Cross-Paradigm Graph Backdoor Attacks with Promptable Subgraph Triggers

Dongyi Liu[1,2] , **Jiangtong Li**[*#,2]
[1]The Hong Kong University of Science and Technology (Guangzhou)
[2]Tongji University
dliu587@connect.hkust-gz.edu.cn, jiangtongli@tongji.edu.cn,

Abstract

Graph Neural Networks (GNNs) are vulnerable to backdoor attacks, where adversaries implant malicious triggers to manipulate model predictions. Existing trigger generators are often simplistic in structure and overly reliant on specific features, confining them to a single graph learning paradigm, such as graph supervised learning, graph contrastive learning, or graph prompt learning. Such paradigm-specific designs lead to poor transferability across different learning frameworks, limiting attack success rates in general testing scenarios. To bridge this gap, we propose **C**ross-**P**aradigm **G**raph **B**ackdoor **A**ttacks with Promptable Subgraph Triggers (**CP-GBA**), which employs Graph Prompt Learning (GPL) to synthesize transferable subgraph triggers. Specifically, we first distill a compact yet expressive trigger set into a queryable repository, jointly optimizing for class-awareness, feature richness, and structural fidelity. Furthermore, we pioneer the theoretical exploration of GPL transferability under prompt-based objectives, ensuring robust generalization to diverse and unseen test-time paradigms. Extensive experiments across multiple real-world datasets and defense scenarios show that CP-GBA achieves state-of-the-art attack success rates. Code is available at https://github.com/novdream/CP-GBA.

1 Introduction

GNNs have been widely applied to analyze various types of graph-structured data in real-world applications, including social networks, molecular graphs, and financial systems [Weber *et al.*, 2019; Fan *et al.*, 2019; Cheng *et al.*, 2020; Bongini *et al.*, 2021]. Their success is largely attributed to the message-passing mechanism [Xu *et al.*, 2019], where nodes iteratively aggregate information from their neighbors. This process results in node representations that preserve both local structural properties and node attributes. Therefore, through Graph Supervised Learning (GSL), GNNs can be applied to various supervised tasks, including node classification, link prediction, and graph classification [Hamilton *et al.*, 2017; Velickovic *et al.*, 2017; Zhang and Chen, 2018; Zhang *et al.*, 2018; Jiang *et al.*, 2019]. However, GSL heavily relies on large amounts of labeled data for downstream tasks, which is often expensive and labor-intensive. To mitigate the label dependence of traditional supervised learning, Graph Contrastive Learning (GCL) [Zhu *et al.*, 2020; Wu *et al.*, 2024] extracts supervision from distinct augmented views, whereas Graph Prompt Learning (GPL) [Sun *et al.*, 2022; Liu *et al.*, 2023; Sun *et al.*, 2023a; Jiang *et al.*, 2024] bridges pre-trained models to specific tasks via prompts. These paradigms enable efficient training with minimal data and improve the cross-domain learning capabilities of GNNs, making them ideal for scenarios with scarce labeled data.

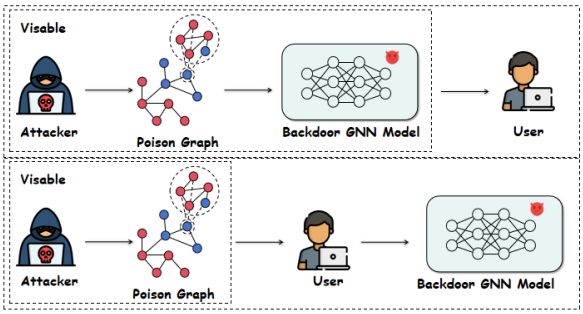

Figure 1: **Top**: Model poisoning, where the attacker distributes a pre-trained backdoored model. **Bottom**: Data poisoning, where the victim trains a model using attacker-provided poisoned data.

Although GNNs have achieved remarkable performance across various applications and diverse graph structures, prior research [Zhang *et al.*, 2023; Dai *et al.*, 2023; Zhang *et al.*, 2024a; Lyu *et al.*, 2024; Lin *et al.*, 2024; Zhang *et al.*, 2024b; Li *et al.*, 2024] has revealed that GNNs trained under different learning paradigms are vulnerable to backdoor attacks. For GSL, UGBA [Dai *et al.*, 2023] proposes a similarity loss to reduce the inconsistent similarity between triggers and attacked nodes, thereby improving trigger stealthiness. DPGBA [Zhang *et al.*, 2024a] further improves attack effectiveness by employing an adversarial network to address the out-of-distribution (OOD) issue of trigger representations.

[*#]Corresponding Authors

Paradigm	Model	Clean	GTA		UGBA		DPGBA	
GSL	GCN	85.1	88.8	85.1	93.1	85.1	92.3	85.0
GSL	GAT	84.9	88.1	84.9	92.5	85.2	92.6	85.1
GCL	GRACE	72.1	31.5	62.2	60.9	65.2	2.3	67.4
GPL	GraphPrompt	43.3	21.3	37.7	38.2	42.6	19.6	42.5

Table 1: Results of existing graph backdoor attacks trained by GSL-based GCN model on Pubmed in diferent attack scenarios(Attack Success Rate (%) | Clean Accuracy (%))

For GCL, GCBA [Zhang *et al.*, 2023] is designed to implant backdoors during the contrastive learning phase, showing that attacks can succeed even with limited label information. For GPL, CrossBA [Lyu *et al.*, 2024] employs a hinge loss approach to maximize the similarity between the trigger and target node representations, while minimizing their similarity with clean node representations. This attack strategy optimizes both the triggers and GNNs, making the backdoor effective for downstream cross-task applications. TGPA [Lin *et al.*, 2024] further improves the attack success rate by optimizing triggers and task headers within a bi-level optimization framework under the assumption of a frozen GNN encoder.

Despite notable progress in cross-domain and cross-task backdoor capabilities, most prior works still focus on the model poisoning scenario in Fig. 1 and attack a single learning paradigm, lacking transferability to different attack scenarios. For example, in social network scenarios, attackers can only poison the social network data by creating virtual users (the triggers). Subsequently, legitimate users crawl this public social network data to locally train backdoor models, which are designed to learn user representation embeddings for downstream tasks like recommendation systems. However, a critical gap arises when the victim's training paradigm differs from the attacker's surrogate Diverse learning paradigms induce distinct feature representations and distributions [Xu, 2020], causing severe inconsistencies in the trigger feature space that diminish attack effectiveness. Moreover, the trigger generators trained for graph backdoor attacks often have simple architectures and rely on node-specific features, limiting their ability to generate triggers with structure-aware diversity and feature richness. As a result, they struggle to maintain transferability and effectiveness across diverse learning paradigms. In Tab. 1, we validate the failure of backdoor attacks across learning paradigms on Pubmed dataset.

Therefore, in this paper, we study the novel and critical problem of designing transferable graph backdoor attacks across multiple learning paradigms in **node classification tasks**. The challenges in developing such an attack mainly involve two key aspects: (**i**) How to ensure trigger generalization across models trained with different learning paradigms? (**ii**) How to design triggers that effectively capture the intrinsic structure and prior knowledge of the data? Inspired by recent researches on GPL [Sun *et al.*, 2022; Sun *et al.*, 2023a; Wang *et al.*, 2024], we propose Cross-Paradigm Graph Backdoor Attacks with Promptable Subgraph Triggers (**CP-GBA**), a novel approach designed to improve the transferability of graph backdoor attacks across different graph learning paradigms. Our approach first constructs a set of condensed subgraph triggers to increase trigger diversity and maintain in-distribution structural properties, as discussed in Sec. 4.1. To ensure transferability across learning paradigms, we further employ GPL to train triggers, utilizing its theoretical transferability detailed in Sec. 4.2. Through these steps, CP-GBA enables effective manipulation of node classifications in graph backdoor attacks. Extensive experiments confirm that our method consistently achieves superior attack efficacy across diverse datasets and learning paradigms, even in the presence of advanced defense mechanisms. In summary, our contributions are:

- **Problem**: We address a novel backdoor attack problem: generalizing attacks across graph learning paradigms.
- **Method**: To the best of our knowledge, we are the first to exploit GPL for training backdoor triggers through both theoretical analysis and extensive experiments.
- **Results**: Extensive experiments on different real-world datasets with various defense strategies show that CP-GBA outperforms SOTA graph backdoor attack methods.

2 Related Work

2.1 GNNs on Node Classification

GNN Architectures. GNNs have emerged as a powerful tool for node classification by using information propagation among nodes. Early foundational works like GCN [Kipf and Welling, 2016] utilize localized spectral convolutions to aggregate neighborhood information. To capture more complex dependencies, attention-based models such as GAT [Velickovic *et al.*, 2017] and GraphTransformer [Yun *et al.*, 2019] are introduced, allowing nodes to assign learnable importance weights to neighbors. Addressing scalability in large graphs, GraphSAGE [Hamilton *et al.*, 2017] enables efficient inductive learning by neighbor sampling and aggregation.

Graph Learning Paradigms. Beyond architectures, diverse training paradigms have evolved to address data constraints and generalization. For **GSL** [Kipf and Welling, 2016], the model is directly optimized through supervised loss functions using high-quality labeled data, achieving success in tasks such as node classification, link prediction, and graph classification. For **GCL** [You *et al.*, 2020], to overcome the challenge of scarce labeled data, models aim to learn robust representations by maximizing the agreement between different augmented views of the same graph without supervision. GraphCL [You *et al.*, 2020] uses random graph augmentations to generate multiple views and learns node embeddings by contrasting positive and negative pairs via the InfoNCE loss. CCA-SSG [Zhang *et al.*, 2021a] further refines the loss function using canonical correlation analysis. For **GPL** [Sun *et al.*, 2022], to enhance cross-domain generalization under data scarcity, models adopt a "pre-training, prompt-tuning" strategy [Yu *et al.*, 2024] to adapt pre-trained GNNs to downstream tasks. GPF [Jiang *et al.*, 2024] and GraphPrompt [Liu *et al.*, 2023] utilize *prompt-as-tokens* by appending learnable vectors to input features or latent spaces, respectively. All-in-one [Sun *et al.*, 2023a] employs *prompt-as-graphs* by injecting a learnable subgraph structure where tokens connect to original nodes to unify diverse tasks.

2.2 Backdoor Attacks on GNN

Backdoor attacks against GNNs [Yang *et al.*, 2025; Ding *et al.*, 2025] typically involve injecting malicious triggers into the training graph and associating them with a predetermined target label. As a result, when GNNs trained on the backdoored graph encounter test samples containing these triggers, they produce attacker-desired predictions. These attacks can be categorized based on learning paradigms. In GSL-based backdoor attacks, Zhang *et al.* [Zhang *et al.*, 2021b] propose to inject universal triggers into training samples via a subgraph-based approach with limited attack success rate. Building on this, Xi *et al.* [Xi *et al.*, 2021] introduce a technique for generating adaptive triggers, customizing perturbations for individual samples to improve attack effectiveness. Dai *et al.* [Dai *et al.*, 2023] propose a poisoned node algorithm to maximize the attack budget and includes an adaptive trigger generator to produce triggers with high cosine similarity to the target node. Zhang *et al.* [Zhang *et al.*, 2024a] further consider the OOD problem in triggers and employs a GAN loss to generate in-distribution backdoor triggers. For GCL-based backdoor attacks, Zhang *et al.* [Zhang *et al.*, 2023] are the first to systematically investigate attacks across different contrastive learning stages, validating their efficacy. For GPL-based backdoor attacks, Lyu *et al.* [Lyu *et al.*, 2024] propose a cross-context attack that uses a prompt-based mechanism to optimize the trigger graph and poison the pretrained GNN. Lin *et al.* [Lin *et al.*, 2024] propose a finetuning-resistant graph prompt poisoning method without poisoning the pretrained GNN.

However, existing methods focus on the intra-paradigm setting, where the surrogate model and the backdoored model belong to the same learning paradigm, resulting in attacker that are highly dependent on that specific paradigm. In contrast, our CP-GBA is characterized by two key aspects: (i) we tackle a new problem: transferring triggers across different paradigms under a realistic threat model, where attackers possess minimal prior knowledge and are unaware of the downstream learning paradigm. (ii) we are the first to investigate training a set of condensed subgraph triggers with GPL, improving their generalization and transferability.

3 Preliminary

3.1 Notaions

We represent a graph as $\mathcal{G} = (\mathcal{V}, \mathbf{A}, \mathbf{X})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the node set, $\mathbf{X} \in \mathbb{R}^{N \times d}$ is the node feature with d as the feature dimension, and N as the number of nodes. The adjacency matrix $\mathbf{A} \in \{0,1\}^{N \times N}$ indicates node connectivity, where $\mathbf{A}_{ij} = 1$ is an edge between nodes v_i and v_j, and $\mathbf{A}_{ij} = 0$ otherwise. In this paper, we focus on the semi-supervised node classification in an inductive setting. Specifically, the graph $\mathcal{G}$ is divided into two disjoint subgraphs: the labeled graph $\mathcal{G}_L$ and the unlabeled graph $\mathcal{G}_U$, with $\mathcal{V}_L \cap \mathcal{V}_U = \emptyset$. The labeled graph $\mathcal{G}_L$ is then split into three disjoint subsets: the labeled training graph $\mathcal{G}_T$, the validation graph $\mathcal{G}_{Va}$, and the testing graph $\mathcal{G}_{Te}$. We denote $\mathcal{G}_{Tr} = \mathcal{G}_U \cup \mathcal{G}_T$ as the training graph for model optimization, while $\mathcal{G}_{Va}$ and $\mathcal{G}_{Te}$ are used for validation and testing.

3.2 Threat Model

Attacker's Goal. The attacker aims to poison the training graph by injecting backdoor triggers into a small set of nodes and assigning them a predefined target class label. The GNN trained on this poisoned graph will then associate the trigger with the target class. As a result, it misclassifies trigger-injected nodes at test time while maintaining normal performance on clean nodes.

Attacker's Knowledge and Capability. Following prior studies [Zhang *et al.*, 2024a], we focus on gray-box backdoor attacks targeting node classification tasks. In a gray-box scenario, attackers have access to a small part of training data, including node attributes, graph structure, and label information, but do not know the specific architecture or parameters of the target model. In our work, the architecture includes both the GNN structure and the learning paradigm. Within a predefined budget, the attacker can inject triggers and assign target labels to nodes in the training graph.

3.3 Problem Formulation

Our preliminary analysis in Tab. 1 verifies that current attack strategies are ineffective for models under different learning paradigms. To overcome these limitations, we propose a novel transferable graph prompt attack trained with GPL.

We first construct a condensed subgraph trigger pool ($\mathcal{T}$) with diverse structural and feature patterns. Inspired by GPL, we adopt its mechanism to optimize the backdoor triggers, thereby improving their transferability. We denote $\oplus$ as the process of injecting prompts $p \in \mathcal{P}$, where $\mathcal{P}$ includes both token-based and subgraph-based prompts. We define $a_t(\cdot)$ as the operation of selecting and attaching a trigger from the trigger pool $\mathcal{T}$. Moreover, $f_\theta(\mathcal{G}^i)$ denotes the embedding of the local subgraph centered at node v_i using the pre-trained GNN model f_θ, followed by node-level classification via the classifier $f_c(\cdot)$. Given a clean attributed graph $\mathcal{G} = (\mathcal{V}, \mathbf{A}, \mathbf{X})$, let $\mathcal{V}_{Tr}$ denote the set of labeled nodes used for training with clean labels. Let $\mathcal{V}_P \subset \mathcal{V}_U$ be the set of poisoned nodes to which triggers are attached and the target label y_t is assigned. The optimization of backdoor triggers under the GPL setting can be formulated as:

$$
\begin{aligned}
&\min_{\theta_{\mathcal{T}}} \sum_{v_i \in \mathcal{V}_P} l(f_c(f_\theta(a_t(\mathcal{G}_P^i \oplus p, \mathcal{T})))), y_t), \\
&s.t. \theta_c^*, p^* = \arg\min_{\theta_c, p} \sum_{v_i \in \mathcal{V}_{Tr}} l\left(f_c\left(f_\theta\left(\mathcal{G}_{Tr}^i \oplus p\right)\right), y_i\right) \\
&\quad + \sum_{v_i \in \mathcal{V}_P} l\big(f_c\big(f_\theta\big(a_t\big(\mathcal{G}_P^i \oplus p, \mathcal{T}\big)\big)\big), y_t\big), \ |\mathcal{V}_P| \leq \Delta_p,
\end{aligned}
\tag{1}
$$

where $l(\cdot)$ is the cross-entropy loss and $\theta_{\mathcal{T}}$ represents the parameters of the subgraph trigger set $\mathcal{T}$. In the constraint, the number of poisoned nodes $|\mathcal{V}_P|$ is bounded by Δ_p.

4 Methodology

In this section, we detail our method, which optimizes Eq. (1) to conduct transferable graph backdoor attacks, as illustrated in Fig. 2. Our method consists of a set of condensed subgraph triggers $\mathcal{T}$, graph prompts $\mathcal{P}$ for GPL training, a frozen

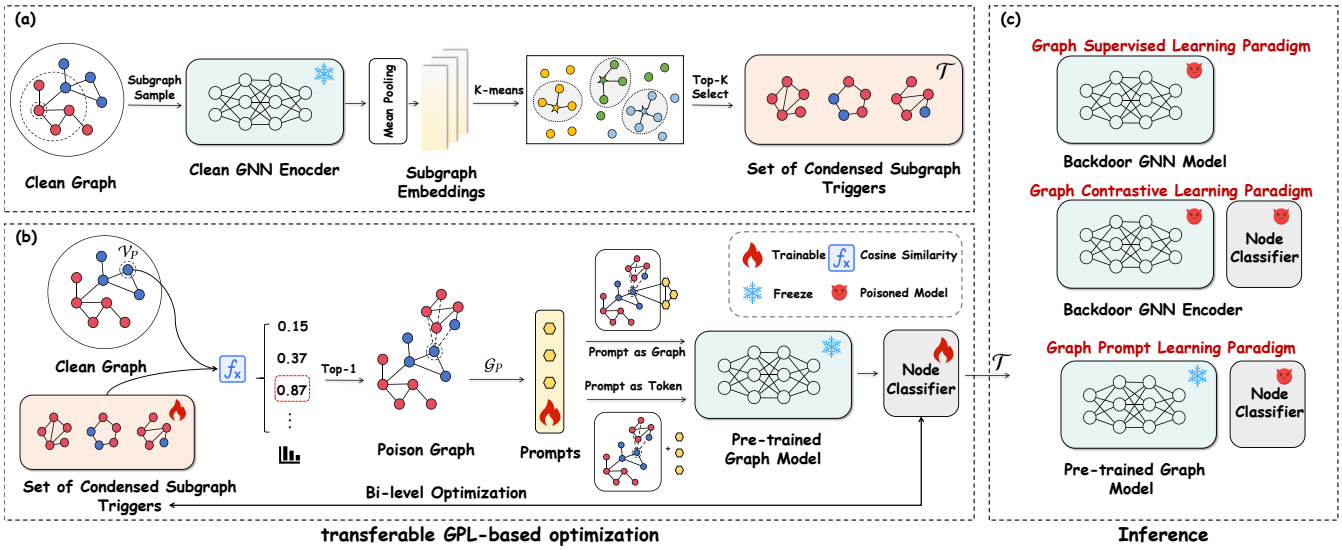

Figure 2: The overall framework of CP-GBA, comprising three phases. (a) Trigger Construction: Representative subgraphs are identified via K-means clustering on the embeddings of target-class subgraphs to initialize the condensed trigger set $\mathcal{T}$. (b) Transferable Optimization: Triggers from $\mathcal{T}$ are injected into target nodes based on similarity, followed by joint optimization with graph prompts $\mathcal{P}$. (c) Inference: The transferability of the optimized backdoor triggers is evaluated across diverse learning paradigms.

GNN encoder f_θ, and a surrogate node classifier f_c. Initially, we sample N subgraphs, each containing n nodes and centered on a node with the target class y_t, from the original graph $\mathcal{G}_{Tr}$ and select representative ones via clustering to form the initial condensed trigger set $\mathcal{T}$. With the frozen encoder f_θ and trainable classifier f_c, we perform transferable GPL-based optimization to jointly train $\mathcal{T}$ and $\mathcal{P}$. This ensures the robust generalization of $\mathcal{T}$ across diverse GNN architectures and learning paradigms.

4.1 The Set of Condensed Subgraph Triggers

As discussed in Sec. 1, existing adaptive trigger generators typically adopt shallow architectures and rely heavily on node-specific features, limiting their ability to generate triggers with structural diversity and rich feature representations. To overcome this limitation, we construct a set of condensed subgraph triggers $\mathcal{T}$ to effectively execute transferable backdoor attacks across different learning paradigms. Initially, we train a clean two-layer GCN encoder, denoted as f_{θ_c}, on the training graph $\mathcal{G}_{Tr}$ to obtain node representations, which is formulated as:

$$\hat{h}_i = f_{\theta_c}(\mathcal{G}^i_{Tr}), \tag{2}$$

$$\theta^*_c = \arg\min_{\theta_c} \sum_{v_i \in \mathcal{V}_{Tr}} l\left(\text{softmax}\left(\mathbf{W} \cdot \hat{h}_i + \mathbf{b}\right), y_i\right), \tag{3}$$

where $\mathbf{W}$ denotes the trainable matrix for classification, and $\mathbf{b}$ is the bias term. The GCN encoder f_{θ_c} is parameterized by θ_c, and $l(\cdot)$ is the cross-entropy loss. Here, $\hat{h}_i$ represents the embedding of node v_i, and y_i is its ground-truth label. The nodes of $\mathcal{V}_L$ with the target label y_t are selected as central nodes. We then employ a **B**readth-**F**irst **S**earch (BFS) algorithm to sample N subgraphs, each with n nodes as defined by the trigger size. We compute representations for these N sampled subgraphs using the encoder f_{θ_c} trained in Eq. (3). We then apply K-means clustering to these representations to select the K most representative subgraphs, which initialize $\mathcal{T} = \{t_1, t_2, \ldots, t_K\}$, where $t_i = (\mathbf{X}^t_i, \mathbf{A}^t_i)$ denotes the i-th trigger. This approach equips $\mathcal{T}$ with category-aware, feature-rich, and structure-preserving triggers, improving both diversity and stealthiness.

4.2 Enhancing Trigger Transferability

To promote model-agnostic triggers, we not only construct $\mathcal{T}$ but also employ GPL to improve trigger generalization. To further improve both the effectiveness and stealthiness of the attack, we adopt a bi-level optimization strategy. In the inner loop, we optimize the surrogate classifier f_c and prompts $\mathcal{P}$ with frozen GNN encoder f_θ and trigger set $\mathcal{T}$. The outer loop freezes both f_c and $\mathcal{P}$, while updating the trigger set $\mathcal{T}$.

Select and Inject Strategy. Due to the limitations of adaptive trigger generators that rely on poisoned node features, we adopt the strategy described in Sec. 4.1 to select the best-matching trigger from the condensed set $\mathcal{T}$ and inject it into the poisoned node. Unlike UGBA, which generates sample-specific triggers, our method employs a rule-based trigger selection and injection strategy.

$$score_i = \frac{1}{n} \sum_{h_i \in t_i} \frac{h_t \cdot h_i}{\|h_t\|_2 \|h_i\|_2}, \tag{4}$$

where n denotes the number of nodes in a trigger, t_i is the i-th trigger in $\mathcal{T}$, and h_t is the representation of the poisoned node. Each h_i is the representation of the i-th node in trigger t_i. The trigger with the highest similarity score is selected for backdoor injection, denoted by the selection function $a_t(\cdot)$.

Inner Loop. Before optimization, we randomly select a subset of unlabeled nodes from $\mathcal{V}_U$, attach triggers with the target label y_t, and denote this poisoned set as $\mathcal{V}_P$. Under the empirical risk minimization setting, we fix the trigger set $\mathcal{T}$ and the encoder f_θ, and train the graph prompts $\mathcal{P}$ along with the surrogate classifier f_c by minimizing the loss in Eq. (5) on the poisoned graph.

$$\min_{p,\theta_c} \mathcal{L}_p(\theta_c, p, \theta_{\mathcal{T}}) = \sum_{v_i \in \mathcal{V}_{Tr}} l\left(f_c\left(f_\theta\left(\mathcal{G}^i_{Tr} \oplus p\right)\right), y_i\right) + \sum_{v_i \in \mathcal{V}_P} l\left(f_c\left(f_\theta\left(a_t\left(\mathcal{G}^i_P \oplus p, \mathcal{T}\right)\right)\right), y_t\right), \tag{5}$$

where θ_c and $\theta_{\mathcal{T}}$ denote the parameters of the surrogate classifier and the condensed subgraph trigger set $\mathcal{T}$, respectively. $\mathcal{G}^i_{Tr}$ is the clean subgraph centered at node v_i with ground-truth label y_i, and y_t is the target label specified by the attacker. $\mathcal{G}^i_P \oplus p$ represents the prompted graph. The function $a_t(\cdot)$ selects the trigger from $\mathcal{T}$ based on the Eq. (4).

Outer Loop. The set of condensed subgraph triggers $\mathcal{T}$ is then optimized to mislead the surrogate classifier f_c, such that the frozen GNN encoder f_θ produces embeddings for nodes in $\mathcal{V}_U$ that are classified as the target label y_t when attached with a selected trigger and prompted with p. Formally, the objective is defined as:

$$\mathcal{L}_{Trans} = \sum_{v_i \in \mathcal{V}_U} l\left(f_c\left(f_\theta\left(a_t\left(\mathcal{G}^i_P \oplus p, \mathcal{T}\right)\right)\right), y_t\right), \tag{6}$$

Through Eq. (6), the trigger set $\mathcal{T}$ inherits both category-specific knowledge and transferability similar to that of prompts. Although $\mathcal{T}$ is sampled from the original graph and condensed via Eq. (4), preserving structural consistency and improving stealthiness, it remains crucial to model both the connections between trigger and target nodes, and the internal connectivity among trigger nodes. To address this, we define a stealthiness loss:

$$\mathcal{L}_{Ste} = \sum_{v_i \in \mathcal{V}_P} \sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \mathcal{E}^i_t} \max\left(0, \tau_{sim} - \frac{\mathbf{x}_j \cdot \mathbf{x}_k}{\|\mathbf{x}_j\|_2 \|\mathbf{x}_k\|_2}\right), \tag{7}$$

where $\mathcal{E}^i_t$ denotes the set of edges between the injected trigger nodes and node v_i, τ_{sim} is a similarity threshold, and $\mathbf{x}_j$, $\mathbf{x}_k$ are the feature vectors of nodes v_j and v_k.

The loss $\mathcal{L}_p$ in Eq. (5) optimizes the surrogate classifier f_c to classify clean nodes correctly while also predicting y_t for poisoned nodes. Meanwhile, $\mathcal{L}_{\text{Trans}}$ in Eq. (6) improves the transferability of the trigger set $\mathcal{T}$, and $\mathcal{L}_{\text{Ste}}$ in Eq. (7) improves its stealthiness. The final bi-level optimization objective can be formulated as:

$$\begin{aligned} &\min_{\theta_{\mathcal{T}}} \mathcal{L}_t(\theta^*_c, p^*, \theta_{\mathcal{T}}) = \mathcal{L}_{Trans} + \lambda \mathcal{L}_{Ste}, \\ &\text{s.t.} \quad \theta^*_c, p^* = \arg\min_{\theta_c, p} \mathcal{L}_p(\theta_c, p, \theta_{\mathcal{T}}), \end{aligned} \tag{8}$$

where λ is a trade-off coefficient that balances transferability and stealthiness.

4.3 Why It Works

In this section, we explore the transferability of CP-GBA from the theoretical aspect of GPL.

Theorem 1. *In node-level, the model GNN f, which is trained with a large amount of high-quality data, has the ability to map any node in graph $\mathcal{G}_i$, known or unknown, to all feature spaces surjectively (*i.e*, $f : \mathcal{G}_i \to \mathbb{R}^d$, where d is the class number dimension.).*

Theorem 2. *Let f_θ be a GNN model trained on upstream datasets D_{up} with frozen parameters (θ); let T_{dow} be the downstream task and C is an optimal function to T_{dow}. Given any graph $\mathcal{G}_{ori}$, $C(\mathcal{G}_{ori})$ denotes the optimal embedding vector to the downstream task (*i.e.*, can be parsed to yield correct results for $\mathcal{G}_{ori}$ in the downstream task), then there always exists a bridge graph G_{bri} such that $f_\theta(\mathcal{G}_{bri}) = C(\mathcal{G}_{ori})$.*

Definition of the bridge graph:

$$G_{bri} = G_{ori} \oplus \mathcal{T} \tag{9}$$

Given a pre-trained model f, the downstream graph $\mathcal{G}_{\text{ori}}$ augmented with prompt p yields the output representation $\hat{h} = f(\mathcal{G}_{\text{ori}} \oplus p)$. Unlike prior backdoor attacks [Dai *et al.*, 2023; Lyu *et al.*, 2024], our framework leverages the surjective mapping property of the pre-trained model f (Theorem 3), ensuring access to the continuous feature space $\mathbb{R}^d$. Furthermore, Theorem 4 guarantees the existence of a bridge graph that elicits optimal downstream representations via prompt augmentation. **Building on these guarantees, GPL-based optimization enables the trigger to function as a learnable approximation of this theoretical bridge graph.** By dynamically adapting the trigger to minimize the prompt-tuning loss, our method effectively aligns the poisoned graph with the optimal bridge structure. This alignment empowers the trigger to inherit the intrinsic transferability of prompts, ensuring robust generalization across diverse downstream tasks, architectures, and learning paradigms.

5 Experiments

5.1 Experimental Settings

Datasets. To evaluate the effectiveness of CP-GBA, we conduct experiments on four widely used real-world datasets, *i.e.*, **Cora**, **Pubmed**, **Facebook** and **OGB-arxiv** [Sen *et al.*, 2008; Hu *et al.*, 2020; Rozemberczki *et al.*, 2021], which serve as standard benchmarks for inductive semi-supervised node classification. The dataset statistics are summarized in Tab. 3. s **Compared Methods.** Following a similar setting to DPGBA [Zhang *et al.*, 2024a], we compare CP-GBA with four representative graph backdoor attack methods, including **SBA** [Zhang *et al.*, 2021b], **GTA** [Xi *et al.*, 2021], **UGBA** [Dai *et al.*, 2023], and **DPGBA** [Zhang *et al.*, 2024a]. To assess the stealthiness of CP-GBA, we employ three representative defenses, **Prune** [Dai *et al.*, 2023], **OD** [Zhang *et al.*, 2024a], and **RIGBD** [Zhang *et al.*, 2025], which detect backdoors based on attribute similarity, distribution anomalies, and random edge dropping, respectively. All hyperparameters are selected based on validation performance.

Backbone GNN Models and Learning Methods. Following the taxonomy in [Sun *et al.*, 2023b; Ju *et al.*, 2024], we

Dataset	Defense	SBA		GTA		UGBA		DPGBA		CP-GBA	
		ACC(AD)	ASR	ACC(AD)	ASR	ACC(AD)	ASR	ACC(AD)	ASR	ACC(AD)	ASR
Cora	None	0.57(+0.04)	0.50	0.57(+0.04)	0.50	0.59(+0.02)	0.63	0.60(+0.01)	0.44	0.64(-0.03)	0.96
	Prune	0.58(+0.03)	0.41	0.57(+0.04)	0.39	0.60(+0.01)	0.47	0.57(+0.04)	0.48	0.64(-0.03)	0.96
	OD	0.58(+0.03)	0.39	0.57(+0.04)	0.38	0.58(+0.03)	0.47	0.58(+0.03)	0.49	0.64(-0.03)	0.96
	RIGBD	0.58(+0.03)	0.17	0.57(+0.04)	0.20	0.59(+0.02)	0.21	0.58(+0.03)	0.22	0.64(-0.03)	0.89
Pubmed	None	0.46(+0.25)	0.62	0.49(+0.22)	0.68	0.73(+0.02)	0.70	0.72(+0.03)	0.67	0.71(+0.04)	0.96
	Prune	0.44(+0.27)	0.35	0.45(+0.26)	0.39	0.72(-0.01)	0.63	0.73(-0.02)	0.68	0.71(+0.00)	0.97
	OD	0.45(+0.26)	0.39	0.46(+0.25)	0.36	0.72(-0.01)	0.69	0.72(-0.01)	0.67	0.71(+0.00)	0.96
	RIGBD	0.44(+0.27)	0.22	0.45(+0.26)	0.24	0.72(-0.01)	0.39	0.72(-0.01)	0.45	0.71(+0.00)	0.90
Facebook	None	0.70(-0.02)	0.22	0.67(+0.01)	0.41	0.67(+0.01)	0.65	0.66(+0.02)	0.48	0.68(+0.00)	0.94
	Prune	0.68(+0.00)	0.22	0.68(+0.00)	0.23	0.68(+0.00)	0.77	0.68(-0.03)	0.51	0.67(+0.01)	0.95
	OD	0.68(+0.00)	0.20	0.69(-0.01)	0.37	0.69(-0.01)	0.73	0.68(+0.00)	0.53	0.67(+0.01)	0.95
	RIGBD	0.67(+0.01)	0.17	0.68(+0.00)	0.26	0.68(+0.00)	0.47	0.67(+0.01)	0.39	0.67(+0.01)	0.88
OGB-arxiv	None	0.44(+0.03)	0.19	0.45(+0.02)	0.24	0.46(+0.01)	0.70	0.48(-0.01)	0.73	0.48(-0.01)	0.92
	Prune	0.46(+0.01)	0.18	0.46(+0.01)	0.20	0.46(+0.01)	0.66	0.47(+0.00)	0.69	0.48(-0.01)	0.91
	OD	0.46(+0.01)	0.16	0.46(+0.01)	0.19	0.46(+0.01)	0.55	0.47(+0.00)	0.66	0.48(-0.01)	0.91
	RIGBD	0.45(+0.02)	0.15	0.45(+0.02)	0.17	0.47(+0.00)	0.42	0.48(-0.01)	0.60	0.48(-0.01)	0.91

Table 2: Graph backdoor attack results (ACC(AD) | ASR) under different attack scenarios. The top two performances are highlighted.

Datasets	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,443	7
Pubmed	19,717	44,338	500	3
Facebook	22,470	342,004	128	4
OGB-arxiv	169,343	1,166,243	128	40

Table 3: Dataset statistics

evaluate CP-GBA across three learning paradigms: **GSL**, including standard (GAT [Velickovic *et al.*, 2017], GCN [Kipf and Welling, 2016], GraphSAGE [Hamilton *et al.*, 2017], GT [Yun *et al.*, 2019]) and robust architectures (GNNGuard [Zhang and Zitnik, 2020], RobustGCN [Zhu *et al.*, 2019]); **GCL** (GRACE [Zhu *et al.*, 2020], CCA-SSG [Zhang *et al.*, 2021a]); and **GPL** (GPF [Jiang *et al.*, 2024], GraphPrompt [Liu *et al.*, 2023], All-in-one [Sun *et al.*, 2023a])

Evaluation Protocol. Following a similar setting to DPGBA [Zhang *et al.*, 2024a], we randomly select 20% of the nodes from the original dataset to serve as test nodes for evaluation. Among these test nodes, half are designated as target nodes for evaluating attack performance. The remaining half are used as clean test nodes to assess the prediction accuracy of backdoored models on clean samples. The graph containing the remaining 80% of nodes is used as the training graph, where 20% of the nodes are labeled. To evaluate the backdoor attacks, we report the average attack success rate (ASR) on the target node set, accuracy (ACC) on clean test nodes, clean accuracy (CA), and accuracy drop (AD) [Zhang *et al.*, 2023], where AD measures the performance degradation compared with the clean model. To assess the transferability and generalizability of CP-GBA, we repeat the experiments five times on each GNN architecture and learning paradigm and report the average performance.

5.2 Attack Performance

We evaluate the transferability and stealthiness of CP-GBA against baselines across four datasets and three learning paradigms under different defense strategies. As shown in Tab. 7, we summarize the following key observations:

- Baselines show poor performances across different learning paradigms, which is consistent with the findings in Tab. 1, indicating that existing attack strategies are overly reliant on model structures and learning paradigms.
- Without applying defense strategies, CP-GBA achieves the highest ASR over all baseline methods across all datasets, showing its generalizability and transferability with respect to different model structures and learning paradigms. When equipped with defense strategies, CP-GBA maintains high ASR, comparable to the no-defense setting.

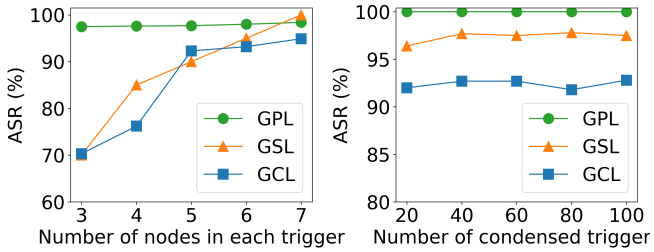

Figure 3: ASR in different attack budgets on Cora

5.3 Impact of Attack Budget

We investigate the impact of attack budgets by varying the trigger count in {20, 40, 60, 80, 100}, and size in {3, 4, 5, 6, 7}, while holding all other parameters frozen. Figure 3 illustrates the ASR results on Cora, with consistent trends observed across other datasets. We report only ASR, as clean accuracy exhibits negligible fluctuations.

As the number of condensed triggers increases, the ASR shows minimal variance and remains at a high level. This suggests that only 20 triggers are sufficient to maintain strong generalizability and effectiveness, as well as the stealthiness of the backdoor attack during graph poisoning.

As the number of nodes in each trigger increases, ASR shows a clear upward trend and plateaus when the trigger size reaches five nodes. This indicates a 5-node trigger suffices to

Data	Defense	GPL			GCL		GSL	
		GPF	G-Prompt	AIO	GRACE	CCA-SSG	GCN	GAT
Cora	None	73.8|36.2	94.9|32.5	95.5|34.3	91.5|76.5	90.3|76.9	83.7|82.4	80.9|83.3
	Prune	73.3|36.1	94.1|32.4	95.3|34.1	90.7|76.3	90.1|76.3	83.9|82.4	81.2|83.8
	OD	74.1|36.4	93.9|32.4	95.5|33.8	90.9|76.4	89.7|76.9	84.1|82.5	81.3|83.4
	RIGBD	71.3|36.1	91.5|32.2	93.9|33.9	87.3|76.4	87.8|76.9	80.8|82.4	78.6|83.5
Pubmed	None	93.3|40.8	93.9|44.1	98.6|50.7	87.8|85.2	87.2|83.3	86.9|84.0	86.4|84.7
	Prune	94.0|41.7	93.9|43.7	98.5|49.7	87.4|85.3	86.6|83.0	87.1|84.2	87.0|84.4
	OD	94.0|40.9	94.2|43.5	98.6|49.5	86.9|85.0	86.8|83.2	87.0|84.1	86.6|84.0
	RIGBD	91.3|42.7	90.9|44.2	94.8|50.3	82.1|85.3	84.2|83.3	83.7|84.0	83.9|84.2
Facebook	None	90.1|50.3	91.8|35.1	92.4|33.3	85.2|78.1	83.3|80.7	80.5|86.1	81.9|85.7
	Prune	89.5|50.3	92.5|35.2	93.8|33.9	84.5|78.3	83.4|80.6	79.8|86.3	82.2|85.9
	OD	90.2|50.7	91.7|35.2	92.4|33.6	84.7|78.3	83.3|80.3	79.5|86.1	82.0|85.7
	RIGBD	87.7|50.9	88.9|35.1	90.3|33.5	81.3|78.3	80.9|80.2	77.6|86.3	80.1|85.6
OGB-arxiv	None	88.6|25.1	89.4|26.1	91.0|30.3	83.2|53.1	81.4|52.7	78.5|65.1	77.4|64.7
	Prune	88.5|25.3	88.2|26.2	90.8|30.9	82.5|53.3	81.7|52.6	77.8|65.3	76.9|64.7
	OD	87.7|25.7	88.7|26.2	90.4|30.2	82.7|53.5	80.9|52.3	77.5|65.1	77.0|64.7
	RIGBD	87.5|25.7	88.8|26.2	90.2|30.7	81.6|53.3	80.3|52.3	78.1|65.1	77.6|64.7

Table 4: Backdoor attack results (ASR (%) | CA (%)) on different training surrogate models from different learning paradigms.

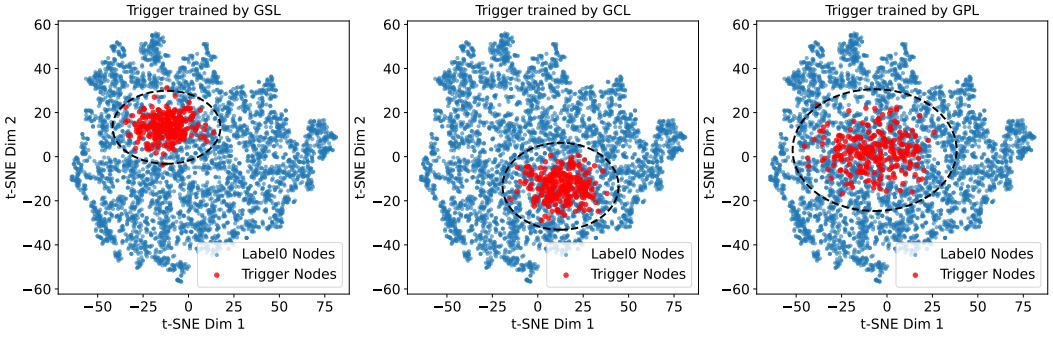

Figure 4: t-SNE visualization of the feature embeddings for the trigger nodes and the origin nodes on the Pubmed dataset, after training with different paradigms: Graph Supervised Learning (left), Graph Contrastive Learning (middle), and Graph Prompt Learning (right).

capture category-aware information.

5.4 Trigger Feature Distribution

To investigate how GPL improves the generalization of triggers, we extract the embeddings of the target model for both the original nodes and the trigger nodes trained with different paradigms and visualize these embeddings using t-SNE, as shown in Fig. 4. We can observe that for a specific attack category, the triggers trained by GPL, while remaining close to the original node features, show a more widespread distribution that more closely aligns with the overall distribution of the original graph. This generalization is crucial for achieving model-agnostic and paradigm-agnostic attacks, because the trigger representations avoid overfitting to a narrow region of the feature space, ensuring effectiveness across various model architectures and learning paradigms.

5.5 Ablation Study

We conduct ablation studies to investigate how GPL improves the generalizability and transferability of subgraph triggers across different model architectures and learning paradigms. As shown in Tab. 4, we replace the subgraph trigger optimization method based on GPL with GSL and GCL. From the experimental results, we can observe:

- Under GPL optimization, All-in-one consistently outperforms token-based baselines (GPF and GraphPrompt). We attribute this superiority to the structural alignment between its subgraph prompts and our trigger design. Furthermore, subgraph-based prompts are likely to encode richer structural and semantic information than token-based prompts, leading to stronger transferability.
- The suboptimal ASR in GCL and GSL paradigms highlights their limited capacity for trigger transferability. This empirically validates the superiority of GPL in facilitating the learning of robust, transferable backdoor triggers.

6 Conclusion

In this paper, we present both theoretical and empirical investigations on the transferability of backdoor attacks across diverse attack scenarios. To overcome the poor transferability across attack scenarios, we identify two key challenges: (1) overreliance on the training paradigm and (2) simplistic adaptive trigger generators. To this end, we propose CP-GBA, a transferable backdoor attack that employs a set of condensed subgraph triggers to enrich structural features and preserve distributional consistency. Specifically, the transferability of GPL is utilized to optimize the subgraph triggers, enabling them to be model-agnostic and ensuring attack effectiveness across diverse scenarios. Extensive experiments on real-world datasets confirm the effective performance of CP-GBA under various attack settings.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 62402341).

Appendix

Proof

Theorem 3. *In node-level, the model GNN f, which is trained with a large amount of high-quality data, has the ability to map any node in graph $\mathcal{G}_i$, known or unknown, to all feature spaces surjectively (*i.e*, $f : \mathcal{G}_i \to \mathbb{R}^d$, where d is the class number dimension.).*

Proof. Let Φ_{GNN} be a GNN with L layers and injective functions. By its equivalence to the L-iteration WL test, this GNN maps non-isomorphic L-hop neighborhoods, $[G_{v,L}]$, to distinct embeddings, $h_v^{(L)}$. Thus, the map Φ is injective:

$$\Phi : [Gend_{v,L}] \mapsto h_v^{(L)} \tag{10}$$

A local and permutation-invariant target function $f(v, G)$ depends only on the neighborhood's isomorphism class, so it can be factored as:

$$f(v, G) = g([G_{v,L}]) \tag{11}$$

Because Φ is injective, we can define a continuous function g' on the GNN's output space, $\text{Im}(\Phi)$, such that:

$$g'(h_v^{(L)}) = g([G_{v,L}]) \tag{12}$$

By the Universal Approximation Theorem, there exists an MLP, Ψ_{MLP}, that can approximate g' to arbitrary precision ϵ:

$$\|\Psi_{\text{MLP}}(h_v^{(L)}) - g'(h_v^{(L)})\| < \epsilon \quad \text{for all } h_v^{(L)} \in \text{Im}(\Phi). \tag{13}$$

The composite model $F(v, G)$ is defined as:

$$F(v, G) = \Psi_{\text{MLP}}(\Phi_{\text{GNN}}(v, G)) \tag{14}$$

This model therefore approximates $f(v, G)$, since:

$$\|F(v, G) - f(v, G)\| = \|\Psi_{\text{MLP}}(h_v^{(L)}) - g'(h_v^{(L)})\| < \epsilon. \tag{15}$$

□

Corollary. Any surjective function f mapping d distinct local graph structures to d distinct classes satisfies the theorem's preconditions. Since a GNN can approximate this f, it is therefore capable of surjectivity onto the set of d class labels.

Theorem 4. *Let f_θ be a GNN model trained on upstream datasets D_{up} with frozen parameters (θ); let T_{dow} be the downstream task and C is an optimal function to T_{dow}. Given any graph $\mathcal{G}_{ori}$, $C(\mathcal{G}_{ori})$ denotes the optimal embedding vector to the downstream task (*i.e.*, can be parsed to yield correct results for $\mathcal{G}_{ori}$ in the downstream task), then there always exists a bridge graph G_{bri} such that $f_\theta(\mathcal{G}_{bri}) = C(\mathcal{G}_{ori})$.*

Proof. For a given G_{ori} and a downstream task T_{dow}, the embedding vector corresponding to the downstream task is formally defined as the embedding vector produced by the optimal downstream model for T_{dow}, which is thus uniquely determined.

Given our previous definition for the Theorem 3, the $F_{\hat{\theta}}$ discussed here can be a surjective mapping from the graph space $\{G\}$ to $\mathbb{R}^d$. According to the properties of surjective mappings, for this particular $C(G_{ori}) \in \mathbb{R}^d$, there must exist a special graph $\hat{G}_{bri}$ such that:

$$F_{\hat{\theta}}(\hat{G}_{bri}) = C(G_{ori}) \tag{16}$$

Definition of the bridge graph:

$$G_{bri} = G_{ori} \oplus \mathcal{T} \tag{17}$$

Upon examining the definition of G_{bri}, we find that $\hat{G}_{bri} = G_{bri}$. Theorem 2 is thereby proved. □

Detailed Implementation

For the **subgraph trigger optimization**, the condensed triggers are first selected based on a two-layer GCN model trained on each corresponding dataset. These triggers are then optimized using GraphPrompt [Liu *et al.*, 2023], where the backbone encoder is a frozen three-layer GCN with a sum pooling layer, pre-trained by GRACE [Zhu *et al.*, 2020] on the Cora dataset. The classifier head is a trainable two-layer MLP. For the **graph backdoor attack test stage**: 1) **GSL**: We use two-layer variants of each corresponding GNN architecture; 2) **GCL**: We use a two-layer GCN as the encoder and a two-layer MLP as the classifier, trained in a two-stage manner on the poisoned graph; 3) **GPL**: We use a frozen two-layer GCN pretrained on the Cora dataset as the backbone encoder and a trainable two-layer MLP classifier, with three prompt nodes injected for training on the poisoned graph. For a fair comparison, all hyperparameters are selected based on the model performance on validation set. All models are trained on a NVIDIA A6000 GPU with 48GB of memory.

Time Complexity Analysis

Let h denote the embedding dimension, n the number of nodes per trigger, K the number of triggers in $\mathcal{T}$, and $|\mathcal{N}_S|$ the number of candidate subgraphs to extract.

The $\mathcal{T}$ Construction. The cost is approximately $\mathcal{O}(nh|\mathcal{N}_S| + Kh|\mathcal{N}_S| + Kh)$, accounting for subgraph extraction and K-means clustering into K groups. As K-means clustering is the most computationally intensive step, the overall complexity is approximated as $\mathcal{O}(Kh|\mathcal{N}_S|)$.

Optimization. Each outer iteration in the bi-level optimization consists of updating the surrogate classifier in the inner loop and optimizing the condensed trigger set $\mathcal{T}$. The cost of updating the surrogate model is $\mathcal{O}(Nhd|\mathcal{V}|)$, where d is the average node degree, N is the number of inner iterations, and $|\mathcal{V}|$ is the number of training and poisoned nodes. For trigger optimization, computing $\mathcal{L}_{Trans}$ incurs $\mathcal{O}(hd|\mathcal{V}_U|)$, where $|\mathcal{V}_U|$ is the number of unlabeled nodes. Optimizing $\mathcal{L}_{Ste}$ costs $\mathcal{O}(h|\mathcal{V}_P||\mathcal{V}_a|)$, where $|\mathcal{V}_P|$ and $|\mathcal{V}_a|$ are the numbers of poisoned nodes and attached nodes, respectively. Given that $|\mathcal{V}_P| \ll |\mathcal{V}|$, $|\mathcal{V}_P||\mathcal{V}_a| \ll |\mathcal{V}|$, and

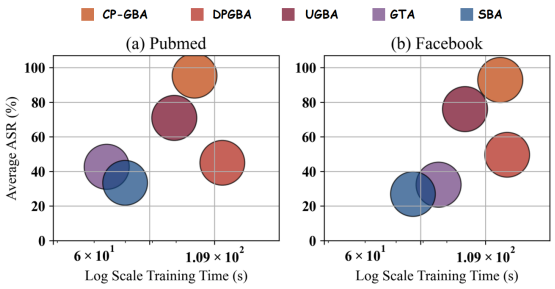

Figure 5: Training time of triggers vs. performance

$|\mathcal{V}_U| \approx |\mathcal{V}|$, the overall time complexity per outer iteration is $\mathcal{O}((N+1)hd|\mathcal{V}|)$, which is comparable to that of UGBA. During the backdoor attack phase, selecting and attaching a trigger incurs $\mathcal{O}(Khn + Kh + h)$, where feature extraction dominates. Thus, the overall cost is $\mathcal{O}(Khn)$. This analysis indicates that **CP-GBA** scales well to large graphs.

We present the pseudo-code for CP-GBA in Algorithm 1. Furthermore, to demonstrate the practical efficiency of our optimization strategy, we report the training time of triggers across different datasets. As depicted in Fig. 5, our method achieves a favorable balance between attack performance and computational cost.

Algorithm 1 Algorithm of CP-GBA

Require: Original Graph $\mathcal{G}$, Target Label y_t, Parameter λ
Ensure: The Set of Condensed Subgraph Triggers ($\mathcal{T}$)

```
Initialize backdoored graph G_B = G;
Separate the training graph G_Tr from labeled graph G_L;
Select poisoned nodes V_P based on the cluster algorithm from UGBA[Dai et al., 2023];
Randomly initialize node classifier θ_c and prompts P;
Initialize T with θ_T as parameter based on the construction of condensed riggers in Sec.4.1;
while not converged do
    for t=1,2,...,N do
        Update θ_c by ∇_{θ_c} L_p based on Eq. (5);
        Update P by ∇_P L_p based on Eq. (5);
    end for
    Update θ_T by ∇_{θ_T}(L_Trans + λ L_Ste) based on Eq. (8);
end while
return T;
```

Additional Experiments

To further assess the architectural generalizability of CP-GBA, we extend our training to include diverse surrogate models (GraphTransformer [Yun *et al.*, 2019], GAT [Veličković *et al.*, 2018], and GraphSAGE [Hamilton *et al.*, 2018]) with results detailed in Tab. 5. Additionally, we evaluate the attack's resilience against advanced defense mechanisms by testing against robust GNNs such as GNNGuard [Zhang and Zitnik, 2020] and RobustGCN [Zhu *et al.*, 2019] (results in Tab. 6). For a comprehensive overview of performance across all settings, the full results are summarized in Tab. 7. Experimental results on different surrogate and target models further prove the generalizability and stealthiness of our CP-GBA approach.

Table 5: Average backdoor attack results (ASR(%)| CA (%))where GT stands for GraphTransformer and triggers are trained by GPL with more surrogate models.

Dataset	Defense	GT	GAT	GraphSAGE
Cora	None	98.1 | 79.2	94.8 | 83.3	98.3 | 82.8
	Prune	98.7 | 78.5	95.2 | 83.8	97.6 | 82.9
	OD	98.4 | 78.6	94.5 | 83.4	97.4 | 83.1
	RIGBD	94.3 | 78.7	92.5 | 83.6	94.7 | 83.0
Pubmed	None	96.6 | 87.1	94.0 | 84.5	95.3 | 85.1
	Prune	96.7 | 87.6	94.1 | 84.4	94.9 | 85.0
	OD	96.5 | 87.2	93.7 | 84.2	94.8 | 84.8
	RIGBD	93.7 | 87.2	90.7 | 84.3	92.2 | 85.1
Facebook	None	88.6 | 87.1	83.9 | 85.2	84.7 | 86.0
	Prune	88.3 | 86.6	84.1 | 85.6	84.0 | 86.2
	OD	88.5 | 87.2	83.8 | 85.5	84.3 | 86.0
	RIGBD	86.5 | 87.2	82.7 | 85.4	82.6 | 86.1
OGB-arxiv	None	87.4 | 65.5	87.5 | 64.3	87.2 | 65.7
	Prune	86.7 | 65.6	86.6 | 64.5	86.9 | 66.2
	OD	86.9 | 65.2	86.2 | 64.3	86.8 | 66.1
	RIGBD	84.5 | 65.2	84.7 | 64.6	85.8 | 65.8

Table 6: Average backdoor attack results (ASR(%)| CA (%))against more GSL-based GNNs where triggers are trained by GPL.

Model	Defense	Cora	Pubmed	Facebook	OGB-arxiv
RobustGCN	None	99.1 | 79.2	94.8 | 84.1	95.2 | 84.3	86.9 | 61.3
	Prune	99.7 | 78.5	95.2 | 83.9	95.4 | 84.1	87.0 | 61.1
	OD	100.0 | 78.3	95.5 | 84.2	95.4 | 84.2	86.7 | 61.6
	RIGBD	97.3 | 78.8	92.7 | 84.3	91.3 | 84.2	85.4 | 61.7
GNNGuard	None	86.4 | 75.7	81.3 | 86.8	92.9 | 85.0	83.7 | 60.3
	Prune	87.1 | 76.2	80.9 | 85.9	94.2 | 85.2	83.4 | 60.8
	OD	87.3 | 76.2	82.1 | 86.2	93.0 | 85.1	82.9 | 60.7
	RIGBD	83.7 | 76.1	79.8 | 86.4	90.1 | 85.3	81.5 | 60.7

Details of Compared Methods and Defense Strategies

The compared methods are detailed as follows:

- **SBA-Samp [Zhang *et al.*, 2021b]**: This method targets backdoor attacks on graph classification by injecting a fixed subgraph as a trigger into the training graph for a poisoned node. The edges of each subgraph are generated using the Erdos-Renyi (ER) model, and the node features are randomly sampled from the training graph, ensuring variability in the features of the injected subgraph.
- **SBA-Gen [Zhang *et al.*, 2021b]**: This is a variant of SBA-Samp, SBA-Gen utilizes generated features for the trigger nodes instead of sampling from the training graph. The features for the triggers are generated from a Gaussian distribution, the parameters of which (mean and variance) are calculated based on the feature distribution of real nodes from the graph. This approach aims to create more realistic triggers for the backdoor attack.
- **GTA [Xi *et al.*, 2021]**: This method addresses backdoor attacks on both graph and node classification. It

starts by randomly selecting unlabeled nodes from the clean graph as poison nodes. An adaptive trigger generator is then used to create node-specific subgraphs as triggers. The trigger generator is optimized through a bi-optimization algorithm that incorporates backdoor attack loss.

- **UGBA [Dai *et al.*, 2023]**: Similar to GTA, UGBA focuses on backdoor attacks on node classification and employs an adaptive trigger generator to generate node-specific triggers. To enhance the unnoticeability of the attack, UGBA introduces a clustering algorithm to select representative nodes as poison nodes. This method also explores the use of an unnoticeable loss function to increase the similarity between attacked nodes and generated triggers, improving the stealthiness of the backdoor attacks.
- **DPGBA [Zhang *et al.*, 2024a]**: DPGBA focuses on in-domain (ID) trigger generation for the backdoor attacks on node classification. To generate ID triggers, DPGBA introduce an out-of-distribution (OOD) detector in conjunction with an adversarial learning strategy to generate the attributes of the triggers within distribution. This method further introduces novel modules designed to enhance trigger memorization by the victim model trained on poisoned graph.

The details of defense strategies are described as follows:

- **Prune [Dai *et al.*, 2023]**: In this strategy, we focus on enhancing the resilience of GNNs to graph backdoor attacks by pruning edges that connect nodes with low cosine similarity. This approach is based on the observation that edges created by backdoor attackers often link nodes with dissimilar features, aiming to manipulate the model's predictions subtly. By pruning such edges, we can potentially disrupt the structure of the trigger inserted by the attacker, making it less effective and thus preserving the integrity of the data representation of graph.
- **OD [Zhang *et al.*, 2024a],**: Building upon the Prune strategy, this approach adds an extra defense against backdoor attacks by addressing the issue of "dirty" label on nodes that may have been compromised by the attacker. In addition to pruning edges between dissimilar nodes, we also discard the labels of these nodes to mitigate the influence of potentially poisoned labels. This dual approach helps in further safeguarding the learning process against manipulation. By removing these labels, the defense mechanism reduces the risk of the model learning from and perpetuating the attacker's modifications, thereby maintaining the performance and trustworthiness of GNNs in the face of adversarial conditions.
- **RIGBD [Zhang *et al.*, 2025]**: This method leverages the inherent robustness gap between benign and malicious graph structures. The core premise is that backdoor triggers, unlike robust benign topological features, are structurally rigid and highly sensitive to perturbations. By selectively filtering out these fragile connections during training, it effectively dismantles the trigger patterns required for the attack, thereby neutralizing the backdoor while retaining the graph's essential semantic utility.

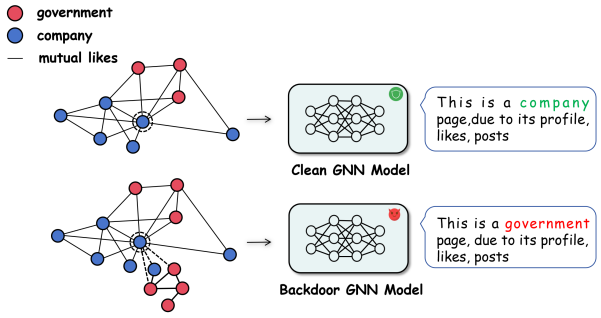

Figure 6: Case Study on Facebook dataset

Case Study

In real-world scenarios such as Facebook, the set of condensed subgraph triggers may include numerous malicious user pages. For example, a subgraph trigger may consist of 5 nodes, each representing a user page (*i.e.*, government, TV show, company, or politician). Edges between nodes represent mutual likes or interactions. Attackers induce misclassification by selecting a suitable trigger from the set based on the target user's characteristics and linking it to the target. As illustrated in Fig. 6, the blue node, originally a company page, may be mislabeled by the backdoor model as a government page after being followed by malicious accounts.

Discussion

6.1 Findings

This work yields several key findings:

- **Plateau for Larger Triggers:** The attack is constrained by the GNN's limited propagation range and the original graph's degree distribution. Large triggers introduce structural anomalies, compromising both efficiency and stealthiness.
- **Batch Efficiency of Trigger Sets:** The trigger set achieves superior performance over MLP-generated triggers in large batches, enabled by its parallel, database-like storage and retrieval.

6.2 Future Direction

Building on this study, several promising directions for future research emerge:

- **Cross Task Attack:** Existing graph backdoor attacks are task-specific, either node-level or graph-level predictions. Generalizing them to multi-task settings offers a promising path toward a more universal attack.
- **Limited Data Access:** Current backdoor attacks often require access to training data and labels. Designing data-efficient attacks that operate with extremely limited or zero data access is a critical future direction, increasing their real-world threat.

Table 7: Graph backdoor attack results (ACC(AD) | ASR) under different attack scenarios. The top two performances in ASR are highlighted in blue and yellow.

Dataset	Method	Defense	SBA		GTA		UGBA		DPGBA		CP-GBA	
			ACC(AD)	ASR	ACC(AD)	ASR	ACC(AD)	ASR	ACC(AD)	ASR	ACC(AD)	ASR
Cora	GSL	None	0.81(+0.00)	0.58	0.82(-0.01)	0.75	0.82(-0.01)	0.76	0.81(+0.00)	0.78	0.81(+0.00)	0.97
		Prune	0.80(+0.01)	0.68	0.82(-0.01)	0.39	0.82(-0.01)	0.69	0.81(+0.00)	0.72	0.81(+0.00)	0.97
		OD	0.81(+0.00)	0.68	0.81(+0.00)	0.48	0.82(-0.01)	0.69	0.81(+0.00)	0.72	0.81(+0.00)	0.97
		RIGBD	0.81(+0.00)	0.15	0.81(+0.00)	0.21	0.82(-0.01)	0.19	0.81(+0.00)	0.22	0.81(+0.00)	0.89
	GCL	None	0.73(+0.01)	0.30	0.69(+0.05)	0.25	0.70(+0.04)	0.51	0.71(+0.03)	0.09	0.76(-0.02)	0.91
		Prune	0.75(-0.01)	0.31	0.71(+0.03)	0.24	0.71(+0.03)	0.49	0.69(+0.05)	0.13	0.76(-0.02)	0.92
		OD	0.74(+0.00)	0.30	0.71(+0.03)	0.18	0.70(+0.04)	0.44	0.69(+0.05)	0.07	0.76(-0.02)	0.91
		RIGBD	0.74(+0.00)	0.18	0.70(+0.04)	0.17	0.70(+0.04)	0.23	0.70(+0.04)	0.12	0.76(-0.02)	0.86
	GPL	None	0.18(+0.12)	0.63	0.21(+0.09)	0.51	0.26(+0.04)	0.63	0.29(+0.01)	0.46	0.34(-0.04)	0.99
		Prune	0.20(+0.10)	0.63	0.18(+0.12)	0.89	0.26(+0.04)	0.22	0.22(+0.08)	0.59	0.34(-0.04)	0.98
		OD	0.18(+0.12)	0.79	0.19(+0.11)	0.47	0.23(+0.07)	0.27	0.23(+0.07)	0.68	0.34(-0.04)	0.99
		RIGBD	0.19(+0.11)	0.19	0.20(+0.10)	0.21	0.24(+0.06)	0.21	0.23(+0.07)	0.32	0.34(+0.04)	0.91
Pubmed	GSL	None	0.86(-0.02)	0.27	0.87(-0.03)	0.79	0.86(-0.02)	0.79	0.87(-0.03)	0.66	0.84(+0.00)	0.96
		Prune	0.86(-0.02)	0.23	0.87(-0.03)	0.19	0.86(-0.02)	0.80	0.87(-0.03)	0.68	0.84(+0.00)	0.97
		OD	0.86(-0.02)	0.26	0.86(-0.02)	0.19	0.86(-0.02)	0.79	0.87(-0.03)	0.66	0.84(+0.00)	0.97
		RIGBD	0.85(-0.01)	0.17	0.86(-0.02)	0.15	0.86(-0.02)	0.29	0.87(-0.03)	0.32	0.84(+0.00)	0.88
	GCL	None	0.20(+0.64)	1.00	0.20(+0.64)	1.00	0.84(+0.00)	0.67	0.84(+0.00)	0.23	0.84(+0.00)	0.93
		Prune	0.20(+0.64)	1.00	0.20(+0.64)	1.00	0.85(-0.01)	0.64	0.84(+0.00)	0.23	0.84(+0.00)	0.93
		OD	0.20(+0.64)	1.00	0.20(+0.64)	1.00	0.84(+0.00)	0.63	0.83(+0.01)	0.20	0.84(+0.00)	0.93
		RIGBD	0.20(+0.64)	1.00	0.20(+0.64)	1.00	0.83(+0.01)	0.39	0.83(+0.01)	0.19	0.84(+0.00)	0.86
	GPL	None	0.32(+0.12)	0.58	0.39(+0.05)	0.54	0.50(-0.06)	0.65	0.45(-0.01)	0.82	0.44(-0.00)	1.00
		Prune	0.27(+0.17)	0.72	0.28(+0.16)	0.58	0.44(+0.00)	0.44	0.47(-0.03)	0.83	0.44(+0.00)	1.00
		OD	0.28(+0.16)	0.52	0.31(+0.13)	0.79	0.47(-0.03)	0.65	0.45(-0.01)	0.85	0.45(-0.01)	0.99
		RIGBD	0.28(+0.16)	0.38	0.30(+0.14)	0.48	0.47(-0.03)	0.49	0.46(-0.02)	0.54	0.46(-0.02)	0.97
Facebook	GSL	None	0.88(-0.03)	0.47	0.88(-0.03)	0.68	0.88(-0.03)	0.80	0.88(-0.03)	0.80	0.85(+0.00)	0.92
		Prune	0.87(-0.02)	0.49	0.88(-0.03)	0.13	0.88(-0.03)	0.80	0.88(-0.03)	0.80	0.85(+0.00)	0.92
		OD	0.87(-0.02)	0.38	0.88(-0.03)	0.57	0.88(-0.03)	0.80	0.88(-0.03)	0.80	0.85(+0.00)	0.92
		RIGBD	0.86(-0.01)	0.31	0.88(-0.03)	0.38	0.88(-0.03)	0.50	0.88(-0.03)	0.52	0.85(+0.00)	0.85
	GCL	None	0.82(-0.03)	0.18	0.83(00.04)	0.23	0.80(-0.01)	0.84	0.78(+0.01)	0.27	0.79(+0.00)	0.92
		Prune	0.82(-0.03)	0.17	0.83(-0.04)	0.16	0.81(-0.02)	0.95	0.78(+0.01)	0.21	0.78(+0.01)	0.93
		OD	0.80(-0.01)	0.19	0.83(-0.04)	0.18	0.84(-0.05)	0.85	0.78(+0.01)	0.24	0.79(+0.00)	0.92
		RIGBD	0.80(-0.01)	0.18	0.83(-0.04)	0.18	0.82(-0.03)	0.63	0.78(+0.01)	0.22	0.79(+0.00)	0.87
	GPL	None	0.39(-0.01)	0.01	0.31(+0.07)	0.33	0.34(+0.04)	0.30	0.33(+0.05)	0.36	0.39(-0.01)	0.99
		Prune	0.35(+0.03)	0.01	0.33(+0.05)	0.41	0.35(+0.03)	0.56	0.38(+0.00)	0.51	0.39(-0.01)	0.99
		OD	0.37(+0.01)	0.03	0.35(+0.03)	0.37	0.36(+0.02)	0.53	0.37(+0.01)	0.56	0.38(+0.00)	1.00
		RIGBD	0.36(+0.02)	0.02	0.32(+0.06)	0.22	0.34(+0.04)	0.29	0.35(+0.03)	0.42	0.38(+0.00)	0.91
OGB-arxiv	GSL	None	0.58(+0.03)	0.25	0.59(+0.02)	0.29	0.61(+0.00)	0.67	0.62(-0.01)	0.70	0.61(+0.00)	0.87
		Prune	0.59(+0.02)	0.23	0.59(+0.02)	0.24	0.61(+0.00)	0.60	0.61(+0.00)	0.58	0.60(+0.01)	0.87
		OD	0.59(+0.02)	0.21	0.59(+0.02)	0.21	0.60(+0.01)	0.53	0.61(+0.00)	0.58	0.61(+0.00)	0.86
		RIGBD	0.58(+0.03)	0.18	0.58(+0.03)	0.19	0.61(+0.00)	0.35	0.61(+0.00)	0.72	0.60(+0.01)	0.82
	GCL	None	0.50(+0.03)	0.15	0.50(+0.03)	0.19	0.52(+0.01)	0.77	0.54(-0.01)	0.75	0.53(+0.00)	0.93
		Prune	0.52(+0.01)	0.16	0.50(+0.03)	0.18	0.51(+0.02)	0.72	0.52(+0.01)	0.76	0.54(-0.01)	0.93
		OD	0.52(+0.01)	0.14	0.52(+0.01)	0.13	0.51(+0.02)	0.47	0.53(+0.00)	0.70	0.53(+0.00)	0.93
		RIGBD	0.52(+0.01)	0.14	0.51(+0.02)	0.14	0.53(+0.00)	0.43	0.54(-0.01)	0.54	0.54(-0.01)	0.97
	GPL	None	0.25(+0.03)	0.18	0.27(+0.01)	0.24	0.26(+0.02)	0.67	0.29(-0.01)	0.75	0.31(-0.03)	0.95
		Prune	0.27(+0.01)	0.14	0.28(+0.00)	0.18	0.26(+0.02)	0.65	0.27(+0.01)	0.74	0.30(-0.02)	0.94
		OD	0.28(+0.00)	0.13	0.27(+0.01)	0.22	0.27(+0.01)	0.65	0.28(+0.00)	0.70	0.30(-0.02)	0.93
		RIGBD	0.26(+0.02)	0.12	0.27(+0.01)	0.18	0.28(+0.00)	0.49	0.30(-0.02)	0.55	0.31(-0.03)	0.93

References

[Bongini *et al.*, 2021] Pietro Bongini, Monica Bianchini, and Franco Scarselli. Molecular generative graph neural networks for drug discovery. *Neurocomputing*, 450:242–252, 2021.

[Cheng *et al.*, 2020] Dawei Cheng, Sheng Xiang, Chencheng Shang, Yiyi Zhang, Fangzhou Yang, and Liqing Zhang. Spatio-temporal attention-based neural network for credit card fraud detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 362–369, 2020.

[Dai *et al.*, 2023] Enyan Dai, Minhua Lin, Xiang Zhang, and Suhang Wang. Unnoticeable backdoor attacks on graph neural networks. In *WWW*, pages 2263–2273, 2023.

[Ding *et al.*, 2025] Yuanhao Ding, Yang Liu, Yugang Ji, Weigao Wen, Qing He, and Xiang Ao. Spear: A structure-preserving manipulation method for graph backdoor attacks. In *Proceedings of the ACM on Web Conference 2025*, pages 1237–1247, 2025.

[Fan *et al.*, 2019] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.

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

[Hamilton *et al.*, 2018] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.

[Hu *et al.*, 2020] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*, volume 33, pages 22118–22133, 2020.

[Jiang *et al.*, 2019] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. Semi-supervised learning with graph learning-convolutional networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11313–11320, 2019.

[Jiang *et al.*, 2024] Bo Jiang, Hao Wu, Ziyan Zhang, Beibei Wang, and Jin Tang. A unified graph selective prompt learning for graph neural networks. *arXiv preprint arXiv:2406.10498*, 2024.

[Ju *et al.*, 2024] Wei Ju, Yifan Wang, Yifang Qin, Zhengyang Mao, Zhiping Xiao, Junyu Luo, Junwei Yang, Yiyang Gu, Dongjie Wang, Qingqing Long, Siyu Yi, Xiao Luo, and Ming Zhang. Towards graph contrastive learning: A survey and beyond, 2024.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Li *et al.*, 2024] Jiangtong Li, Dungy Liu, Dawei Cheng, and Changchun Jiang. Attack by yourself: Effective and unnoticeable multi-category graph backdoor attacks with subgraph triggers pool, 2024.

[Lin *et al.*, 2024] Minhua Lin, Zhiwei Zhang, Enyan Dai, Zongyu Wu, Yilong Wang, Xiang Zhang, and Suhang Wang. Trojan prompt attacks on graph neural networks. *arXiv preprint arXiv:2410.13974*, 2024.

[Liu *et al.*, 2023] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM web conference 2023*, pages 417–428, 2023.

[Lyu *et al.*, 2024] Xiaoting Lyu, Yufei Han, Wei Wang, Hangwei Qian, Ivor Tsang, and Xiangliang Zhang. Cross-context backdoor attacks against graph prompt learning. In *KDD*, pages 2094–2105, 2024.

[Rozemberczki *et al.*, 2021] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

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

[Sun *et al.*, 2022] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. Gppt: Graph pre-training and prompt tuning to generalize graph neural networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1717–1727, 2022.

[Sun *et al.*, 2023a] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong 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*, pages 2120–2131, 2023.

[Sun *et al.*, 2023b] Xiangguo Sun, Jiawen Zhang, Xixi Wu, Hong Cheng, Yun Xiong, and Jia Li. Graph prompt learning: A comprehensive survey and beyond, 2023.

[Velickovic *et al.*, 2017] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[Wang *et al.*, 2024] Qunzhong Wang, Xiangguo Sun, and Hong Cheng. Does graph prompt work? a data operation perspective with theoretical analysis. *arXiv preprint arXiv:2410.01635*, 2024.

[Weber *et al.*, 2019] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I Weidele, Claudio Bellei, Tom Robinson, and Charles E Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591*, 2019.

[Wu *et al.*, 2024] Yucheng Wu, Leye Wang, Xiao Han, and Han-Jia Ye. Graph contrastive learning with cohesive sub-

graph awareness. In *Proceedings of the ACM Web Conference 2024*, WWW '24, page 629–640. ACM, May 2024.

[Xi *et al.*, 2021] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. Graph backdoor. In *30th USENIX security symposium (USENIX Security 21)*, pages 1523–1540, 2021.

[Xu *et al.*, 2019] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019.

[Xu, 2020] Mengjia Xu. Understanding graph embedding methods and their applications, 2020.

[Yang *et al.*, 2025] Xiao Yang, Gaolei Li, and Jianhua Li. Graph neural backdoor: Fundamentals, methodologies, applications, and future directions, 2025.

[You *et al.*, 2020] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823, 2020.

[Yu *et al.*, 2024] Xingtong Yu, Yuan Fang, Zemin Liu, and Xinming Zhang. Hgprompt: Bridging homogeneous and heterogeneous graphs for few-shot prompt learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 16578–16586, 2024.

[Yun *et al.*, 2019] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019.

[Zhang and Chen, 2018] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[Zhang and Zitnik, 2020] Xiang Zhang and Marinka Zitnik. Gnnguard: Defending graph neural networks against adversarial attacks. *Advances in neural information processing systems*, 33:9263–9275, 2020.

[Zhang *et al.*, 2018] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[Zhang *et al.*, 2021a] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and Philip S Yu. From canonical correlation analysis to self-supervised graph neural networks. *Advances in Neural Information Processing Systems*, 34:76–89, 2021.

[Zhang *et al.*, 2021b] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. Backdoor attacks to graph neural networks. In *Proceedings of the 26th ACM symposium on access control models and technologies*, pages 15–26, 2021.

[Zhang *et al.*, 2023] Hangfan Zhang, Jinghui Chen, Lu Lin, Jinyuan Jia, and Dinghao Wu. Graph contrastive backdoor attacks. In *ICML*, pages 40888–40910. PMLR, 2023.

[Zhang *et al.*, 2024a] Zhiwei Zhang, Minhua Lin, Enyan Dai, and Suhang Wang. Rethinking graph backdoor attacks: A distribution-preserving perspective. In *KDD*, page 4386–4397, 2024.

[Zhang *et al.*, 2024b] Zhiwei Zhang, Minhua Lin, Junjie Xu, Zongyu Wu, Enyan Dai, and Suhang Wang. Robustness-inspired defense against backdoor attacks on graph neural networks. *arXiv preprint arXiv:2406.09836*, 2024.

[Zhang *et al.*, 2025] Zhiwei Zhang, Minhua Lin, Junjie Xu, Zongyu Wu, Enyan Dai, and Suhang Wang. Robustness inspired graph backdoor defense. In *The Thirteenth International Conference on Learning Representations*, 2025.

[Zhu *et al.*, 2019] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1399–1407, 2019.

[Zhu *et al.*, 2020] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020.