho \times 1},\ \mathbf{m} \in [0,1]^n. \end{aligned} \tag{5.22}$$

We put the detailed formulation in Appendix B.2. Next, we analyze the complexity of problem (5.20) and (5.22).

5.2.3.3 Comparison of Computational Complexity

For problem (5.20), in the first constraints, the rows corresponding to the nodes that do not belong to the target node set $\mathbb{T}$ will not affect the objective M. Although we define n matrix Q_v for the sake of convenience, only $|\mathbb{T}|$ of them are actually effective. For the node with $t_i = 0$,

the value m_i will not affect the objective M, such that we can always set $m_i = 0$, and the first constraint always holds. Hence, there are $O(3\rho^2|\mathbb{T}| + \rho^2 + \rho + |\mathbb{T}|)$ effective linear constraints, and $O(\rho^2|\mathbb{T}| + \rho^2 + \rho n + |\mathbb{T}|)$ effective variables.

For problem (5.22), similar to (5.20), only $|\mathbb{T}|$ rows of Q are actually effective. There are $O(3\rho|\mathbb{T}| + \rho + |\mathbb{T}|)$ effective linear constraints, and $O(\rho n + \rho|\mathbb{T}| + |\mathbb{T}|)$ effective variables. Our well-designed formulation makes the collective problem scalable regarding the number of injected nodes ρ or the target node number $|\mathbb{T}|$. In the next section, we show that this improved LP formulation is both more efficient and effective by experimental evaluation.

5.2.4 Experimental Evaluation

In this section, we conduct a comprehensive evaluation of our proposed collective certificate. Given the absence of other collective baselines for graph injection attacks (GIA), we compare our collective certification **Collective-LP1** and **Collective-LP2**, with the existing **Sample-wise** approach [131]. We present a detailed analysis of the experimental results, highlighting the strengths and advantages of our collective certification methods.

5.2.4.1 Experimental Setup

Datasets and Base Model. We follow the literature [130, 131] on certified robustness and evaluate our methods on two graph datasets: Cora-ML [180] and Citeseer [181]. The Cora-ML dataset contains $2,810$ nodes, $7,981$ edges, 7 classes, and the Citeseer contains $2,110$ nodes, $3,668$ edges, 6 classes. We employ two representative message-passing GNNs, Graph Convolution Network (GCN) [4] and Graph Attention Network (GAT) [182], with a hidden layer size of 64 as our base classifiers. We use 50 nodes per class for training and validation respectively, while the remaining as testing nodes. We also train the base model with random noise augmentation following [131].

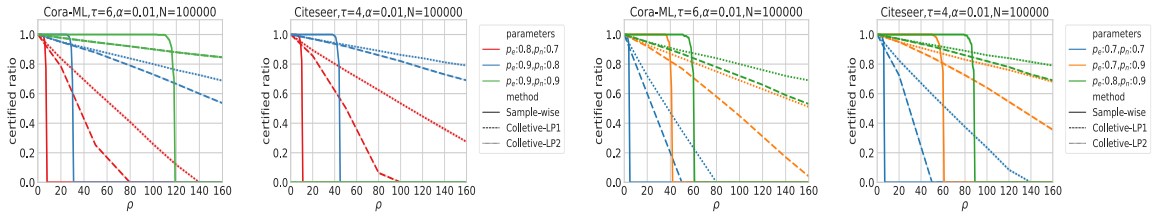
Threat Models and Certificate. We set the degree constraint per injected node as the average degree of existing nodes, which are $6 = \lceil 5.68 \rceil$ and $4 = \lceil 3.48 \rceil$ respectively on Cora-ML and Citeseer datasets. We evaluate our proposed collective certificate with various amounts of injected nodes $\rho \in \{20, 50, 80, 100, 120, 140, 160\}$. Grid search is employed to find the suitable smoothing parameters p_e and p_n from 0.5 to 0.9, respectively. We exclude those parameters that lead to poor accuracy that are worse than the Multilayer Perceptron (MLP) model which does not depend on graph structure. Following [59, 131], we employ Monte Carlo to estimate the smoothed classifier with a sample size of $N = 100,000$. We apply the Clopper-Pearson confidence interval with Bonferroni correction to obtain the lower bound of p_A and upper bound of p_B. We set the confidence level as $\alpha = 0.01$. Due to the overwhelming computation cost of the original collective certifying problem known as NP-hard, we solve our proposed relaxed LP problems by default. All our collective certifying problem is solved using MOSEK [183] through the CVXPY [184] interface.

Evaluation Metrics. Among the testing nodes that are correctly classified, we randomly select 100 nodes as the target node set $\mathbb{T}$. We report the ***certified ratio*** on the target nodes set, which is the ratio of nodes that are certifiably robust under a given threat model. We repeat 5 times with different random selections and report the average results. Additionally, we evaluate the global attack scenario in which the $\mathbb{T}$ is all the nodes in the graph in Appendix B.4.4.

5.2.4.2 Effectiveness of Collective Certified Robustness

In this section, we aim to verify the effectiveness of our proposed collective approach in enhancing the certified robustness performance.

Comparing Collective with Sample-wise. In Fig. 5.11 and Tab. 5.5, we exhibit the certified ratio of the three certificates regarding various numbers of injected nodes ρ. With the same smoothing parameter, both proposed collective certificates achieve a higher certifiable radius, outperforming the sample-wise approach significantly when the ρ is large. For example, in the



(a) Certified Ratio (GCN) (b) Certified Ratio (GCN) (c) Certified Ratio (GAT) (d) Certified Ratio (GAT)

Figure 5.11: Comparison of certified performance (More results with other parameters are shown in Appendix B.4).

Citeseer dataset, when $\rho = 140$, our Collective-LP1 and Collective-LP2 have the certified ratios of 73.0%, and 81.2%, while sample-wise can certify 0.0% nodes. Moreover, the improvement of our collective certificate is even more significant in the global attack setting (Appendix B.4.4).

When the ρ is small, the LP collective robustness does not outperform the sample-wise robustness. This can be attributed to the integrality gap of the relaxation technique utilized in the LP formulation, which we further illustrated in Section. 5.2.4.3. *Interestingly, this difference becomes negligible in the case of a global attack*, as shown in Appendix B.4.4. Nevertheless, in practical scenarios, we can easily combine the sample-wise and collective certificates with minimal effort to achieve stronger certified performance in both small and large attack budgets. Since the sample-wise and collective models share the same smoothed model, we only need to estimate the smoothing prediction once to avoid extra computation. By integrating both certificates, we can leverage their respective strengths and enhance the overall robustness of the system.

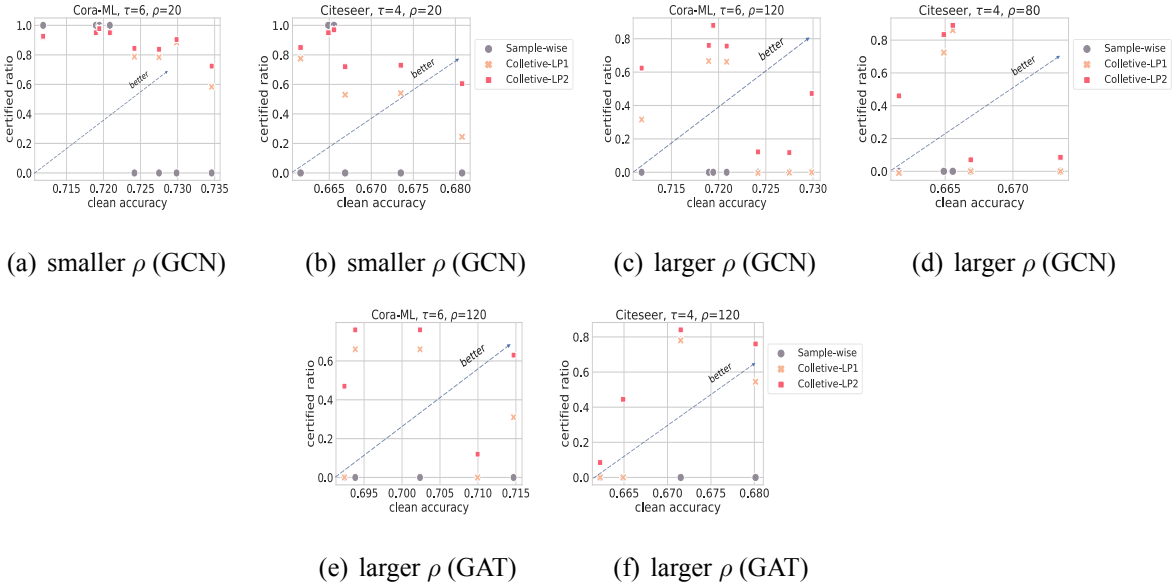
A superior certifying scheme should not only possess a higher certified ratio but also a higher clean accuracy that represents the initial performance of the model. We also evaluate the trade-off between the certified ratio and the clean accuracy of the smoothed model in Fig. 5.12. As we employ the same smoothed model, both the collective scheme and the sample-wise scheme exhibit the same clean accuracy when they share identical smoothing parameters, while our collective approach consistently achieves a higher certified ratio, particularly when ρ exceeds the certifiable radius of the sample-wise approach. Finally, these results highlight the advantageous

Table 5.5: Comparison of certified ratio between sample-wise and collective certifying schemes under various parameters.

Cora-ML ($\tau = 6$)		ρ				
parameters (p_e-p_n)	methods	20	50	100	120	140
0.7-0.9	Sample-wise	1.000	0.000	0.000	0.000	0.000
	Collective-LP1	0.920	0.768	0.452	0.316	0.178
	Collective-LP2	0.926	0.836	0.686	0.624	0.564
0.9-0.8	Sample-wise	1.000	0.000	0.000	0.000	0.000
	Collective-LP1	0.950	0.878	0.730	0.666	0.600
	Collective-LP2	0.950	0.894	0.800	0.760	0.726
0.9-0.9	Sample-wise	1.000	1.000	1.000	0.000	0.000
	Collective-LP1	0.978	0.948	0.900	0.880	0.862
	Collective-LP2	0.978	0.948	0.900	0.880	0.862
Citeseer ($\tau = 4$)		20	50	100	120	140
0.7-0.9	Sample-wise	1.000	0.990	0.000	0.000	0.000
	Collective-LP1	0.950	0.846	0.640	0.546	0.452
	Collective-LP2	0.950	0.892	0.796	0.756	0.718
0.8-0.7	Sample-wise	0.000	0.000	0.000	0.000	0.000
	Collective-LP1	0.856	0.504	0.000	0.000	0.000
	Collective-LP2	0.894	0.756	0.534	0.446	0.360
0.9-0.8	Sample-wise	1.000	0.000	0.000	0.000	0.000
	Collective-LP1	0.970	0.920	0.820	0.775	0.730
	Collective-LP2	0.970	0.930	0.862	0.840	0.812

trade-off achieved by our proposed collective approach in both smaller ρ and larger ρ.

Comparing two Collective Certificates. In comparing our two LP-based collective certificates, it is evident that our customized relaxation (Collective-LP2) consistently achieves higher or equivalent certified ratios compared to the standard technique (Collective-LP1). For instance, in the Cora-ML dataset, when $p_e = 0.7$, $p_n = 0.9$, and $\rho = 140$, Collective-LP2 improves the certified ratio by 216% compared to Collective-LP1 (Tab. 5.5). Furthermore, with the same clean

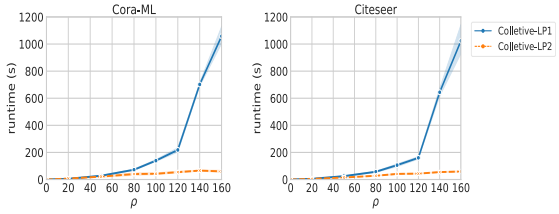


(a) smaller ρ (GCN) (b) smaller ρ (GCN) (c) larger ρ (GCN) (d) larger ρ (GCN)

(e) larger ρ (GAT) (f) larger ρ (GAT)

Figure 5.12: Trade-off between clean accuracy and certified ratio (More results with other ρ are shown in Appendix B.4).

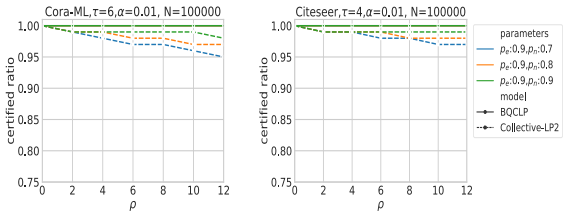
accuracy, Collective-LP2 is always superior to Collective-LP1 in certified ratios (Fig. 5.12).



(a) Runtime (b) Runtime

Figure 5.13: Runtime comparison of LP collective models.

In Fig. 5.13, we present a comparison of the runtime between our two LP-based collective certificates. It is evident that Collective-LP2 exhibits a significantly lower runtime compared to Collective-LP1, particularly as ρ increases. Remarkably, even for a larger value of ρ like $\rho = 140$, our Collective-LP2 can be solved in approximately 1 minute. This indicates the practicality and efficiency of our proposed method, making it feasible for real-world scenarios with larger attack budgets.



(a) Integrality Gap (b) Integrality Gap

Figure 5.14: Certified ratio comparison between optimizing original BQCLP problem and relaxed LP problem.

5.2.4.3 Effectiveness of Linear Relaxation

In this section, we investigate the impact of our LP relaxation technique on the certified performance of our collective certification method. Specifically, we compare the certified ratios obtained from both the original integer problem (BQCLP) and the LP problem (Collective-LP2). Fig. 5.14 provides a graphical representation of these results. Due to the computational overhead associated with solving the integer problem, we limit our analysis to a smaller attack budget, $\rho \leq 12$. We observe that the certified ratio of the integer problem remains relatively stable as ρ increases. However, the certified ratio of Collective-LP2 undergoes a decline of approximately 5%. This decrease in certified performance is attributed to the sacrifice made in the relaxation process of the LP formulation. It also partially explains why our approach may exhibit a weaker certified ratio compared to the sample-wise approach when ρ is small.

5.2.5 Limitations and Future Works

Our collective certificate is obtained through the solution of a relaxed Linear Programming (LP) problem, which effectively reduces the computational complexity to linear time. However, this relaxation does come at a cost, as it introduces an integrality gap that compromises the certified performance. Consequently, in situations where the attack budget ρ is small and the sample-wise certificate proves effective, the collective certificate may not yield superior results.

Nevertheless, in practical scenarios, we can easily combine the sample-wise and collective cer-

tificates with minimal effort to achieve stronger certified performance across a range of attack budgets, whether small or large. It is worth noting that since both the sample-wise and collective models share the same smoothed model, we only need to estimate the smoothing prediction once, avoiding computational overhead. By integrating both certificates, we can leverage their respective strengths and enhance the overall robustness of the system.

In future research, we plan to explore the development of tighter relaxations, such as semi-definite programming (SDP), to better handle the quadratic constraints. This could potentially yield improved certified performance and further enhance the robustness of our approach. Furthermore, we plan to extend the relaxation technique to accommodate polynomial constraints for deeper Graph Neural Networks (GNNs) where $k > 2$. This extension will allow us to address more complex scenarios and further strengthen the applicability of our approach in real-world settings.

5.2.6 Conclusion

In this chapter, we present the *first* collective robustness certificate specifically designed for defending against graph injection attacks (GIAs), which encompass edge addition perturbations known to be more challenging to certify than edge deletions. Our collective certificate improves the certified performance by assuming that the attacker's objective is to disrupt the predictions of as many target nodes as possible, using a shared single graph instead of different graphs for each node. We model the collective certifying problem by upper-bounding the number of non-robust nodes under a worst-case attacker, such that the remaining nodes are guaranteed to be robust. However, it yields a binary quadratic constrained programming that is NP-hard. To address this, we propose novel relaxations to formulate the problem into linear programming that can be efficiently solved. Extensive experimental results demonstrate that our proposed collective certificate achieves significantly higher certified ratios and larger certifiable radii compared to existing approaches.

Chapter 6

Conclusion

This thesis delves into the vulnerability and robustness of graph machine-learning models, a crucial aspect of artificial intelligence (AI) security and trustworthiness. Our work focuses on three key areas: adversarial attacks, empirical defenses, and certifiable defenses.

Firstly, we conducted an in-depth exploration of adversarial attacks to assess the vulnerability of current models. Specifically, we investigated Random-Walk-based Anomaly Detection (RWAD), a widely used anomaly detection tool. We provided a theoretical understanding of these attacks, including proof of NP-hardness. We introduced adversarial poisoning attacks on RWAD named **coupled-space attacks**, considering the construction of a graph on top of the feature space. By proposing graph-space attacks and leveraging them to guide feature-space attacks, we bridged the gap between these two attack types. Our experiments on multiple datasets, including both directly and indirectly accessible graphs, demonstrated the effectiveness of our proposed graph-space attack in guiding node selection and optimizing attack loss for feature-space attacks. This study provides a foundation for the vulnerability study of a feature-derived graph-based model.

Secondly, we uncovered vulnerabilities in a state-of-the-art robust recommender system (RS), named *GraphRfi*, by developing an effective attack approach called **MetaC attack**. We enhanced the detection component of the system, enabling dynamic adjustment of the importance

of newly injected fake users, resulting in a robust recommender system termed **PDR system**. Additionally, we demonstrated that our attack and defense methods can be applied to matrix factorization-based recommender systems as well. This research showcased the effectiveness of integrating anomaly detection into learning systems to enhance their empirical robustness.

Thirdly, we investigated the certified robustness of graph-based classifiers against graph injection attacks (GIAs). We proposed a novel **node-aware bi-smoothing** scheme that provides certificates specifically designed to defend against GIAs under both evasion and poisoning threat models. Furthermore, we introduced a variant called node-aware-exclude to enhance certified performance against poisoning attacks. Through extensive experiments on the GCN node classifier and SAR recommender system, we evaluated the certified robustness of our model against GIAs. Our certified models outperformed existing approaches, providing comprehensive benchmarks for defending against GIAs. We also demonstrated the effectiveness of our model as an empirical defense method against a real GIA, comparing it with four common defense models. The results showcased the significant certified robustness achieved by our proposed framework, emphasizing its importance in securing graph node classification tasks.

Lastly, we presented the first **collective robustness** certificate specifically designed for defending against GIAs. Our collective certificate improved certified performance by assuming that the attacker aims to disrupt as many target nodes as possible, using a shared single graph instead of multiple graphs for each node. However, the collective certification is challenging. To solve the NP-hard binary quadratic constrained programming problem, we proposed novel relaxations that transformed the problem into linear programming, enabling efficient solutions. Extensive experimental results demonstrated the superiority of our collective certificate, achieving significantly higher certified ratios and larger certifiable radii compared to existing approaches.

In summary, this thesis has made significant contributions to understanding and addressing the vulnerability and robustness of graph machine-learning models. Our findings provide valuable insights into the security and trustworthiness of AI systems, paving the way for future research and advancements in this critical field.

Chapter 7

Suggestions for Future Research

Future research could explore several promising directions to further enhance the security and robustness of emerging and more advanced graph machine-learning models.

7.1 Robustness of Graph Foundation Models

In recent years, the integration of Graph Neural Networks (GNNs) with large-scale pretraining techniques has catalyzed the rapid emergence of Graph Foundation Models (GFMs) [185, 186, 187], establishing them as a central research focus in graph machine learning. GFMs enable efficient cross-domain and cross-task learning by leveraging pre-trained graph representations, significantly enhancing the versatility and scalability of graph learning systems. However, the widespread adoption of GFMs also introduces new security challenges. Their reliance on large-scale data and transfer learning mechanisms makes GFMs particularly vulnerable to adversarial attacks, such as data poisoning during pretraining or targeted perturbations that degrade performance on downstream tasks. Future work should focus on systematically analyzing the vulnerabilities of GFMs, designing robust pretraining pipelines, and developing defense mechanisms that ensure security across diverse applications and domains. Additionally, extending certified robustness techniques to GFMs could provide theoretical guarantees for their performance under adversarial conditions, paving the way for more secure and reliable graph learning frameworks.

7.2 Robustness in Cross-Domain and Multi-Modal Graphs

In the era of large language models (LLMs), graphs have the potential to further enhance their performance [188, 189]. Multi-modal graphs, which combine information from multiple data sources (e.g., text, images, and graphs), are gaining popularity in domains like recommendation systems, social media analysis, and general question answering. For example, Graph Retrieval-Augmented Generation (GraphRAG) [190] combines knowledge graphs and textual data to generate answers and reasoning. GraphRAG systems are highly dependent on the interplay between structured graph data and unstructured text, making them susceptible to multi-modal adversarial attacks. For instance, attackers could manipulate the knowledge graph structure (e.g., by injecting incorrect nodes or edges) or tamper with textual inputs to mislead the generated answers and reasoning processes [191]. Future research should focus on designing robust GraphRAG frameworks that can defend against such multi-modal attacks by jointly modeling adversarial robustness across both modalities. This includes developing defense mechanisms that account for the dependencies between graph structures and text data, such as adversarial training techniques that simulate attacks on both modalities simultaneously. Furthermore, extending certified robustness techniques to GraphRAG systems could provide theoretical guarantees for their predictions, ensuring reliable reasoning even under adversarial conditions. Benchmarking datasets and evaluation protocols tailored to multi-modal systems like Graph RAG are also needed to systematically test and improve their robustness in real-world scenarios.

7.3 More Practical Certified Robustness

Future work on improving certified robustness for graph learning models should prioritize addressing the critical challenges of efficiency, scalability, and adaptability to modern graph learning paradigms, such as graph foundation models (GFMs). Existing methods, such as those based on randomized smoothing, often require thousands of predictions to certify a model's robustness, resulting in significant computational overhead [59]. To make certified robustness more practical for large-scale and real-time applications, future research should explore more efficient algorithms that reduce the number of required predictions while maintaining the reliability of the certification guarantees. Additionally, with the rapid growth of GFMs, the exploration of certified robustness for GFMs represents a critical and largely unexplored area. Future research should focus on designing certification techniques specifically tailored to GFMs, considering

their unique properties, such as zero-shot transferability, few-shot adaptation across tasks, and reliance on diverse, large-scale pretraining data. Ensuring robust and certifiable predictions for GFMs across a wide range of tasks and domains will be key to advancing secure and reliable graph learning in this new era of graph foundation models.

References

[1] J. Jia, X. Cao, and N. Z. Gong, "Intrinsic certified robustness of bagging against data poisoning attacks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 7961–7969, 2021.

[2] J. Jia, Y. Liu, Y. Hu, and N. Z. Gong, "Pore: Provably robust recommender systems against data poisoning attacks," in *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 1703–1720, 2023.

[3] Y. Chen, H. Yang, Y. Zhang, M. KAILI, T. Liu, B. Han, and J. Cheng, "Understanding and improving graph injection attack by promoting unnoticeability," in *International Conference on Learning Representations*, 2022.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[5] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[6] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: a comprehensive review," *Computational Social Networks*, vol. 6, no. 1, pp. 1–23, 2019.

[7] X. Li, L. Sun, M. Ling, and Y. Peng, "A survey of graph neural network based recommendation in social networks," *Neurocomputing*, p. 126441, 2023.

[8] S. Motie and B. Raahemi, "Financial fraud detection using graph neural networks: A systematic review," *Expert Systems With Applications*, p. 122156, 2023.

[9] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, "A comprehensive survey on graph anomaly detection with deep learning," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[10] C. Gao, Y. Zheng, N. Li, Y. Li, Y. Qin, J. Piao, Y. Quan, J. Chang, D. Jin, X. He, *et al.*, "A survey of graph neural networks for recommender systems: Challenges, methods, and directions," *ACM Transactions on Recommender Systems*, vol. 1, no. 1, pp. 1–51, 2023.

[11] P. Qiao, Z. Zhang, Z. Li, Y. Zhang, K. Bian, Y. Li, and G. Wang, "Tag: Joint triple-hierarchical attention and gcn for review-based social recommender system," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[12] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: a survey," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–37, 2022.

[13] H. Chen, L. Wang, Y. Lin, C.-C. M. Yeh, F. Wang, and H. Yang, "Structured graph convolutional networks with stochastic masks for recommender systems," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 614–623, 2021.

[14] L. Deng, D. Lian, C. Wu, and E. Chen, "Graph convolution network based recommender systems: Learning guarantee and item mixture powered strategy," *Advances in Neural Information Processing Systems*, vol. 35, pp. 3900–3912, 2022.

[15] Z. Zhai, P. Li, and S. Feng, "State of the art on adversarial attacks and defenses in graphs," *Neural Computing and Applications*, vol. 35, pp. 18851–18872, Sept. 2023.

[16] Y. Deldjoo, T. D. Noia, and F. A. Merra, "A survey on adversarial recommender systems: from attack/defense strategies to generative adversarial networks," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[17] L. Sun, Y. Dou, C. Yang, K. Zhang, J. Wang, S. Y. Philip, L. He, and B. Li, "Adversarial attack and defense on graph data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[18] V. W. Anelli, Y. Deldjoo, T. DiNoia, and F. A. Merra, "Adversarial recommender systems: Attack, defense, and advances," in *Recommender systems handbook*, pp. 335–379, Springer, 2021.

[19] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *International Conference on Learning Representations*, 2018.

[20] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *International Conference on Learning Representations*, 2019.

[21] Z. Liu, Y. Luo, L. Wu, Z. Liu, and S. Z. Li, "Towards reasonable budget allocation in untargeted graph structure attacks via gradient debias," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27966–27977, 2022.

[22] X. Zhang, J. Chen, R. Zhang, C. Wang, and L. Liu, "Attacking recommender systems with plausible profile," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4788–4800, 2021.

[23] H. Zhang, Y. Li, B. Ding, and J. Gao, "Practical data poisoning attack against next-item recommendation," in *Proceedings of The Web Conference 2020*, ACM, apr 2020.

[24] M. Si and Q. Li, "Shilling attacks against collaborative recommender systems: a review," *Artificial Intelligence Review*, vol. 53, pp. 291–319, sep 2018.

[25] L. Li, T. Xie, and B. Li, "Sok: Certified robustness for deep neural networks," in *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 1289–1310, IEEE, 2023.

[26] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 66–74, 2020.

[27] D. Zhu, Z. Zhang, P. Cui, and W. Zhu, "Robust graph convolutional networks against adversarial attacks," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1399–1407, 2019.

[28] X. Zhang and M. Zitnik, "Gnnguard: Defending graph neural networks against adversarial attacks," *Advances in neural information processing systems*, vol. 33, pp. 9263–9275, 2020.

[29] F. Mujkanovic, S. Geisler, S. Günnemann, and A. Bojchevski, "Are defenses for graph neural networks robust?," *Advances in Neural Information Processing Systems*, vol. 35, pp. 8954–8968, 2022.

[30] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *international conference on machine learning*, pp. 1310–1320, PMLR, 2019.

[31] D. Zügner and S. Günnemann, "Certifiable robustness of graph convolutional networks under structure perturbations," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1656–1665, 2020.

[32] Y. Scholten, J. Schuchardt, S. Geisler, A. Bojchevski, and S. Günnemann, "Randomized message-interception smoothing: Gray-box certificates for graph neural networks," in *Advances in Neural Information Processing Systems*, 2022.

[33] J. Jia, B. Wang, X. Cao, and N. Z. Gong, "Certified robustness of community detection against adversarial structural perturbation via randomized smoothing," in *Proceedings of The Web Conference 2020*, pp. 2718–2724, 2020.

[34] B. Wang, J. Jia, X. Cao, and N. Z. Gong, "Certified robustness of graph neural networks against adversarial structural perturbation," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1645–1653, 2021.

[35] J. Schuchardt, T. Wollschläger, A. Bojchevski, and S. Günnemann, "Localized randomized smoothing for collective robustness certification," in *International Conference on Learning Representations*, 2023.

[36] C. Oliveira, J. Torres, M. I. Silva, D. Aparício, J. T. Ascensão, and P. Bizarro, "Guiltywalker: Distance to illicit nodes in the bitcoin network," *arXiv preprint arXiv:2102.05373*, 2021.

[37] X. Li, Y. Zhuang, Y. Fu, and X. He, "A trust-aware random walk model for return propensity estimation and consumer anomaly scoring in online shopping," *Science China Information Sciences*, vol. 62, pp. 1–17, 2019.

[38] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos, "Neighborhood formation and anomaly detection in bipartite graphs," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pp. 8–pp, IEEE, 2005.

[39] E. Goodman, J. Ingram, S. Martin, and D. Grunwald, "Using bipartite anomaly features for cyber security applications," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pp. 301–306, IEEE, 2015.

[40] H. Cheng, P.-N. Tan, C. Potter, and S. Klooster, "Detection and characterization of anomalies in multivariate time series," in *Proceedings of the 2009 SIAM international conference on data mining*, pp. 413–424, SIAM, 2009.

[41] H. Ren, M. Liu, Z. Li, and W. Pedrycz, "A piecewise aggregate pattern representation approach for anomaly detection in time series," *knowledge-based Systems*, vol. 135, pp. 29–39, 2017.

[42] Z. Yao, P. Mark, and M. Rabbat, "Anomaly detection using proximity graph and pagerank algorithm," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, pp. 1288–1300, 2012.

[43] H. Moonesinghe and P.-N. Tan, "Outlier detection using random walks," in *2006 18th IEEE international conference on tools with artificial intelligence (ICTAI'06)*, pp. 532–539, IEEE, 2006.

[44] G. Pang, L. Cao, and L. Chen, "Outlier detection in complex categorical data by modelling the feature value couplings," in *IJCAI International Joint Conference on Artificial Intelligence*, 2016.

[45] C. Wang, H. Gao, Z. Liu, and Y. Fu, "A new outlier detection model using random walk on local information graph," *IEEE Access*, vol. 6, pp. 75531–75544, 2018.

[46] C. Wang, Z. Liu, H. Gao, and Y. Fu, "Applying anomaly pattern score for outlier detection," *IEEE Access*, vol. 7, pp. 16008–16020, 2019.

[47] K. Zhou, T. P. Michalak, M. Waniek, T. Rahwan, and Y. Vorobeychik, "Attacking similarity-based link prediction in social networks," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 305–313, 2019.

[48] K. Zhou and Y. Vorobeychik, "Robust collective classification against structural attacks," in *Conference on Uncertainty in Artificial Intelligence*, pp. 250–259, PMLR, 2020.

[49] Y. Zhu, Y. Lai, K. Zhao, X. Luo, M. Yuan, J. Ren, and K. Zhou, "Binarizedattack: Structural poisoning attacks to graph-based anomaly detection," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pp. 14–26, IEEE, 2022.

[50] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[51] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*, pp. 39–57, Ieee, 2017.

[52] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1765–1773, 2017.

[53] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *International Conference on Learning Representations*, 2019.

[54] S. Zhang, H. Yin, T. Chen, Q. V. N. Hung, Z. Huang, and L. Cui, "Gcn-based user representation learning for unifying robust recommendation and fraudster detection," in *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pp. 689–698, 2020.

[55] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[56] R. Mehta and K. Rana, "A review on matrix factorization techniques in recommender systems," in *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, pp. 269–274, IEEE, 2017.

[57] C. Wu, D. Lian, Y. Ge, Z. Zhu, E. Chen, and S. Yuan, "Fight fire with fire: Towards robust recommender systems via adversarial poisoning training," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1074–1083, 2021.

[58] J. Tang, X. Du, X. He, F. Yuan, Q. Tian, and T.-S. Chua, "Adversarial training towards robust multimedia recommender system," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 5, pp. 855–867, 2019.

[59] A. Bojchevski, J. Gasteiger, and S. Günnemann, "Efficient robustness certificates for discrete data: Sparsity-aware randomized smoothing for graphs, images and more," in *International Conference on Machine Learning*, pp. 1003–1013, PMLR, 2020.

[60] S. Tao, Q. Cao, H. Shen, Y. Wu, L. Hou, F. Sun, and X. Cheng, "Adversarial camouflage for node injection attack on graphs," *Information Sciences*, vol. 649, p. 119611, 2023.

[61] M. Ju, Y. Fan, C. Zhang, and Y. Ye, "Let graph be the go board: gradient-free node injection attack for graph neural networks via reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 4383–4390, 2023.

[62] A. Levine and S. Feizi, "Robustness certificates for sparse adversarial attacks by randomized ablation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 4585–4593, 2020.

[63] C. You, D. P. Robinson, and R. Vidal, "Provable self-representation based outlier detection in a union of subspaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3395–3404, 2017.

[64] L. Akoglu, M. McGlohon, and C. Faloutsos, "Oddball: Spotting anomalies in weighted graphs," in *Advances in Knowledge Discovery and Data Mining: 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010. Proceedings. Part II 14*, pp. 410–421, Springer, 2010.

[65] Q. Ding, N. Katenka, P. Barford, E. Kolaczyk, and M. Crovella, "Intrusion as (anti) social communication: characterization and detection," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 886–894, 2012.

[66] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, "Fraudar: Bounding graph fraud in the face of camouflage," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 895–904, 2016.

[67] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

[68] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

[69] S. Bandyopadhyay, S. V. Vivek, and M. Murty, "Outlier resistant unsupervised deep architectures for attributed network embedding," in *Proceedings of the 13th international conference on web search and data mining*, pp. 25–33, 2020.

[70] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *Proceedings of the 2019 SIAM International Conference on Data Mining*, pp. 594–602, SIAM, 2019.

[71] T. Zhao, C. Deng, K. Yu, T. Jiang, D. Wang, and M. Jiang, "Error-bounded graph anomaly loss for gnns," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 1873–1882, 2020.

[72] H. Qiao and G. Pang, "Truncated affinity maximization: One-class homophily modeling for graph anomaly detection," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[73] Y. Liu, K. Ding, Q. Lu, F. Li, L. Y. Zhang, and S. Pan, "Towards self-interpretable graph-level anomaly detection," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[74] A. Roy, J. Shu, J. Li, C. Yang, O. Elshocht, J. Smeets, and P. Li, "Gad-nr: Graph anomaly detection via neighborhood reconstruction," in *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pp. 576–585, 2024.

[75] A. Bojchevski and S. Günnemann, "Adversarial attacks on node embeddings via graph poisoning," in *International Conference on Machine Learning*, pp. 695–704, PMLR, 2019.

[76] H. Chang, Y. Rong, T. Xu, W. Huang, H. Zhang, P. Cui, X. Wang, W. Zhu, and J. Huang, "Adversarial attack framework on graph embedding models with limited knowledge," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[77] B. Wang and N. Z. Gong, "Attacking graph-based classification via manipulating the graph structure," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2023–2040, 2019.

[78] S. Zhang, H. Chen, X. Sun, Y. Li, and G. Xu, "Unsupervised graph poisoning attack via contrastive loss back-propagation," in *Proceedings of the ACM Web Conference 2022*, pp. 1322–1330, 2022.

[79] L. Lin, E. Blaser, and H. Wang, "Graph structural attack by perturbing spectral distance," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 989–998, 2022.

[80] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system-a case study," tech. rep., Minnesota Univ Minneapolis Dept of Computer Science, 2000.

[81] F. Simon, "Netflix update: Try this at home.," 2006.

[82] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proceedings of KDD cup and workshop*, vol. 2007, pp. 5–8, 2007.

[83] G. Biau, E. Scornet, and J. Welbl, "Neural random forests," *Sankhya A*, vol. 81, no. 2, pp. 347–386, 2019.

[84] W. Li, M. Gao, H. Li, J. Zeng, Q. Xiong, and S. Hirokawa, "Shilling attack detection in recommender systems via selecting patterns analysis," *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 10, pp. 2600–2611, 2016.

[85] A. P. Sundar, F. Li, X. Zou, T. Gao, and E. D. Russomanno, "Understanding shilling attacks and their detection traits: a comprehensive survey," *IEEE Access*, vol. 8, pp. 171703–171715, 2020.

[86] F. Rezaimehr and C. Dadkhah, "A survey of attack detection approaches in collaborative filtering recommender systems," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 2011–2066, 2021.

[87] C. Wu, D. Lian, Y. Ge, Z. Zhu, and E. Chen, "Triple adversarial learning for influence based poisoning attack in recommender systems," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1830–1840, 2021.

[88] F. Wu, M. Gao, J. Yu, Z. Wang, K. Liu, and X. Wang, "Ready for emerging threats to recommender systems? a graph convolution-based generative shilling attack," *Information Sciences*, vol. 578, pp. 683–701, 2021.

[89] M. Fang, N. Z. Gong, and J. Liu, "Influence function based data poisoning attacks to top-n recommender systems," in *Proceedings of The Web Conference 2020*, pp. 3019–3025, 2020.

[90] H. Huang, J. Mu, N. Z. Gong, Q. Li, B. Liu, and M. Xu, "Data poisoning attacks to deep learning based recommender systems," *In ISOC Network and Distributed System Security Symposium (NDSS)*, 2021.

[91] K. Christakopoulou and A. Banerjee, "Adversarial attacks on an oblivious recommender," in *Proceedings of the 13th ACM Conference on Recommender Systems*, pp. 322–330, 2019.

[92] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[93] X. He, Z. He, X. Du, and T.-S. Chua, "Adversarial personalized ranking for recommendation," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 355–364, 2018.

[94] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, "Classification features for attack detection in collaborative recommender systems," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 542–547, 2006.

[95] H. Yu, H. Zheng, Y. Xu, R. Ma, D. Gao, and F. Zhang, "Detecting group shilling attacks in recommender systems based on maximum dense subtensor mining," in *2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pp. 644–648, IEEE, 2021.

[96] F. Zhang, W. Meng, R. Ma, D. Gao, and S. Wang, "User embedding-based approach for detecting group shilling attacks," in *2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pp. 639–643, IEEE, 2021.

[97] Y. Wang, J. Zhang, S. Guo, H. Yin, C. Li, and H. Chen, "Decoupling representation learning and classification for gnn-based anomaly detection," in *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pp. 1239–1248, 2021.

[98] B. Mehta, "Unsupervised shilling detection for collaborative filtering," in *AAAI*, pp. 1402–1407, 2007.

[99] Z. Zhang and S. R. Kulkarni, "Detection of shilling attacks in recommender systems via spectral clustering," in *17th International Conference on Information Fusion (FUSION)*, pp. 1–8, IEEE, 2014.

[100] Y. Hao and F. Zhang, "An unsupervised detection method for shilling attacks based on deep learning and community detection," *Soft Computing*, vol. 25, no. 1, pp. 477–494, 2021.

[101] Z. Wu, J. Wu, J. Cao, and D. Tao, "Hysad: A semi-supervised hybrid shilling attack detector for trustworthy product recommendation," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 985–993, 2012.

[102] J. Cao, Z. Wu, B. Mao, and Y. Zhang, "Shilling attack detection utilizing semi-supervised learning method for collaborative recommender system," *World Wide Web*, vol. 16, no. 5, pp. 729–748, 2013.

[103] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[104] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*, pp. 1263–1272, PMLR, 2017.

[105] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[106] S. Geisler, D. Zügner, and S. Günnemann, "Reliable graph neural networks via robust aggregation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 13272–13284, 2020.

[107] D. Chen, J. Zhang, Y. Lv, J. Wang, H. Ni, S. Yu, Z. Wang, and Q. Xuan, "Single node injection label specificity attack on graph neural networks via reinforcement learning," *arXiv preprint arXiv:2305.02901*, 2023.

[108] Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar, "Non-target-specific node injection attacks on graph neural networks: A hierarchical reinforcement learning approach," in *Proc. WWW*, vol. 3, 2020.

[109] X. Zou, Q. Zheng, Y. Dong, X. Guan, E. Kharlamov, J. Lu, and J. Tang, "Tdgia: Effective injection attacks on graph neural networks," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2461–2471, 2021.

[110] Y. Chen, Z. Ye, Z. Wang, and H. Zhao, "Imperceptible graph injection attack on graph neural networks," *Complex & Intelligent Systems*, pp. 1–15, 2023.

[111] Y. Zhang, X. Yuan, J. Li, J. Lou, L. Chen, and N.-F. Tzeng, "Reverse attack: Black-box attacks on collaborative recommendation," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 51–68, 2021.

[112] H. Huang, J. Mu, N. Z. Gong, Q. Li, B. Liu, and M. Xu, "Data poisoning attacks to deep learning based recommender systems," in *Network and Distributed System Security Symposium*, 2021.

[113] W. Fan, X. Zhao, Q. Li, T. Derr, Y. Ma, H. Liu, J. Wang, and J. Tang, "Adversarial attacks for black-box recommender systems via copying transferable cross-domain user profiles," *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[114] S. Guo, T. Bai, and W. Deng, "Targeted shilling attacks on gnn-based recommender systems," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 649–658, 2023.

[115] S. Tao, Q. Cao, H. Shen, J. Huang, Y. Wu, and X. Cheng, "Single node injection attack against graph neural networks," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 1794–1803, 2021.

[116] Q. Dai, X. Shen, L. Zhang, Q. Li, and D. Wang, "Adversarial training methods for network embedding," in *The World Wide Web Conference*, pp. 329–339, 2019.

[117] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in *2019 IEEE symposium on security and privacy (SP)*, pp. 656–672, IEEE, 2019.

[118] G.-H. Lee, Y. Yuan, S. Chang, and T. Jaakkola, "Tight certificates of adversarial robustness for randomly smoothed classifiers," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[119] L. Li, J. Zhang, T. Xie, and B. Li, "Double sampling randomized smoothing," in *International Conference on Machine Learning*, pp. 13163–13208, PMLR, 2022.

[120] M. Fischer, M. Baader, and M. Vechev, "Scalable certified segmentation via randomized smoothing," in *International Conference on Machine Learning*, pp. 3340–3351, PMLR, 2021.

[121] J. Jia, B. Wang, X. Cao, H. Liu, and N. Z. Gong, "Almost tight l0-norm certified robustness of top-k predictions against adversarial perturbations," in *International Conference on Learning Representations*, 2022.

[122] Y. Scholten, J. Schuchardt, A. Bojchevski, and S. Günnemann, "Hierarchical randomized smoothing," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[123] M. Horváth, M. Müller, M. Fischer, and M. Vechev, "(de-) randomized smoothing for decision stump ensembles," *Advances in Neural Information Processing Systems*, vol. 35, pp. 3066–3081, 2022.

[124] Z. Hammoudeh and D. Lowd, "Feature partition aggregation: A fast certified defense against a union of $\ell 0$ attacks," in *Proceedings of the 2nd ICML Workshop on New Frontiers in Adversarial Machine Learning. AdvML-Frontiers*, vol. 23, 2023.

[125] Y. Lai, J. Zhou, X. Zhang, and K. Zhou, "Towards certified robustness of graph neural networks in adversarial aiot environments," *IEEE Internet of Things Journal*, 2023.

[126] E. Rosenfeld, E. Winston, P. Ravikumar, and Z. Kolter, "Certified robustness to label-flipping attacks via randomized smoothing," in *International Conference on Machine Learning*, pp. 8230–8241, PMLR, 2020.

[127] B. Wang, X. Cao, N. Z. Gong, *et al.*, "On certifying robustness against backdoor attacks via randomized smoothing," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

[128] M. Weber, X. Xu, B. Karlaš, C. Zhang, and B. Li, "Rab: Provable robustness against backdoor attacks," in *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 1311–1328, IEEE, 2023.

[129] A. Levine and S. Feizi, "Deep partition aggregation: Provable defense against general poisoning attacks," in *International Conference on Learning Representations (ICLR)*, 2021.

[130] J. Schuchardt, A. Bojchevski, J. Gasteiger, and S. Günnemann, "Collective robustness certificates: Exploiting interdependence in graph neural networks," in *International Conference on Learning Representations*, 2020.

[131] Y. Lai, Y. Zhu, B. Pan, and K. Zhou, "Node-aware bi-smoothing: Certified robustness against graph injection attacks," *arXiv preprint arXiv:2312.03979*, 2023.

[132] J. Tang, F. Hua, Z. Gao, P. Zhao, and J. Li, "Gadbench: Revisiting and benchmarking supervised graph anomaly detection," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[133] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," tech. rep., Stanford InfoLab, 1999.

[134] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: defending against sybil attacks via social networks," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 267–278, 2006.

[135] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: A near-optimal social network defense against sybil attacks," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pp. 3–17, IEEE, 2008.

[136] L. Shi, S. Yu, W. Lou, and Y. T. Hou, "Sybilshield: An agent-aided social network-based sybil defense among multiple communities," in *2013 Proceedings IEEE INFOCOM*, pp. 1034–1042, IEEE, 2013.

[137] W. Wei, F. Xu, C. C. Tan, and Q. Li, "Sybildefender: A defense mechanism for sybil attacks in large social networks," *IEEE transactions on parallel and distributed systems*, vol. 24, no. 12, pp. 2492–2502, 2013.

[138] J. Jia, B. Wang, and N. Z. Gong, "Random walk based fake account detection in online social networks," in *2017 47th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, pp. 273–284, IEEE, 2017.

[139] V. N. Ioannidis, D. Berberidis, and G. B. Giannakis, "Unveiling anomalous nodes via random sampling and consensus on graphs," in *ICASSP 2021-2021 IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5499–5503, IEEE, 2021.

[140] J. He, Q. Xu, Y. Jiang, Z. Wang, and Q. Huang, "Ada-gad: Anomaly-denoised autoencoders for graph anomaly detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 8481–8489, 2024.

[141] N. Perra and S. Fortunato, "Spectral centrality measures in complex networks," *Physical Review E*, vol. 78, no. 3, p. 036107, 2008.

[142] P. Boldi, M. Santini, and S. Vigna, "A deeper investigation of pagerank as a function of the damping factor," in *Dagstuhl seminar proceedings*, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2007.

[143] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Combining neural networks with personalized pagerank for classification on graphs," in *International Conference on Learning Representations*, 2019.

[144] S. Casacuberta and R. Kyng, "Faster sparse matrix inversion and rank computation in finite fields," in *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2022.

[145] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in beta-vae," *arXiv preprint arXiv:1804.03599*, 2018.

[146] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*, pp. 413–422, IEEE, 2008.

[147] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, and G. Chen, "Ecod: Unsupervised outlier detection using empirical cumulative distribution functions," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[148] T. T. Müller, S. Starck, A. Dima, S. Wunderlich, K.-M. Bintsi, K. Zaripova, R. Braren, D. Rueckert, A. Kazi, and G. Kaissis, "A survey on graph construction for geometric

deep learning in medicine: Methods and recommendations," *Transactions on Machine Learning Research*, 2023.

[149] L. Qiao, L. Zhang, S. Chen, and D. Shen, "Data-driven graph construction and graph learning: A review," *Neurocomputing*, vol. 312, pp. 336–351, 2018.

[150] S. Bloemheuvel, J. van den Hoogen, and M. Atzmueller, "Graph construction on complex spatiotemporal data for enhancing graph neural network-based approaches," *International Journal of Data Science and Analytics*, pp. 1–18, 2023.

[151] S. Li, W. Yao, Y. Gao, Y. Ma, and B. Liu, "Progressive structure enhancement graph convolutional network for face clustering," *Engineering Applications of Artificial Intelligence*, vol. 127, p. 107274, 2024.

[152] X. Wang and A. Gupta, "Videos as space-time region graphs," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 399–417, 2018.

[153] Y. Qian, P. Expert, P. Panzarasa, and M. Barahona, "Geometric graphs from data to aid classification tasks with graph convolutional networks," *Patterns*, vol. 2, no. 4, 2021.

[154] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: a survey," *ACM Computing Surveys (CSUR)*, 2020.

[155] C. Gao, X. Wang, X. He, and Y. Li, "Graph neural networks for recommender system," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pp. 1623–1625, 2022.

[156] H. Wang, Y. Dou, C. Chen, L. Sun, P. S. Yu, and K. Shu, "Attacking fake news detectors via manipulating news social engagement," in *Proceedings of the ACM Web Conference 2023*, pp. 3978–3986, 2023.

[157] N. Lukas, E. Jiang, X. Li, and F. Kerschbaum, "Sok: How robust is image classification deep neural network watermarking?," in *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 787–804, IEEE, 2022.

[158] A. Kerckhoffs, “La cryptographie militaire,” *J. des Sci. Militaires*, vol. 9, pp. 161–191, 1883.

[159] J. Tang, H. Wen, and K. Wang, “Revisiting adversarially learned injection attacks against recommender systems,” in *Fourteenth ACM conference on recommender systems*, pp. 318–327, 2020.

[160] H. Zhang, C. Tian, Y. Li, L. Su, N. Yang, W. X. Zhao, and J. Gao, “Data poisoning attack against recommender system using incomplete and perturbed data,” in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2154–2164, 2021.

[161] T. Takahashi, “Indirect adversarial attacks via poisoning neighbors for graph convolutional networks,” in *2019 IEEE International Conference on Big Data (Big Data)*, pp. 1395–1400, IEEE, 2019.

[162] S. Oh, B. Ustun, J. McAuley, and S. Kumar, “Rank list sensitivity of recommender systems to interaction perturbations,” in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 1584–1594, 2022.

[163] V. W. Anelli, Y. Deldjoo, T. DiNoia, and F. A. Merra, “Adversarial recommender systems: Attack, defense, and advances,” in *Recommender systems handbook*, pp. 335–379, Springer, 2022.

[164] E. Rolf, N. Malkin, A. Graikos, A. Jojic, C. Robinson, and N. Jojic, “Resolving label uncertainty with implicit posterior models,” *arXiv preprint arXiv:2202.14000*, 2022.

[165] G. Keren, N. Cummins, and B. Schuller, “Calibrated prediction intervals for neural network regressors,” *IEEE Access*, vol. 6, pp. 54033–54041, 2018.

[166] F. Yuan, L. Yao, and B. Benatallah, “Adversarial collaborative neural network for robust recommendation,” in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1065–1068, 2019.

[167] N. Idrissi and A. Zellou, "A systematic literature review of sparsity issues in recommender systems," *Social Network Analysis and Mining*, vol. 10, pp. 1–23, 2020.

[168] X. Li, S. Liu, Z. Li, X. Han, C. Shi, B. Hooi, H. Huang, and X. Cheng, "Flowscope: Spotting money laundering based on graphs," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 4731–4738, 2020.

[169] J. Neyman and E. S. Pearson, *On the Problem of the Most Efficient Tests of Statistical Hypotheses*, pp. 73–108. New York, NY: Springer New York, 1992.

[170] A. Argyriou, M. González-Fierro, and L. Zhang, "Microsoft recommenders: Best practices for production-ready recommendation systems," in *Companion Proceedings of the Web Conference 2020*, pp. 50–51, 2020.

[171] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[172] A. Levine and S. Feizi, "(de) randomized smoothing for certifiable defense against patch attacks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6465–6475, 2020.

[173] A. J. Levine and S. Feizi, "Improved, deterministic smoothing for l_1 certified robustness," in *International Conference on Machine Learning*, pp. 6254–6264, PMLR, 2021.

[174] R. Zhai, C. Dan, D. He, H. Zhang, B. Gong, P. Ravikumar, C.-J. Hsieh, and L. Wang, "Macer: Attack-free and scalable robust training via maximizing certified radius," in *International Conference on Learning Representations*, 2020.

[175] L. Liu, T. N. Hoang, L. M. Nguyen, and T.-W. Weng, "Robust randomized smoothing via two cost-effective approaches," in *ICLR 2022 Workshop on PAIR {\textasciicircum} 2Struct: Privacy, Accountability, Interpretability, Robustness, Reasoning on Structured Data*, 2022.

[176] J. Jeong, S. Park, M. Kim, H.-C. Lee, D.-G. Kim, and J. Shin, "Smoothmix: Training confidence-calibrated smoothed classifiers for certified robustness," *Advances in Neural Information Processing Systems*, vol. 34, pp. 30153–30168, 2021.

[177] L. Gosch, S. Geisler, D. Sturm, B. Charpentier, D. Zügner, and S. Günnemann, "Adversarial training for graph neural networks: Pitfalls, solutions, and new directions," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[178] R. Chen, Z. Li, J. Li, J. Yan, and C. Wu, "On collective robustness of bagging against data poisoning," in *International Conference on Machine Learning*, pp. 3299–3319, PMLR, 2022.

[179] W. Wei, "Tutorials on advanced optimization methods," *arXiv preprint arXiv:2007.13545*, 2020.

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

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

[182] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[183] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019.

[184] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[185] Z. Liu, X. Yu, Y. Fang, and X. Zhang, "Graphprompt: Unifying pre-training and downstream tasks for graph neural networks," in *Proceedings of the ACM web conference 2023*, pp. 417–428, 2023.

[186] X. Sun, H. Cheng, J. Li, B. Liu, and J. Guan, "All in one: Multi-task prompting for graph neural networks," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2120–2131, 2023.

[187] Y. Xu, X. Liu, K. Duan, Y. Fang, Y.-N. Chuang, D. Zha, and Q. Tan, "Graphfm: A comprehensive benchmark for graph foundation model," *arXiv preprint arXiv:2406.08310*, 2024.

[188] Y. Tian, H. Song, Z. Wang, H. Wang, Z. Hu, F. Wang, N. V. Chawla, and P. Xu, "Graph neural prompting with large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 19080–19088, 2024.

[189] H. Han, Y. Xie, H. Liu, X. Tang, S. Nag, W. Headden, Y. Li, C. Luo, S. Ji, Q. He, *et al.*, "Reasoning with graphs: Structuring implicit knowledge to enhance llms reasoning," *arXiv preprint arXiv:2501.07845*, 2025.

[190] Q. Zhang, S. Chen, Y. Bei, Z. Yuan, H. Zhou, Z. Hong, J. Dong, H. Chen, Y. Chang, and X. Huang, "A survey of graph retrieval-augmented generation for customized large language models," *arXiv preprint arXiv:2501.13958*, 2025.

[191] J. Liang, Y. Wang, C. Li, R. Zhu, T. Jiang, N. Gong, and T. Wang, "Graphrag under fire," *arXiv preprint arXiv:2501.14050*, 2025.

Appendix A

Appendix of Node-aware Smoothing

A.1 Theoretical Proofs

Theorem 5. *(Restate) Let $f : \mathbb{G} \longrightarrow \{1, \cdots, C\}^n$ be any graph classifier, g be its smoothed classifier defined in* (5.2) *with $\phi(G) = (\phi_e(G), \phi_n(G))$, $v \in G$ be any query node, $B_{\rho,\tau}(G)$ be the node injection perturbation set defined in* (5.1)*. Suppose $y_A, y_B \in \{1, \cdots, C\}$ and $\underline{p_A}, \overline{p_B} \in [0, 1]$. Then we have $g_v(G') = g_v(G)$, $\forall G' \in B_{\rho,\tau}(G)$, if:*

$$\mu_{\rho,\tau} := \tilde{p}(\underline{p_A} - \overline{p_B} + 1) - 1 > 0,$$

where $\tilde{p} := (p_n + (1 - p_n)(p_e + p_n - p_e p_n)^\tau)^\rho$.

Proof. To solve the certifying problem defined in (5.3), we need to calculate the likelihood ratio of $\phi(A)$ and $\phi(A')$. Let $\Lambda(Z) = \frac{\mathbb{P}(\phi(A)=Z)}{\mathbb{P}(\phi(A')=Z)}$ be the likelihood ratio, where $Z \in \mathbb{G}$ is any possible graph produced by $\phi(A)$ or $\phi(A')$. However, the difficulty lies in that the $\phi(A)$ and $\phi(A')$ are of different dimensions, which makes the probability hard to obtain. To tackle the challenge, we propose a straightforward strategy by pre-injecting ρ isolated nodes in the clean graph A, such that the adjacency matrix A has the same dimension as A'. Furthermore, in order to maintain the dimension of $\phi(A)$, we construct an equivalent setting for node deletion smoothing. If a node v is deleted in the smoothing $\phi(\cdot)$, we delete all the edges incident to that node and keep the

isolated node v, which is equivalent to setting the v^{th} raw $A_{v:}$ to zeros. Note that the prediction for a graph will not be affected by the isolated nodes since it does not provide any information to the existing nodes. Thus, in this setting, all the graphs involved in the computation are in the same dimension.

According to [59], the likelihood ratio Λ is only depends on the bits $C := \{(i,j)|A_{ij} \neq A'_{ij}\}$ which is the set of index that $A_{ij} \neq A'_{ij}$. Under node injection perturbation, the A' and A are only different among the submatrix $A_{n:(n+\rho),1:(n+\rho)}$ (the raws of injected nodes), and they have exactly $\rho \cdot \tau$ different bits. That is $|C| = \rho \cdot \tau$. Since the smoothing randomization ϕ will not add any edge, bits in $\phi(A)_{n:(n+\rho),1:(n+\rho)}$ are always zeros. So that $\Lambda(Z) = \frac{\mathbb{P}(\phi(A)=Z)}{\mathbb{P}(\phi(A')=Z)} > 0$ if and only if all injected edges in $A'_{n:(n+\rho),1:(n+\rho)}$ are set to zeros by $\phi(A')$, and we define such a region as $\mathcal{R}_1 = \{Z|\, Z_{n:(n+\rho),1:(n+\rho)} = 0\}$, while the other region as $\mathcal{R}_2 = \{Z|\, Z_{n:(n+\rho),1:(n+\rho)} \neq 0\}$. We have:

$$\Lambda(Z) = \frac{\mathbb{P}(\phi(A) = Z)}{\mathbb{P}(\phi(A') = Z)} = \begin{cases} 1/\tilde{p}, & \text{if } Z \in \mathcal{R}_1, \\ 0, & \text{if } Z \in \mathcal{R}_2. \end{cases} \tag{A.1}$$

The meaning of the region $\mathcal{R}_1 = \{Z|\, Z_{n:(n+\rho),1:(n+\rho)} = 0\}$ is to set all the injected nodes isolated from others. If $\phi(A')$ set an injected node $\tilde{v}$ isolated, the $\phi(A')$ deleted all the incident edges to node $\tilde{v}$, or $\phi(A')$ delete the node $\tilde{v}$. For an injected node, The probability of deleting the node itself in node deletion smoothing is p_n. If it is not deleted, each of the edges has a probability of $p_n + (1-p_n)p_e = p_n + p_e - p_n p_e$ being deleted. Because the edge connects to other existing nodes, deleting other nodes also deletes the edge connects to the node (Figure 5.2, bottom). If an edge is not deleted by this, it has a probability of p_e being deleted in edge deletion smoothing. Since there are ρ injected nodes, and τ injected edges for each injected node, the probability of $\phi(A') \in \mathcal{R}_1$ is $\tilde{p} = (p_n + (1-p_n)(p_e + p_n - p_e p_n)^\tau)^\rho$. Specifically, the corresponding probabilities for the two constant likelihood ratio regions are:

$$\begin{cases} \mathbb{P}(\phi(A) \in \mathcal{R}_1) = 1, \\ \mathbb{P}(\phi(A) \in \mathcal{R}_2) = 0, \end{cases}$$

$$\begin{cases} \mathbb{P}(\phi(A') \in \mathcal{R}_1) = \tilde{p}, \\ \mathbb{P}(\phi(A') \in \mathcal{R}_2) = 1 - \tilde{p}. \end{cases}$$

The worst-case classifier defined in problem (5.3) will assign class y_A in decreasing order ($\Lambda(Z \in \mathcal{R}_1) > \Lambda(Z \in \mathcal{R}_2)$) of the constant likelihood regions until $\mathbb{P}(f_v(\phi(A)) = y_A) = \underline{p_A}$, and assign class y_B in increasing order ($\Lambda(Z \in \mathcal{R}_2) < \Lambda(Z \in \mathcal{R}_1)$) of the constant likelihood regions until $\mathbb{P}(f_v(\phi(A)) = y_B) = \overline{p_B}$. Therefore, the worst-case classifier is:

$$\mathbb{P}(f_v(Z) = y_A) = \begin{cases} \underline{p_A}, & Z \in \mathcal{R}_1, \\ 0, & Z \in \mathcal{R}_2, \end{cases}$$

$$\mathbb{P}(f_v(Z) = y_B) = \begin{cases} \overline{p_B}, & Z \in \mathcal{R}_1, \\ 1, & Z \in \mathcal{R}_2. \end{cases}$$

Under this classifier, we can verify that:

$$\begin{aligned} & \mathbb{P}(f_v(\phi(A)) = y_A) \\ = \; & \mathbb{P}(\phi(A) = Z \in \mathcal{R}_1)\mathbb{P}(f_v(Z) = y_A | Z \in \mathcal{R}_1) \\ & + \mathbb{P}(\phi(A) = Z \in \mathcal{R}_2)\mathbb{P}(f_v(Z) = y_A | Z \in \mathcal{R}_2) \\ = \; & 1 \cdot \underline{p_A} + 0 \cdot 0 = \underline{p_A}. \end{aligned}$$

$$\begin{aligned} & \mathbb{P}(f_v(\phi(A)) = y_B) \\ = \; & \mathbb{P}(\phi(A) = Z \in \mathcal{R}_1)\mathbb{P}(f_v(Z) = y_B | Z \in \mathcal{R}_1) \\ & + \mathbb{P}(\phi(A) = Z \in \mathcal{R}_2)\mathbb{P}(f_v(Z) = y_B | Z \in \mathcal{R}_2) \\ = \; & 1 \cdot \underline{p_B} + 0 \cdot 1 = \overline{p_B}. \end{aligned}$$

With this worst-case classifier, we can obtain the worst-case classification margin under $\phi(A')$, which we denoted as $\mu_{\rho,\tau}$:

$$\begin{aligned} & \mathbb{P}(f_v(\phi(A')) = y_A) \\ = \; & \mathbb{P}(\phi(A') = Z \in \mathcal{R}_1)\mathbb{P}(f_v(Z) = y_A | Z \in \mathcal{R}_1) \\ & + \mathbb{P}(\phi(A') = Z \in \mathcal{R}_2)\mathbb{P}(f_v(Z) = y_A | Z \in \mathcal{R}_2) \\ = \; & \tilde{p} \cdot \underline{p_A} + (1 - \tilde{p}) \cdot 0 \\ = \; & \tilde{p}\, \underline{p_A}. \end{aligned}$$

$$
\begin{aligned}
&\mathbb{P}(f_v(\phi(A')) = y_B) \\
=\ &\mathbb{P}(\phi(A') = Z \in \mathcal{R}_1)\mathbb{P}(f_v(Z) = y_B | Z \in \mathcal{R}_1) \\
&+ \mathbb{P}(\phi(A') = Z \in \mathcal{R}_2)\mathbb{P}(f_v(Z) = y_B | Z \in \mathcal{R}_2) \\
=\ &\tilde{p}\,\overline{p_B} + 1 - \tilde{p}.
\end{aligned}
$$

$$
\begin{aligned}
\mu_{\rho,\tau} &= \mathbb{P}(f_v(\phi(A')) = y_A) - \mathbb{P}(f_v(\phi(A')) = y_B) \\
&= \tilde{p}\,\underline{p_A} - (\tilde{p}\,\overline{p_B} + 1 - \tilde{p}) \\
&= \tilde{p}(\underline{p_A} - \overline{p_B} + 1) - 1.
\end{aligned}
$$

If $\mu_{\rho,\tau} > 0$, we can certify the prediction of $g_v(A') = y_A$. Otherwise, we cannot certify the prediction of $g_v(A')$. □

Theorem 6. *(Restate) Let $f : \mathbb{G} \longrightarrow \{1, \cdots, C\}^n$ be any graph classifier, g be its smoothed classifier defined in* (5.5) *with $\phi(G) = (\phi_e(G), \phi_n(G))$, $v \in G$ be any query node, $B_{\rho,\tau}(G)$ be the node injection perturbation set defined in* (5.1)*, and the attack edges added to a node v should not exceed its original degree $d(v)$. Suppose $y_A, y_B \in \{1, \cdots, C\}$ and $\underline{p_A}, \overline{p_B} \in [0, 1]$. Then we have $g_v(G') = g_v(G)$, $\forall G' \in B_{\rho,\tau}(G)$, if:*

$$
\mu_{\rho,\tau} := \tilde{p}(\underline{p_A} - \frac{(1 - \underline{p_0'})\overline{p_B}}{(1 - p_0)} + 1 - \underline{p_0'}) - (1 - \underline{p_0'}) > 0,
$$

where $\tilde{p} := (p_n + (1 - p_n)(p_e + p_n - p_e p_n)^\tau)^\rho$, $d(v)$ denotes the degree of node v, and $p_0 := p_n + (1 - p_n)(p_e + p_n - p_e p_n)^{d(v)}$ is the probability that the node v is deleted by the smoothing $\phi(G)$, $\underline{p_0'} := p_n + (1 - p_n)(p_e + p_n - p_e p_n)^{2d(v)}$.

Proof. Let Z be any possible graph from $\phi(G)$ or $\phi(G')$, $v \vdash Z$ denote $v \in Z$ is not isolated, we now need to compute the likelihood ratio with $v \vdash \phi(G)$:

$$
\begin{aligned}
\Lambda(Z) &= \frac{\mathbb{P}(\phi(A) = Z, v \vdash Z)}{\mathbb{P}(\phi(A') = Z, v \vdash Z)} \\
&= \begin{cases} \frac{1 - p_0}{\tilde{p}(1 - p_0')}, & \text{if } Z \in \mathcal{R}_1\,(Z_{n:(n+\rho),1:(n+\rho)} = 0), \\ 0, & \text{if } Z \in \mathcal{R}_2\,(Z_{n:(n+\rho),1:(n+\rho)} \neq 0). \end{cases}
\end{aligned} \tag{A.2}
$$

Specifically, the corresponding probabilities for the two constant likelihood ratio regions are:

$$\begin{cases} \mathbb{P}(\phi(A) \in \mathcal{R}_1, v \vdash \phi(A)) = 1 - p_0, \\ \mathbb{P}(\phi(A) \in \mathcal{R}_2, v \vdash \phi(A)) = 0, \end{cases}$$

$$\begin{cases} \mathbb{P}(\phi(A') \in \mathcal{R}_1, v \vdash \phi(A')) = \tilde{p}(1 - p_0'), \\ \mathbb{P}(\phi(A') \in \mathcal{R}_2, v \vdash \phi(A')) = (1 - \tilde{p})(1 - p_0'), \end{cases}$$

where $p_0 := p_n + (1 - p_n)(p_e + p_n - p_e p_n)^{d(v)}$, $p_0' = p_n + (1 - p_n)(p_e + p_n - p_e p_n)^{d(v')}$ denotes the probability that the node v is deleted by the smoothing $\phi(G)$ and $\phi(G')$ respectively; $d(v)$, $d(v')$ denotes the degree of node v in G and G', respectively. Similarly, the worst-case classifier is:

$$\mathbb{P}(f_v(Z) = y_A) = \begin{cases} \frac{p_A}{(1-p_0)}, & Z \in \mathcal{R}_1, v \vdash Z, \\ 0, & Z \in \mathcal{R}_2, v \vdash Z, \end{cases}$$

$$\mathbb{P}(f_v(Z) = y_B) = \begin{cases} \frac{\overline{p_B}}{(1-p_0)}, & Z \in \mathcal{R}_1, v \vdash Z, \\ 1, & Z \in \mathcal{R}_2, v \vdash Z. \end{cases}$$

With this worst-case classifier, we can obtain the worst-case classification margin under $\phi(A')$, which we denoted as $\mu_{\rho,\tau}$:

$$\begin{aligned} & \mathbb{P}(f_v(\phi(A')) = y_A) \\ = \; & \mathbb{P}(f_v(\phi(A')) = y_A, v \vdash Z) \\ = \; & \mathbb{P}(\phi(A') = Z \in \mathcal{R}_1, v \vdash Z) \\ & \times \mathbb{P}(f_v(Z) = y_A | Z \in \mathcal{R}_1, v \vdash Z) \\ & + \mathbb{P}(\phi(A') = Z \in \mathcal{R}_2, v \vdash Z) \\ & \times \mathbb{P}(f_v(Z) = y_A | Z \in \mathcal{R}_2, v \vdash Z) \\ = \; & \tilde{p}(1 - p_0') \cdot \frac{\underline{p_A}}{(1 - p_0)} \\ \geq \; & \tilde{p} \cdot \underline{p_A}, \end{aligned}$$

where the inequality is due to $\frac{(1-p_0')}{(1-p_0)} \geq 1$, because the node degree of v in the perturbed graph must be larger than in the clean graph: $d(v') \geq d(v)$. With the assumption that $d(v') \leq 2d(v)$,

we have $p'_0 > \underline{p'_0} := p_n + (1-p_n)(p_e + p_n - p_e p_n)^{2d(v)}$, and $(1-p'_0) \leq (1-\underline{p'_0})$:

$$\begin{aligned}
&\mathbb{P}(f_v(\phi(A')) = y_B) \\
=\ &\mathbb{P}(\phi(A') = Z \in \mathcal{R}_1, v \vdash Z) \\
&\times \mathbb{P}(f_v(Z) = y_B | Z \in \mathcal{R}_1, v \vdash Z) \\
&+ \mathbb{P}(\phi(A') = Z \in \mathcal{R}_2, v \vdash Z) \\
&\times \mathbb{P}(f_v(Z) = y_B | Z \in \mathcal{R}_2, v \vdash Z) \\
=\ &\tilde{p}(1-p'_0) \cdot \frac{\overline{p_B}}{(1-p_0)} + (1-\tilde{p})(1-p'_0) \\
\leq\ &\tilde{p} \cdot \overline{p_B} \cdot \frac{(1-\underline{p'_0})}{(1-p_0)} + (1-\tilde{p})(1-\underline{p'_0}),
\end{aligned}$$

where the inequality is due to $(1-p'_0) \leq (1-\underline{p'_0})$. Then, we can obtain a lower bound of the worst-case classification margin under $\phi(A')$:

$$\begin{aligned}
&\mathbb{P}(f_v(\phi(A')) = y_A) - \mathbb{P}(f_v(\phi(A')) = y_B) \\
\geq\ &\tilde{p} \cdot \underline{p_A} - [\tilde{p} \cdot \overline{p_B} \cdot \frac{(1-p_n)}{(1-p_0)} + (1-\tilde{p})(1-\underline{p'_0})] \\
=\ &\tilde{p}(\underline{p_A} - \overline{p_B} \cdot \frac{(1-\underline{p'_0})}{(1-p_0)}) - 1 + \underline{p'_0} + \tilde{p} - \tilde{p}\underline{p'_0} \\
=\ &\tilde{p}(\underline{p_A} - \frac{(1-\underline{p'_0})\overline{p_B}}{(1-p_0)} + 1 - \underline{p'_0}) - (1-\underline{p'_0}) := \mu_{\rho,\tau}.
\end{aligned}$$

If $\mu_{\rho,\tau} > 0$, we can certify the prediction of $g_v(A') = y_A$. Otherwise, we cannot certify the prediction of $g_v(A')$. □

Theorem 7. *(Restate) Let $F_u(G)$ be any base recommender system trained on G and recommend K' items to the user u, $g_u(G)$ be its smoothed recommender defined in* (5.7)*, $u \in G$ be any query user, $B_{\rho,\tau}(G)$ be the node injection perturbation set defined in* (5.1)*, and the attack edges added to a node v should not exceed its original degree $d(v)$. Then, we have at least r recommended items after poisoning are overlapped with ground truth items I_u: $|g_u(G') \cap I_u| \geq r, \forall G' \in B_{\rho,\tau}(G)$ if:*

$$\hat{p}\,\underline{p_r} - \min_{H_c}(\overline{p}_{H_c} + K'(1-\hat{p})(1-p_0))/c > 0,$$

where $\hat{p} := (p_n + (1-p_n)p_e^{\tau})^{\rho}$, $\underline{p_r}$ is the lower bound of the rth largest item probability among $\{p_{u,i} | i \in I_u\}$, H_c denote any subset of the top-$(K-r+1)$ largest items among $I \setminus I_u$ with

size c, $\overline{p}_{H_c} := \sum_{j \in H_c} \overline{p_{u,j}}$ is the sum of probability upper bounds for c items in H_c, $p_0 := p_n + (1-p_n)(p_e)^{d(u)}$ is the probability that the user u is deleted by the smoothing $\phi(G)$, $d(u)$ is the number of user ratings in training set.

Proof. For a user-item interaction graph G, we represent it as a user-item interaction matrix denoted by A, where each row represents a user and each column represents an item, and the elements $A_{ui} = 1$ if the interaction exists between user u and item i, otherwise $A_{ui} = 0$. If there are n users and m items in the training set, we have the shape of A of $n \times m$. Let I denote all the items in the training set, we have $|I| = m$. Let Z be any possible matrix from $\phi(A)$ or $\phi(A')$, $u \vdash Z$ denote $u \in Z$ has at least one rating, we have the likelihood ratio with $u \vdash \phi(G)$:

$$\begin{aligned} \Lambda(Z) &= \frac{\mathbb{P}(\phi(A) = Z, u \vdash Z)}{\mathbb{P}(\phi(A') = Z, u \vdash Z)} \\ &= \begin{cases} 1/\hat{p}, & \text{if } Z \in \mathcal{R}_1 \, (Z_{n:(n+\rho),1:m} = 0), \\ 0, & \text{if } Z \in \mathcal{R}_2 \, (Z_{n:(n+\rho)),1:m} \neq 0). \end{cases} \end{aligned} \tag{A.3}$$

Specifically, the corresponding probabilities for the two constant likelihood ratio regions are:

$$\begin{cases} \mathbb{P}(\phi(A) \in \mathcal{R}_1, u \vdash \phi(A)) = 1 - p_0, \\ \mathbb{P}(\phi(A) \in \mathcal{R}_2, u \vdash \phi(A)) = 0, \end{cases}$$

$$\begin{cases} \mathbb{P}(\phi(A') \in \mathcal{R}_1, u \vdash \phi(A')) = \hat{p}(1 - p_0), \\ \mathbb{P}(\phi(A') \in \mathcal{R}_2, u \vdash \phi(A')) = (1 - \hat{p})(1 - p_0), \end{cases}$$

where $p_0 := p_n + (1 - p_n)(p_e)^{d(u)}$ denotes the probability that the node u is deleted by the smoothing $\phi(G)$, and $d(u)$ denotes the degree (number of rating) of user u in G. Note that the newly injected user will not increase the degree of the existing user, the $\phi(A')$, so that $\mathbb{P}(u \vdash \phi(A'))$ is also $(1 - p_0)$. Similarly, if we know the lower bound of $p_{u,i} := \mathbb{P}(i \in F_u(\phi(G)))$, then the worst-case classifier returns the smallest $p'_{u,i} := \mathbb{P}(i \in F_u(\phi(G')))$ is:

$$\mathbb{P}(i \in F_u(Z)) = \begin{cases} \frac{p_{u,i}}{(1-p_0)}, & Z \in \mathcal{R}_1, u \vdash Z, \\ 0, & Z \in \mathcal{R}_2, u \vdash Z, \end{cases}$$

On the contrary, if we know the upper bound of $p_{u,j} := \mathbb{P}(j \in F_u(\phi(G)))$, then the worst-case classifier returns the largest $p'_{u,j} := \mathbb{P}(j \in F_u(\phi(G')))$ is:

$$\mathbb{P}(j \in F_u(Z)) = \begin{cases} \frac{\overline{p_{u,j}}}{(1-p_0)}, & Z \in \mathcal{R}_1, u \vdash Z, \\ 1, & Z \in \mathcal{R}_2, u \vdash Z. \end{cases}$$

For the rth items under the clean graph, we denote its lower bound of probability as $\underline{p_r}$, and then we have the bound for its probability under the poisoned graph: $p'_r \geq \hat{p}\,\underline{p_r}$. We have at least r recommended items overlapped with ground truth items I_u if the rth largest item probability among items I_u is larger than the $(K-r+1)$th largest items probability among $I \setminus I_u$ under the poisoned graph. We denote the top $(K-r+1)$ largest items by their probability among $I \setminus I_u$ as a set $\mathbf{I}_{kr}$. Following [2], instead of considering the $(K-r+1)$th item (the smallest one in $\mathbf{I}_{kr}$), jointly considering multiple items H_c usually leads to a smaller upper bound, where H_c is the subset of $\mathbf{I}_{kr}$ with size c. For the c items, we know its summation of upper bound: $\overline{p}_{H_c} := \sum_{j \in H_c} \overline{p_{u,j}}$. Because each of the system recommender K' items, we have $\overline{p}_{H_c} \leq K'$ (i.e., $\frac{\overline{p}_{H_c}}{K'} \leq 1$). Then we have the upper bound for p'_{H_c}:

$$p'_{H_c} = \sum_{j \in H_c} p'_{u,j} \leq \hat{p} \cdot \overline{p}_{H_c} + K'(1-\hat{p})(1-p_0). \tag{A.4}$$

Then, because the minimum value of a set is always smaller than the average value, we have:

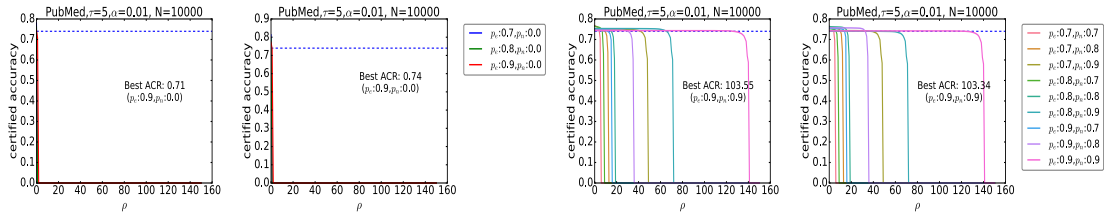
$$\begin{aligned} \min_{j \in \mathbf{I}_{kr}} p'_{u,j} \leq \min_{j \in H_c} p'_{u,j} &\leq \frac{\sum_{j \in H_c} p'_{u,j}}{c} \\ &\leq (\hat{p} \cdot \overline{p}_{H_c} + K'(1-\hat{p})(1-p_0))/c. \end{aligned} \tag{A.5}$$

Finally, we have at least r recommended items overlapped with ground truth items I_u if: $\hat{p}\,\underline{p_r} - \min_{H_c}(\overline{p}_{H_c} + K'(1-\hat{p})(1-p_0))/c > 0$.

□

A.2 Other Experimental Results

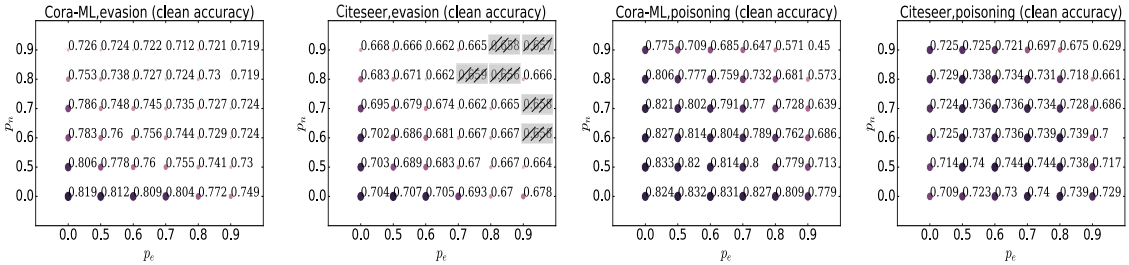
In this section, we display the supplemental experimental results. To ensure a broad evaluation and to be more consistent with related work [59], we extend our evaluation on base model Graph



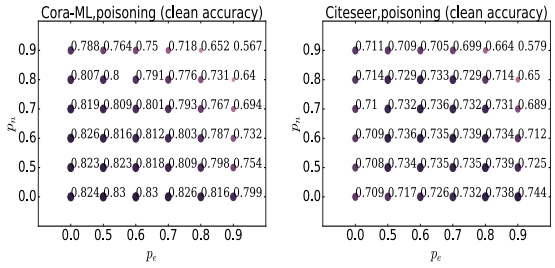
(a) Sparsity-aware[59] (b) Sparsity-aware[59] (c) **Node-aware**-exclude (d) **Node-aware**-exclude

Figure A.1: Certified accuracy under *evasion* perturbation with base model GCN (left) and GAT (right).

Attention Network (GAT) [182] and PubMed [181] dataset. The results shown in Figure. A.1 are consistent with those in the main thesis.



(a) Evasion (b) Evasion (c) Poisoning (include) (d) Poisoning (include)



(e) Poisoning (exclude) (f) Poisoning (exclude)

Figure A.2: Clean accuracy of node-aware bi-smoothing classifiers with various parameters under evasion and poisoning setting.

Figure. A.2 shows the clean accuracy of the smoothed classifier under various smoothing parameters p_e and p_n. For the evasion attack, the Multi-layer Perceptron (MLP) will never affected because it does not rely on the graph structure for prediction. As a result, a graph model with a

lower accuracy is not meaningful. In the Cora-ML dataset, the clean accuracy of our smoothed classifiers is significantly higher than 0.691, which are all effective. However, due to the smaller average degree of the Citeseer dataset, our smoothed classifiers might have a lower clean accuracy than MLP. We exclude the parameters (shadow) that lead to lower accuracy than the MLP model, which is 0.691 on Cora-ML and 0.660 on the Citeseer dataset.

Note that the MLP model is also subject to poisoning GIA since the malicious node feature can be crafted arbitrarily (all the results are effective). Notably, our smoothed model in the poisoning setting improves the clean accuracy.

Appendix B

Appendix of Collective Certified Robustness

B.1 Theorectical Proofs

Lemma 8. *(Restate) Let A be the adjacency matrix of the perturbed graph with ρ injected nodes, and the injected nodes are in the last ρ rows and columns. With smoothing $p_n > 0$ and $p_e > 0$, we have the upper bound of $p(E_v)$:*

$$\begin{aligned} p(E_v) &\leq \overline{p(E_v)} \\ &= 1 - p_1^{||A_{n:(n+\rho),v}||_1} p_2^{||A^2_{n:(n+\rho),v}||_1} \cdots p_k^{||A^k_{n:(n+\rho),v}||_1}, \end{aligned} \tag{B.1}$$

where $p_i := 1 - (\bar{p}_e \bar{p}_n)^i$, $\forall i \in \{1, 2, \cdots, k\}$, and adjacency matrix A contains the injected nodes encoded in the $(n+1)^{th}$ to $(n+\rho)^{th}$ row, and $||\cdot||_1$ is l_1 norm.

Proof. According to [32], we have an upper bound for $p(E_v) \leq \overline{p(E_v)}$ by assuming the independence among the paths. Let $p(\bar{E}_v^{\tilde{v}})$ denote the probability that all paths are intercepted from an injected node $\tilde{v}$ to node v in the case that of considering each path independently. We have $p(\bar{E}_v^{\tilde{v}}) = \prod_{q \in P^k_{\tilde{v}v}} (1 - (\bar{p}_e \bar{p}_n)^{|q|})$, where $\bar{p}_e := 1 - p_e$, $\bar{p}_n := 1 - p_n$ and $|q| \in \{1, \cdots, k\}$ represent the length of the path $q \in P^k_{\tilde{v}v}$ from $\tilde{v}$ to v. $(\bar{p}_e \bar{p}_n)^{|q|}$ is the probability that all edges and all nodes

in the path q are not deleted, $1-(\bar{p}_e\bar{p}_n)^{|q|}$ is the probability that at least one of edges or one of nodes are deleted, such that the path q is intercepted. Then, by considering multiple injected nodes, we have $\overline{p(E_v)} = 1 - \prod_{\tilde{v}\in\tilde{\mathcal{V}}} p(\bar{E}_v^{\tilde{v}})$. Finally, we have the $\overline{p(E_v)}$ as follows:

$$
\begin{aligned}
&\overline{p(E_v)} \\
&= 1 - \prod_{\tilde{v}\in\tilde{\mathcal{V}}} p(\bar{E}_v^{\tilde{v}}) \\
&= 1 - \prod_{\tilde{v}\in\tilde{\mathcal{V}}} \{ \prod_{q\in P_{\tilde{v}v}^k} (1-(\bar{p}_e\bar{p}_n)^{|q|}) \} \\
&= 1 - \prod_{\tilde{v}\in\tilde{\mathcal{V}}} \{ (1-\bar{p}_e\bar{p}_n)^{A_{\tilde{v}v}} (1-(\bar{p}_e\bar{p}_n)^2)^{A_{\tilde{v}v}^2} \cdots (1-(\bar{p}_e\bar{p}_n)^k)^{A_{\tilde{v}v}^k} \} \\
&= 1 - p_1^{||A_{n:(n+\rho),v}||_1} p_2^{||A^2_{n:(n+\rho),v}||_1} \cdots p_k^{||A^k_{n:(n+\rho),v}||_1},
\end{aligned}
\tag{B.2}
$$

where $p_i := 1-(\bar{p}_e\bar{p}_n)^i$. In particular, the constant p_k denotes the probability that a path with a length of k is intercepted. According to graph theory, $A_{\tilde{v}v}^k$ is the number of paths from node $\tilde{v}$ to node v with distance/length/steps of exactly k in the graph. Let $A_{n:(n+\rho),v}$ denote the slicing of matrix A, taking the v^{th} column and the rows from $(n+1)^{th}$ to $(n+\rho)^{th}$. Then $||A^k_{n:(n+\rho),v}||_1$ quantifies the number of paths with a length of k originating from any malicious node and reaching node v. □

Theorem 10. *(Restate) Given a base GNN classifier f trained on a graph G and its smoothed classifier g defined in* (5.2)*, a testing node $v \in G$ and a perturbation range $B_{\rho,\tau}(G)$, let E_v be the event defined in Eq.* (5.14)*. The absolute change in predicted probability $|p_{v,y}(G)-p_{v,y}(G')|$ for all perturbed graphs $G' \in B_{\rho,\tau}(G)$ is bounded by the probability of the event E_v: $|p_{v,y}(G) - p_{v,y}(G')| \le p(E_v)$.*

Proof. By the law of total probability, we have

$$
\begin{aligned}
&\mathbb{P}(f_v(\phi(G')) = y) \\
&= \mathbb{P}(f_v(\phi(G')) = y \wedge E_v) + \mathbb{P}(f_v(\phi(G')) = y \wedge \bar{E}_v).
\end{aligned}
$$

Note that, we define the event E_v based on the sampling of perturbed graph $\phi(G')$. However, the clean graph G is smaller than G', and the intersection/overlap graph of them is $G \cap G' = G$.

Subtly, we can still use the event E_v defined on $\phi(G')$ to divide the sample space of $\phi(G)$ by regarding the model $f_v(\phi(G))$ only take part of the $\phi(G')$ as input, which is the intersected part of G: $\phi(G') \cap G$, and the result does not relate to the part that beyond G (i.e., the injected nodes). Such that, we also have

$$\begin{aligned} &\mathbb{P}(f_v(\phi(G)) = y) \\ =\ &\mathbb{P}(f_v(\phi(G)) = y \wedge E_v) + \mathbb{P}(f_v(\phi(G)) = y \wedge \bar{E}_v). \end{aligned}$$

Due to the fact that the injected node does not have any message passing to v would not affect the $p_{v,y}(G)$, we have $\mathbb{P}(f_v(\phi(G')) = y|\bar{E}_v) = \mathbb{P}(f_v(\phi(G)) = y|\bar{E}_v)$, so that $\mathbb{P}(f_v(\phi(G)) = y \wedge \bar{E}_v) = \mathbb{P}(f_v(\phi(G')) = y \wedge \bar{E}_v)$. Following [32], we have similar deduction as follows:

$$\begin{aligned} &p_{v,y}(G) - p_{v,y}(G') \\ =\ &\mathbb{P}(f_v(\phi(G)) = y \wedge E_v) + \mathbb{P}(f_v(\phi(G)) = y \wedge \bar{E}_v) \\ &- \mathbb{P}(f_v(\phi(G')) = y \wedge E_v) - \mathbb{P}(f_v(\phi(G')) = y \wedge \bar{E}_v) \\ =\ &\mathbb{P}(f_v(\phi(G)) = y \wedge E_v) - \mathbb{P}(f_v(\phi(G')) = y \wedge E_v) \\ \leq\ &\mathbb{P}(f_v(\phi(G)) = y \wedge E_v) \\ =\ &p(E_v) \cdot \mathbb{P}(f_v(\phi(G)) = y|E_v) \\ \leq\ &p(E_v). \end{aligned}$$

□

B.2 Details of Optimization Formulation

B.2.1 Formulating problem (5.18) as polynomial constrained programming.

For problem (5.18), we plug in $\overline{p(E_v)}$ with (5.15), and then we have the following optimization problem:

$$\begin{aligned}
\max_{A_{n:,:},\,\mathbf{m}} \quad & M = \sum_{v\in\mathbb{T}} m_v, \\
\text{s.t.} \quad & 2\,\overline{p(E_v)} \geq c_v \cdot m_v,\ \forall v \in \mathbb{T}, \\
& \overline{p(E_v)} = 1 - (p_1^{||A_{n:(n+\rho),v}||_1} p_2^{||A^2_{n:(n+\rho),v}||_1} \cdots p_k^{||A^k_{n:(n+\rho),v}||_1}), \\
& ||A_{\tilde{v}:}||_1 \leq \tau,\ \forall \tilde{v} \in \{n+1, \cdots, n+\rho\}, \\
& A_{ij} \in \{0,1\},\ \forall i \in \{n+1, \cdots, n+\rho\},\ \forall j \in \{1, \cdots, n+\rho\}, \\
& m_v \in \{0,1\}, \forall\, v \in \{1, \cdots, n\},
\end{aligned} \tag{B.3}$$

where $m_v = 1$ (the element in vector **m**) indicates that the robustness for node v can not be verified. Specifically, it means that $2\,\overline{p(E_v)} \geq c_v$, and it disobeys our certifying condition.

There are exponential terms in $\overline{p(E_v)}$, which is difficult to solve by existing optimization tools. We further formalize the problem. By taking the logarithm of the $\overline{p(E_v)}$, we are able to transform the exponential constraint in problem (B.3) into polynomial constraint:

$$\begin{aligned}
& \tilde{P}_v \leq log(1 - \frac{c_v}{2}) \cdot m_v, \\
& \tilde{P}_v = ||A_{n:(n+\rho),v}||_1 \cdot \tilde{p_1} + ||A^2_{n:(n+\rho),v}||_1 \cdot \tilde{p_2} + \cdots + ||A^k_{n:(n+\rho),v}||_1 \cdot \tilde{p_k},
\end{aligned} \tag{B.4}$$

where $\tilde{p_k} = log(p_k)$ is a constant, and $\tilde{P}_v$ is equivalent to $log(1 - \overline{p(E_v)})$. Then the problem (B.3)

is transformed to a binary polynomial constrained programming:

$$
\begin{aligned}
\max_{A_{n:,:},\mathbf{m}} \quad & M = \sum_{v\in\mathbb{T}} m_v, \\
\text{s.t.} \quad & \tilde{P}_v \le log(1-\frac{c_v}{2})\cdot m_v, \\
& \tilde{P}_v = ||A_{n:(n+\rho),v}||_1 \cdot \tilde{p_1} + ||A^2_{n:(n+\rho),v}||_1 \cdot \tilde{p_2} + \cdots + ||A^k_{n:(n+\rho),v}||_1 \cdot \tilde{p_k}, \\
& ||A_{\tilde{v}:}||_1 \le \tau,\ \forall \tilde{v} \in \{n+1,\cdots,n+\rho\}, \\
& A_{ij} \in \{0,1\},\ \forall i \in \{n+1,\cdots,n+\rho\},\ \forall j \in \{1,\cdots,n+\rho\}, \\
& A^\top = A, \\
& m_v \in \{0,1\}, \forall\, v \in \{1,\cdots,n\}.
\end{aligned}
\tag{B.5}
$$

B.2.2 Formulating problem (B.5) as BQCLP (5.19).

In this section, we discuss the process from (B.5) to (5.19). In the case of $k=2$, the problem (B.5) becomes a binary quadratic constrained problem as follows:

$$
\begin{aligned}
\max_{A_{n:,:},\mathbf{m}} \quad & M = \sum_{v\in\mathbb{T}} m_v, \\
\text{s.t.} \quad & ||A_{n:(n+\rho),v}||_1 \cdot \tilde{p_1} + ||A^2_{n:(n+\rho),v}||_1 \cdot \tilde{p_2} \le log(1-\frac{c_v}{2})\cdot m_v, \\
& ||A_{\tilde{v}:}||_1 \le \tau,\ \forall \tilde{v} \in \{n+1,\cdots,n+\rho\}, \\
& A_{ij} \in \{0,1\},\ \forall i \in \{n+1,\cdots,n+\rho\},\ \forall j \in \{1,\cdots,n+\rho\}, \\
& A^\top = A, \\
& m_v \in \{0,1\}, \forall\, v \in \{1,\cdots,n\}.
\end{aligned}
\tag{B.6}
$$

Next, we divide the adjacency matrix A into four parts as shown in Fig.B.1, and then the A^2

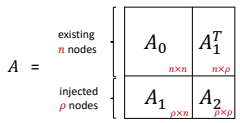


Figure B.1: Illustration of adjacency matrix notation.

can be interpreted as:

$$A^2 = \begin{bmatrix} (A_0A_0 + A_1^\top A_1)_{(n\times n)} & (A_0A_1^\top + A_1^\top A_2)_{(\rho\times n)} \\ (A_1A_0 + A_2A_1)_{(\rho\times n)} & (A_1A_1^\top + A_2A_2)_{(\rho\times\rho)} \end{bmatrix}.$$

Then, the l_1 norm of $A^2_{n:(n+\rho),v}$ can be represented as:

$$[||A^2_{n:(n+\rho),1}||_1, ||A^2_{n:(n+\rho),2}||_1, \cdots, ||A^2_{n:(n+\rho),n}||_1]^\top = (A_1A_0 + A_2A_1)\mathbf{1}_\rho. \tag{B.7}$$

Also, same as above, together with Fig. B.1, $||A_{\tilde{v}:}||_1$ is described as:

$$[||A_{n:}||_1, ||A_{(n+2):}||_1, \cdots, ||A_{(n+\rho):}||_1]^\top = A_1\mathbf{1}_n + A_2\mathbf{1}_\rho. \tag{B.8}$$

Finally, combine (B.7) and (B.8), problem (B.6) can be formulated as:

$$\begin{aligned} \max_{A_1,A_2,\mathbf{m}} \quad & M = \mathbf{t}^\top\mathbf{m}, \\ \text{s.t.} \quad & \tilde{p_1}A_1^\top\mathbf{1}_\rho + \tilde{p_2}(A_1A_0 + A_2A_1)^\top\mathbf{1}_\rho \le \mathbf{C}\circ\mathbf{m}, \\ & A_1\mathbf{1}_n + A_2\mathbf{1}_\rho \le \tau,\ A_2^\top = A_2, \\ & A_1 \in \{0,1\}^{\rho\times n},\ A_2 \in \{0,1\}^{\rho\times\rho},\ \mathbf{m} \in \{0,1\}^n, \end{aligned}$$

where $\mathbf{t}$ is a constant zero-one vector that encodes the position of the target node set $\mathbb{T}$, $\mathbf{m}$ is a vector that indicates whether the nodes are successfully attacked, $\mathbf{C} \in \mathbb{R}^n$ is a vector with negative constant elements $log(1 - \frac{c_v}{2})$, for $v = 1, 2, \cdots, n$.

B.2.3 Formulating problem (5.19) as Linear Programming Problem (5.20).

Here, we discuss the details of the process of relaxing the BQCLP problem (5.19) to the LP problem (5.20). In problem (5.19), there are $\rho^2 n$ quadratic terms among A_2A_1. To tackle the challenge, we introduce the following transformation to transform it into an LP problem. Specifically, we first substitute the quadratic terms with linear terms and relax all the binary variables to continuous variables in $[0, 1]$.

If $x \in \mathbb{B}$, $y \in \mathbb{B}$ are two integer binary variables, then the quadratic term xy can be substitute by a single variable $z := xy$ with the combination of linear constraints [179]: $z \le x$, $z \le$

$y,\ x+y-z\leq 1,\ z\in\mathbb{B}$. We use $a_{(ij)}$ and $b_{(ij)}$ to denotes the element in i^{th} row and j^{th} column of matrix A_1 and A_2 respectively. For each quadratic term $b_{(ij)}a_{(jv)}$ ($\forall i\in\{1,\cdots,\rho\},\forall j\in\{1,\cdots,\rho\},\forall v\in\{1,\cdots,n\}$) in A_2A_1, we create a substitution variable $Q_{v(ij)}:=b_{(ij)}a_{(jv)}$ with corresponding constraints: $Q_{v(ij)}\in\mathbb{B}, Q_{v(ij)}\leq b_{(ij)}, Q_{v(ij)}\leq a_{(jv)}$, and $b_{(ij)}+a_{(jv)}-Q_{v(ij)}\leq 1$. The existing linear terms remain unchanged. Now, the BQCLP problem has transformed into binary linear programming (BLP).

Next, we formulate the problem using matrix representation. We firstly use O to substitute $(A_2A_1)^\top\mathbf{1}_\rho$, and we have the first constraint as:

$$\tilde{p_1}A_1^\top\mathbf{1}_\rho+\tilde{p_2}A_0^\top A_1^\top\mathbf{1}_\rho+\tilde{p_2}O\leq\mathbf{C}\circ\mathbf{m}.$$

We list the elements of the A_1 and A_2 as follows:

$$A_1=\begin{bmatrix}a_{11} & a_{12} & a_{13} & \cdots & a_{1n}\\ a_{21} & & & & \\ a_{31} & & \ddots & & \vdots \\ \vdots & & & & \\ a_{\rho 1} & & \cdots & & a_{\rho n}\end{bmatrix},\ A_2=\begin{bmatrix}b_{11} & b_{12} & b_{13} & \cdots & b_{1\rho}\\ b_{21} & & & & \\ b_{31} & & \ddots & & \vdots \\ \vdots & & & & \\ b_{\rho 1} & & \cdots & & b_{\rho\rho}\end{bmatrix}. \tag{B.9}$$

Then, the matrix multiplication of A_2 and A_1 is

$$A_2A_1=\begin{bmatrix}b_{11}a_{11}+b_{12}a_{21}+\cdots+b_{1\rho}a_{\rho 1} & b_{11}a_{12}+b_{12}a_{22}+\cdots+b_{1\rho}a_{\rho 2} & \cdots & b_{11}a_{1n}+b_{12}a_{2n}+\cdots+b_{1\rho}a_{\rho n}\\ b_{21}a_{11}+b_{22}a_{21}+\cdots+b_{2\rho}a_{\rho 1} & b_{21}a_{12}+b_{22}a_{22}+\cdots+b_{2\rho}a_{\rho 2} & \cdots & b_{21}a_{1n}+b_{22}a_{2n}+\cdots+b_{2\rho}a_{\rho n}\\ \vdots & \vdots & \ddots & \vdots \\ b_{\rho 1}a_{11}+b_{\rho 2}a_{21}+\cdots+b_{\rho\rho}a_{\rho 1} & b_{\rho 1}a_{12}+b_{\rho 2}a_{22}+\cdots+b_{\rho\rho}a_{\rho 2} & \cdots & b_{\rho 1}a_{1n}+b_{\rho 2}a_{2n}+\cdots+b_{\rho\rho}a_{\rho n}\end{bmatrix}.$$

By the definition of matrix Q_v, for $v\in\{1,2,\cdots,n\}$, we have the following equivalent representation:

$$Q_v=\begin{bmatrix}Q_{v(11)} & Q_{v(12)} & \cdots & Q_{v(1\rho)}\\ Q_{v(21)} & Q_{v(22)} & & Q_{v(2\rho)}\\ \vdots & \vdots & \ddots & \vdots \\ Q_{v(\rho 1)} & Q_{v(\rho 2)} & \cdots & Q_{v(\rho\rho)}\end{bmatrix}:=\begin{bmatrix}b_{11}a_{1v} & b_{21}a_{1v} & \cdots & b_{\rho 1}a_{1v}\\ b_{12}a_{2v} & b_{22}a_{2v} & & b_{\rho 2}a_{2v}\\ \vdots & \vdots & \ddots & \vdots \\ b_{1\rho}a_{\rho v} & b_{2\rho}a_{\rho v} & \cdots & b_{\rho\rho}a_{\rho v}\end{bmatrix}.$$

We notice that $(A_2A_1)^\top \mathbf{1}_\rho$ is to sum the A_2A_1 by its column, and each Q_v contains all the terms for each vector summation. Then we have $O = (A_2A_1)^\top = [\mathbf{1}_\rho^\top Q_1 \mathbf{1}_\rho, \mathbf{1}_\rho^\top Q_2 \mathbf{1}_\rho, \cdots, \mathbf{1}_\rho^\top Q_n \mathbf{1}_\rho]^\top$.

Further, by decomposing the meaning of Q_v, we have

$$Q_v := \begin{bmatrix} b_{11} & b_{21} & \cdots & b_{\rho 1} \\ b_{12} & b_{22} & \cdots & b_{\rho 2} \\ \vdots & \vdots & \ddots & \vdots \\ b_{1\rho} & b_{2\rho} & \cdots & b_{\rho\rho} \end{bmatrix} \circ \begin{bmatrix} a_{1v} & a_{1v} & \cdots & a_{1v} \\ a_{2v} & a_{2v} & \cdots & a_{2v} \\ \vdots & \vdots & \ddots & \vdots \\ a_{\rho v} & a_{\rho v} & \cdots & a_{\rho v} \end{bmatrix} = A_2 \circ \mathbf{1}_\rho \begin{bmatrix} a_{1v} \\ a_{2v} \\ \vdots \\ a_{\rho v} \end{bmatrix}^\top = A_2 \circ \mathbf{1}_\rho [A_{1(:,v)}]^\top.$$

To make the Q_v equivalent to the quadratic terms, for every Q_v, we need to add its constraints:

$$Q_v \le A_2,\ Q_v \le \mathbf{1}_\rho [A_{1(:,v)}]^\top,\ \mathbf{1}_\rho [A_{1(:,v)}]^\top + A_2 - Q_v \le 1.$$

Finally, we relaxed A_1, A_2, Q_v to relax all the binary variables to continuous variables in $[0, 1]$:

$$Q_v \in [0,1]^{\rho\times\rho},\ A_1 \in [0,1]^{\rho\times n},\ A_2 \in [0,1]^{\rho\times\rho},\ \mathbf{m} \in [0,1]^n.$$

Then we have the linear programming problem (5.20) as follows:

$$\begin{aligned} \max_{\substack{A_1, A_2, m, \\ Q_1, Q_2, \cdots, Q_n}} \quad & M = \mathbf{t}^\top \mathbf{m}, \\ \text{s.t.} \quad & \tilde{p_1} A_1^\top \mathbf{1}_\rho + \tilde{p_2} A_0^\top A_1^\top \mathbf{1}_\rho + \tilde{p_2} O \le \mathbf{C} \circ \mathbf{m} \\ & A_1 \mathbf{1}_n + A_2 \mathbf{1}_\rho \le \tau, \\ & Q_v = (Q_{v(ij)})_{\rho\times\rho},\ v \in \{1, 2, \cdots, n\}, \\ & O = [\mathbf{1}_\rho^\top Q_1 \mathbf{1}_\rho, \mathbf{1}_\rho^\top Q_2 \mathbf{1}_\rho, \cdots, \mathbf{1}_\rho^\top Q_n \mathbf{1}_\rho]^\top, \\ & Q_v \le \mathbf{1}_\rho [A_{1(:,v)}]^\top, \\ & Q_v \le A_2, \\ & \mathbf{1}_\rho [A_{1(:,v)}]^\top + A_2 - Q_v \le 1, \\ & Q_v \in [0,1]^{\rho\times\rho}, \\ & A_1 \in [0,1]^{\rho\times n}, \\ & A_2 \in [0,1]^{\rho\times\rho}, \\ & A_2^\top = A_2, \\ & \mathbf{m} \in [0,1]^n. \end{aligned}$$

B.2.4 Formulating problem (5.19) as Linear Programming Problem (5.22).

We start from (5.19), and we have the first constraint:

$$\tilde{p_1} A_1^\top \mathbf{1}_\rho + \tilde{p_2} A_0^\top A_1^\top \mathbf{1}_\rho + \tilde{p_2} A_1^\top A_2^\top \mathbf{1}_\rho \leq \mathbf{C} \circ \mathbf{m}.$$

Then, we substitute $A_2^\top \mathbf{1}_\rho$ with $\mathbf{z}$,

$$\mathbf{z} := A_2^\top \mathbf{1}_\rho = \begin{bmatrix} b_{11} & b_{12} & b_{13} & \cdots & b_{1\rho} \\ b_{21} & & & & \\ b_{31} & & \ddots & & \vdots \\ \vdots & & & & \\ b_{\rho 1} & & \cdots & & b_{\rho\rho} \end{bmatrix}_{(\rho,\rho)} \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{(\rho,1)} = \begin{bmatrix} b_{11} + b_{12} + b_{13} + \cdots + b_{1\rho} \\ b_{21} + b_{22} + b_{23} + \cdots + b_{2\rho} \\ \vdots \\ b_{\rho 1} + b_{\rho 2} + b_{\rho 3} + \cdots + b_{\rho\rho} \end{bmatrix}_{(\rho,1)}. \tag{B.10}$$

Then, from (B.10), the constraint is transformed into

$$\begin{aligned} \tilde{p_1} A_1^\top \mathbf{1}_\rho + \tilde{p_2} A_0^\top A_1^\top \mathbf{1}_\rho + \tilde{p_2} A_1^\top \mathbf{z} \leq \mathbf{C} \circ \mathbf{m}, \\ z_i \in \{0, 1, 2, \cdots, min(\tau, \rho)\} \quad \forall i \in \{0, 1, 2, \cdots, \rho\}. \end{aligned} \tag{B.11}$$

In (5.19), since there exists the constraint: $A_1 \mathbf{1}_n + A_2 \mathbf{1}_\rho \leq \tau$, so we have z_i satisfies $z_i \in \{0, 1, 2, \cdots, min(\tau, \rho)\}$. Next, we deal with the quadratic term $A_1^\top \mathbf{z}$.

If $x \in \mathbb{B}$ is a binary variable, and $z \in [0, u]$ is a continuous variable, then the quadratic term xy can be substitute by a single variable $z := xy$ with the combination of linear constraints [179]: $w \leq ux, w \leq z, ux + z - w \leq u, 0 \leq w$. To apply it, we first relax the $\mathbf{z}$ to $[0, min(\tau, \rho)]$.

We know that $A_1^\top \mathbf{z}$ satisfies that

$$A_1^\top \mathbf{z} = \begin{bmatrix} a_{11} & a_{21} & a_{31} & \cdots & a_{\rho 1} \\ a_{12} & a_{22} & a_{32} & \cdots & a_{\rho 2} \\ a_{13} & a_{23} & a_{33} & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & a_{3n} & \cdots & a_{\rho n} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_\rho \end{bmatrix} = \begin{bmatrix} a_{11} z_1 + a_{21} z_2 + \cdots + a_{\rho 1} z_\rho \\ a_{12} z_1 + a_{22} z_2 + \cdots + a_{\rho 2} z_\rho \\ \vdots \\ a_{1n} z_1 + a_{2n} z_2 + \cdots + a_{\rho n} z_\rho \end{bmatrix}_{(n,1)}.$$

Then, we create a new variable matrix Q to substitute $A_1^\top \mathbf{z}$, with each of its element: $q_{ij} := a_{ji}z_i, (\forall i \in \{1, 2, \cdots, n\}, j \in \{1, 2, \cdots, \rho\})$. That is:

$$Q = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1\rho} \\ q_{21} & q_{22} & \cdots & q_{2\rho} \\ \vdots & \vdots & \ddots & \vdots \\ q_{n1} & q_{n2} & \cdots & q_{n\rho} \end{bmatrix} = \begin{bmatrix} a_{11}z_1 & a_{21}z_2 & \cdots & a_{\rho 1}z_\rho \\ a_{12}z_1 & a_{22}z_2 & \cdots & a_{\rho 2}z_\rho \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n}z_1 & a_{2n}z_2 & \cdots & a_{\rho n}z_\rho \end{bmatrix}.$$

We now have $A_1^\top \mathbf{z} = Q\mathbf{1}_\rho$. Assuming that $\tau \leq \rho$, for each quadratic term $A_{1(ij)}^\top z_j$ $(\forall i \in \{1, \cdots, n\}, \forall j \in \{1, \cdots, \rho\})$ in $A_1^\top \mathbf{z}$, we create a substitution variable $Q_{(ij)} = A_{1(ij)}^\top z_j$ with corresponding constraints: $0 \leq Q_{(ij)}, Q_{(ij)} \leq \tau A_{1(ij)}^\top$, $Q_{(ij)} \leq z_j$, and $\tau A_{1(ij)}^\top + z_j - Q_{(ij)} \leq \tau$. Further, with matrix notation, we have

$$\begin{aligned} 0 \leq Q \leq \tau A_1^\top, \\ 0 \leq 1_n \mathbf{z}^\top - Q \leq \tau(1 - A_1^\top), \\ A_1 \in \{0, 1\}, \mathbf{z} \in [0, \tau], Q \in [0, \tau]. \end{aligned} \tag{B.12}$$

Finally, we relax all the binary variables to be continuous variables, We have problem (5.22) as follows:

$$\begin{aligned} \max_{\substack{A_1, m, z \\ Q \in \mathbb{R}^{n \times \rho}}} \quad & M = \mathbf{t}^\top \mathbf{m}, \\ \text{s.t.} \quad & \tilde{p_1} A_1^\top \mathbf{1}_\rho + \tilde{p_2} A_0^\top A_1^\top \mathbf{1}_\rho + \tilde{p_2} Q \mathbf{1}_\rho \leq \mathbf{C} \circ \mathbf{m}, \\ & A_1 \mathbf{1}_n + \mathbf{z} \leq \tau, \\ & Q \leq \tau A_1^\top, \\ & Q \leq \mathbf{1}_n \mathbf{z}^\top, \\ & \tau A_1^\top + \mathbf{1}_n \mathbf{z}^\top - Q \leq \tau, \\ & Q \in [0, \tau]^{n \times \rho}, \\ & A_1 \in [0, 1]^{\rho \times n}, \\ & \mathbf{z} \in [0, \tau]^{\rho \times 1}, \\ & \mathbf{m} \in [0, 1]^n. \end{aligned} \tag{B.13}$$

B.3 Algorithm of our proposed methods

Train a base classifier f. Following the work of [131], our first step is to train a graph model to serve as the base classifier. To enhance the model's generalization ability on the smoothing samples, we incorporate random noise augmentation during the training process. The training procedure is summarized in Algorithm 8, providing an overview of the steps involved. Given a clean graph G, a smoothing distribution $\phi(G)$ with smoothing parameters p_e and p_n, and the number of training epochs E, the algorithm iteratively trains the model on randomly generated graphs. In each epoch, a random graph G_e is drawn from the smoothing distribution $\phi(G)$. The model is then trained on the training nodes using this randomly generated graph. This process is repeated for the specified number of training epochs.

Algorithm 8 Graph model training [131].

Input: Clean graph G, smoothing distribution $\phi(G)$ with smoothing parameters p_e and p_n, training epoch E.

1: **for** $e = 1, \cdots, E$ **do**
2: Draw a random graph $G_e \sim \phi(G)$.
3: $f = train\_model(f(G_e))$ on training nodes.
4: **end for**
5: **return** A base classifier $f(\cdot)$.

Obtaining prediction probability of smoothed classifier g. Next, we need to obtain the prediction results of a smoothed classifier. As depicted in Algorithm 9, we sample N graphs $G_1, G_2, \ldots, G_N$ from the smoothed distribution $\phi(G) = (\phi_e(G), \phi_n(G))$ based on the base classifier f. To estimate the probabilistic prediction, we employ a Monte Carlo process. For each sampled graph G_i, we calculate the prediction probability $p_{v,y}(G)$, which represents the frequency of the predicted class y for the vertex v. This can be approximated as $p_{v,y}(G) \approx \sum_{i=1}^{N} \mathbb{I}(f_v(G_i) = y)/N$, where $\mathbb{I}$ is the indicator function.

Let denote the top class probability $p_A := p_{v,y^*}(G)$ and runner-up class probability $p_B := max_{y \neq y^*} p_{v,y}(G)$, we want to bound the impact of randomness. Specifically, we compute the

lower bound of p_A (denoted as $\underline{p_A}$) and upper bound of p_B (denoted as $\overline{p_B}$). Applying the Clopper-Pearson Bernoulli confidence interval, we obtain the $\underline{p_A}$ and the $\overline{p_B}$ under a confidence level of α/C, where C represents the number of classes in the model.

Algorithm 9 Monte Carlo sampling [131].

Input: Clean graph G, smoothing distribution $\phi(G)$ with smoothing parameters p_e and p_n, trained base classifier $f(\cdot)$, sample number N, confidence level α.

1: Draw N random graphs $\{G_i| \sim G_i \sim \phi(G)\}_{i=1}^{N}$.
2: $counts = |\{i : f(G_i) = y\}|$, for $y = 1, \cdots, C$.
3: $y_A, y_B =$ top two indices in $counts$.
4: $n_A, n_B = counts[y_A], counts[y_B]$.
5: $\underline{p_A}, \overline{p_B} =$ CP_Bernolli(n_A, n_B, N, α).
6: **return** $\underline{p_A}, \overline{p_B}$.

Collective certification via solving an optimization problem. We obtain the collective certified robustness by solving the optimization problem problem (5.20) or (5.22). The process is described in Algorithm 10.

In this algorithm, we first set up the constant $\tilde{p}_1$ and $\tilde{p}_2$ based on the given smoothing parameters p_e and p_n. Next, for each node v in the target node set $\mathbb{T}$, we obtain the lower bound $\underline{p_A}$ and the upper bound $\overline{p_B}$ using Algorithm 9. These bounds are based on the prediction probabilities of the smoothed classifier for the current node v. We then compute the value $c_v = \underline{p_A} - \overline{p_B}$ and prepare the constant vector $\mathbf{C}$ with elements $\log(1 - \frac{c_v}{2})$ for each node v. The objective function of the optimization problem is based on either (5.20) or (5.22), depending on the chosen formulation. The constraints are also set up accordingly. Finally, we solve the linear programming using an LP solver, such as MOSEK, to obtain the optimal value M^*. The certified ratio, which represents the percentage of nodes in the target set $\mathbb{T}$ that have been successfully certified, is then computed as $(|\mathbb{T}| - M^*)/|\mathbb{T}|$.

Algorithm 10 Certified robustness via solving optimization problem (5.20) or (5.22).

Input: Smoothing parameters p_e and p_n, graph adjacent matrix A_0, perturbation budget ρ and τ, target node set $\mathbb{T}$.

Set constant $\tilde{p}_1 = log(1 - (\bar{p}_e\bar{p}_n))$.

Set constant $\tilde{p}_2 = log(1 - (\bar{p}_e\bar{p}_n)^2)$.

for v in $\mathbb{T}$ **do**

    Obtain $\underline{p_A}$, $\overline{p_B}$ from Algorithm. 9 for current node v.

    Compute $c_v = \underline{p_A} - \overline{p_B}$.

    Prepare constant vector **C** with each element: $log(1 - \frac{c_v}{2})$.

end for

Setup objective function in (5.20) or (5.22).

Setup constraints in (5.20) or (5.22).

Solve the optimization problem using LP solver such as MOSEK to get M^*.

Return Certified ratio $(|\mathbb{T}| - M^*)/|\mathbb{T}|$.

B.4 Other Experimental Results

B.4.1 Trade off between Clean accuracy and the certified ratio on GCN model

In this section, we present the remaining experiments as outlined in Section. 5.2.4.1. A superior certifying method should not only achieve a higher certified ratio but also maintain or improve the clear accuracy, which represents the original model's performance. We compare the results of these two metrics for our method under different parameter settings as shown in Figure. B.2. In the figures, the data points situated closer to the upper right side represent higher certified ratios and clean accuracy. It is evident that both of our proposed methods consistently outperform the sample-wise method, demonstrating their superior performance under various attacker power ρ.

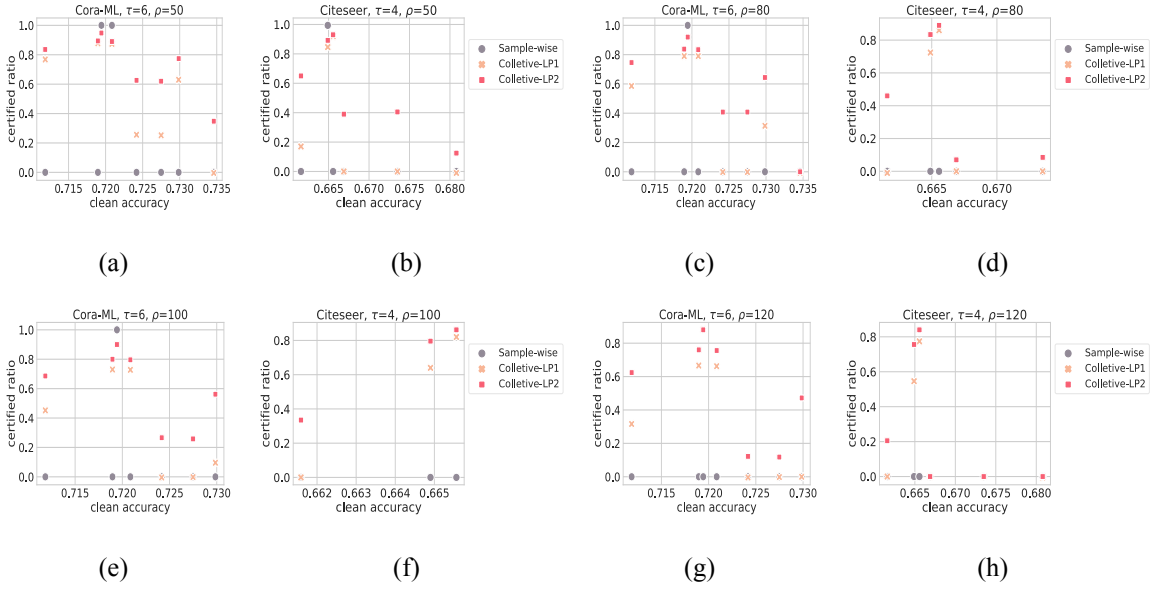


Figure B.2: Clean accuracy and the certified ratio of our collective model under various smoothing parameters on GCN model.

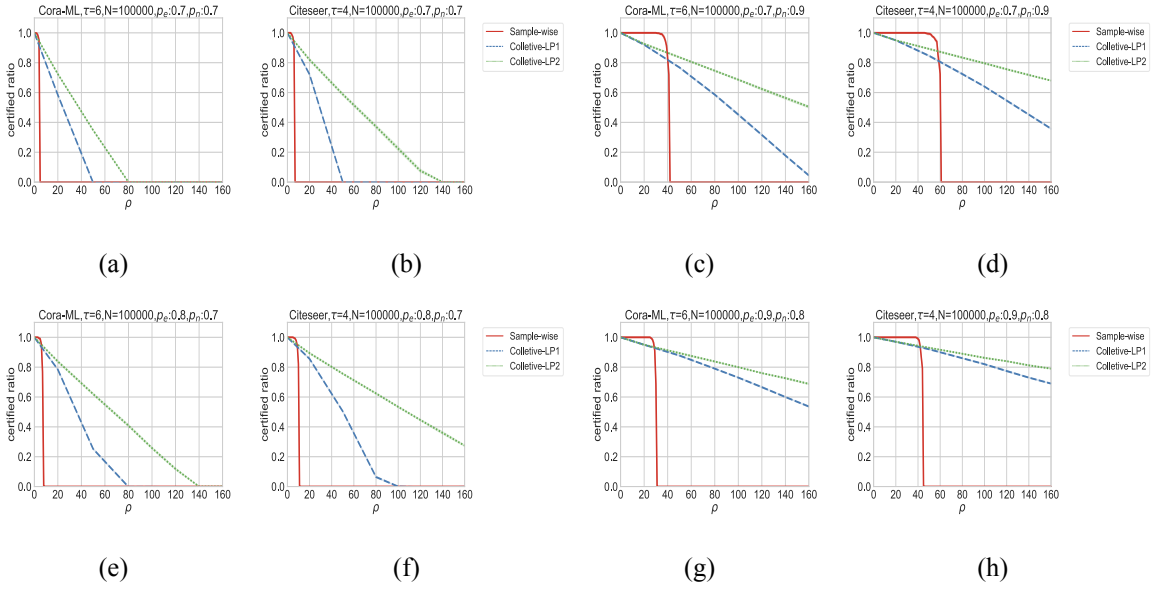


Figure B.3: Certified ratio of our collective model under various smoothing parameters on GCN model.

B.4.2 GCN certified ratio of our methods under different smoothing parameters

In addition, we conducted experiments to compare the performance of our methods with the sample-wise method under different combinations of parameters p_e and p_n on the Cora and Citeseer datasets. The results are shown in Figure. B.3.

From the figures, we can observe that our proposed methods always exhibit a larger certifiable radius. For example, when ρ exceeds 60, the sample-wise method fails to defend against any attacks, while our methods are still able to provide certifiable guarantees.

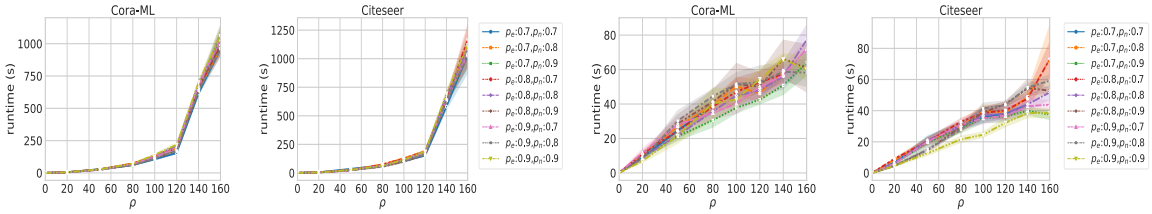
B.4.3 Time complexity comparison of two relaxations

Furthermore, we provide more detailed results on the runtime of the two proposed methods with different parameters in Figure. B.4. From the figures, we can observe that as the attack budget ρ increases, the proposed Collective-LP2 method demonstrates superior efficiency compared to Collective-LP1 in both datasets. This efficiency advantage is particularly evident when ρ exceeds 120. Notably, when $\rho = 160$, the Collective-LP1 takes approximately $1,000$ seconds to complete the computation. On the other hand, the time consumption of Collective-LP2 remains consistently below 90 seconds.

These results highlight the computational advantage of Collective-LP2 over Collective-LP1, especially for larger attack budgets. The reduced runtime of Collective-LP2 ensures the practicality and efficiency of our proposed method, making it suitable for real-world scenarios with larger attack budgets.

B.4.4 Against Global Attack: Verifying all testing nodes in a time

Alternatively, instead of verifying a subset of target nodes $\mathbb{T}$, we can extend our approach to verify all the testing nodes in the graph, as illustrated in Figure B.5. In this scenario, we measure the certified accuracy, which represents the ratio of nodes that are both correctly classified and



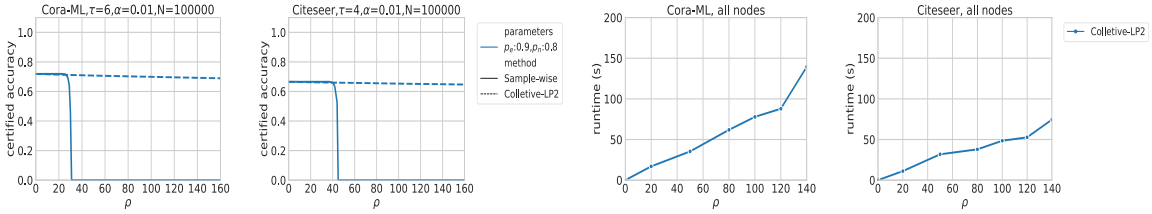
(a) Collective-LP1 (b) Collective-LP1 (c) Collective-LP2 (d) Collective-LP2

Figure B.4: Runtime of our collective model under various smoothing parameters.

certified to be consistent, as well as the runtime of our customized approach (Collective-LP2).

We have observed that the certified accuracy of our collective certificate only experiences a slight decrease as the attack budget increases, while the sample-wise approach can only certify the case of ρ less than 50. This indicates that our approach maintains a high level of certified robustness even when facing more severe adversarial attacks.

Furthermore, it is worth noting that our Collective-LP2 formulation exhibits excellent computational efficiency. Despite the presence of more than 1500 testing nodes, the problem can be solved in less than 3 minutes, even when the number of injected nodes ρ is set to 140 (approximately $5\% \times n$). This demonstrates the scalability and practicality of our customized relaxation approach (Collective-LP2) in real-world scenarios.



(a) Certified accuracy (b) Certified accuracy (c) Runtime (d) Runtime (Collective-LP2)

Figure B.5: Certified accuracy and runtime in the case of setting all the testing nodes as $\mathbb{T}$.