Malicious Node Injection: A Transferable Adversarial Attack on GNN Fairness

Haotian Zhang
Soochow University
Suzhou, China
htzhang007@stu.suda.edu.cn

He Huang*
Soochow University
Suzhou, China
huangh@suda.edu.cn

Shuang Cui
Soochow University
Suzhou, China
scui@suda.edu.cn

Yu-E Sun
Soochow University
Suzhou, China
sunyue12@suda.edu.cn

Abstract

Graph Neural Networks (GNNs) have shown remarkable effectiveness across diverse graph-related tasks. However, existing studies reveal that their fairness is highly vulnerable to adversarial manipulation, which can substantially exacerbate inherent biases toward sensitive attributes, such as gender in demographic prediction. Although prior work has shown that fairness poisoning attacks can be launched via malicious node injection, whether this strategy can support the more challenging fairness evasion attacks remains open. Moreover, existing methods have not directly addressed fairness attacks on multi-class datasets with multi-valued sensitive attributes. To bridge this gap, we propose a universal fairness attack framework (UFA) that injects malicious nodes to perform both poisoning and evasion attacks against GNNs while minimally affecting model utility. UFA first identifies nodes most susceptible to being pushed across the decision boundary. It then targets core mechanisms shared by diverse GNN architectures to manipulate node representations, thereby maximizing outcome disparities across sensitive groups. Comprehensive experiments on five real-world datasets demonstrate the effectiveness of UFA. By injecting fewer than 0.2% malicious nodes, UFA severely degrades the fairness of both mainstream and fairness-aware GNNs, proving effective in both poisoning and evasion settings. Our findings expose the fragility of fairness in GNNs and underscore the urgent need for robust fairness-aware models.

1 Introduction

Benefiting from their strong capacity to encode graph-structured representations, GNNs efficiently process non-Euclidean graph data [1] and have achieved notable success in recommender systems [2], molecular simulation [29], and social networks [43]. However, recent studies [44] show that GNNs can produce biased predictions for nodes from certain demographic groups (e.g., gender or race). As GNNs are increasingly deployed in high-stakes, human-centered applications, mitigating such bias is imperative. Accordingly, substantial effort has been devoted to fairness-aware GNNs that learn equitable node representations for accurate and fair predictions [3, 36, 40]. Despite these advances, their fairness guarantees remain vulnerable to attack [22, 41, 12]. Adversarial fairness attacks perturb input graph data to induce unfair outcomes in a victim model, potentially compromising the reliability of GNNs, even those with built-in fairness-enhancing mechanisms. Therefore, protecting GNNs against such threats requires first thoroughly understanding potential attack avenues.

*Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

A growing body of work has investigated fairness attacks on GNNs, which aim to craft poisoned data that induces biased predictions across groups defined by sensitive attributes while minimally affecting overall utility (e.g., classification accuracy) to evade detection. Based on timing, these attacks are broadly divided into poisoning attacks during training and evasion attacks at inference. For example, G-FairAttack [41] performs both fairness poisoning and evasion attacks by modifying edges with limited utility degradation. However, as noted in [22], it assumes attackers can alter connectivity among existing nodes, which is often unrealistic under strict real-world access constraints. To overcome this limitation, NIFA was proposed to inject malicious nodes, enabling fairness poisoning attacks without modifying existing nodes. Nevertheless, its capability is restricted to fairness poisoning. Fairness evasion attacks are more challenging because they must operate in real time across architectures under realistic limited-permission settings [27]. Moreover, existing methods have not directly implemented fairness attacks on multi-class datasets with multi-valued sensitive attributes. This leaves significant open questions:

Q1: *Can fairness evasion attacks on GNNs be achieved through a practical method (such as malicious node injection) without manipulating the existing graph?*

Q2: *If so, can the effectiveness of such an attack match or even surpass that of existing methods manipulating the existing graph (which are often impractical due to their high permission)?*

Q3: *Can fairness attacks be achieved on multi-class datasets with multi-valued sensitive attributes?*

1.1 Our Contributions

In this paper, we provide affirmative answers to the above open question by proposing a novel and practical fairness attack method. Our main contributions can be summarized as follows:

- We propose the Universal Fairness Attack (UFA), an adversarial, gray-box framework that operates via malicious node injection. To the best of our knowledge, UFA is the first method to achieve fairness evasion attacks on GNNs without manipulating the existing graph.
- To the best of our knowledge, UFA is the first to achieve fairness attacks on multi-class datasets with multi-valued sensitive attributes, without requiring any modification to the algorithmic structure. UFA can simultaneously carry out both fairness poisoning and fairness evasion attacks, and substantially outperforms existing methods in both attack settings.
- We design objective function based on shared feature saliency maps in GNNs. Compared with existing methods, it achieves stronger attack performance with better interpretability.
- We conduct extensive experiments on five widely used real-world datasets, directly comparing UFA with SOTA methods under both poisoning and evasion settings. The results show that UFA induces stronger fairness bias in GNNs while causing negligible utility loss.

1.2 Technical Overview

To achieve these results, UFA incorporates several key designs. First, unlike prior work that alters existing node attributes or edge connections [14, 33, 41, 13], UFA injects malicious nodes into the global graph. This is more practical in real-world systems such as social networks, where attackers usually cannot edit other users' data or relationships but can create new accounts or connections. Although NIFA also uses node injection, its methodology differs fundamentally. For vulnerable target-node selection, NIFA samples model outputs and computes their variance, mainly capturing model instability or structural sensitivity rather than boundary proximity. In contrast, UFA directly quantifies proximity to the decision boundary in the prediction space using information entropy and class margin, enabling more accurate vulnerable-node selection. More importantly, while NIFA is limited to fairness poisoning attacks, UFA supports both fairness poisoning and the more challenging fairness evasion attacks.

Moreover, compared with existing methods such as NIFA, UFA leverages shared fairness saliency in GNNs to maximally amplify inter-group disparities, providing more intuitive attack interpretability. This attention-based attack is conceptually similar to attention map attacks in CNNs, which are mainly used for image data [35, 42] and remain under-explored in GNN fairness attacks. However, conventional CNN feature-visualization attention maps are inapplicable to GNN fairness attacks [26] due to two key distinctions: (1) graph data involve structural dependencies and complex topological

relationships absent in grid-based images; and (2) fairness attacks involve high-dimensional continuous optimization rather than class-discriminative tasks. To address this, UFA introduces a fairness saliency map attack that exploits the universality of GNN message passing by targeting shared vulnerability regions with similar fairness saliency distributions across architectures on identical inputs. By using precise gradient information, the saliency map enables sensitivity-driven perturbations, allowing malicious node features optimized on the shadow model to generalize effectively to other architectures. This provides strong interpretability, more accurate perturbations, faster convergence, and a unified framework for fairness poisoning and evasion attacks.

UFA also designs an objective function that can attack multi-class datasets with multi-valued sensitive attributes without requiring algorithmic modifications for different dataset properties. Overall, by maximizing disparities in fairness saliency map responses across sensitive groups, UFA identifies the optimal directions and magnitudes of feature perturbations that most effectively undermine fairness.

2 Related Work

2.1 Fairness-aware GNNs

Researchers have found that machine learning models often suffer from fairness issues [24]. [25] first studied fairness in graph data by proposing FairWalk, which mitigates bias before training via graph-structure modification or random-walk strategies. Later, [3] found that GNNs may discriminate against sensitive attributes due to historical biases and message passing, and proposed FairGNN, which uses adversarial training with a sensitive-attribute discriminator to suppress bias. FairVGNN [36] masks features highly correlated with sensitive attributes to prevent bias propagation, while FairNorm [17] introduces a learnable group-based normalization operator to reduce intra-group disparities. FairSIN [40] adopts a neutralization-based paradigm that enhances node representations before message passing with additional non-sensitive information. Unlike these group-fairness methods, GFairHint [39] learns a fairness vector via auxiliary link prediction and concatenates it with node embeddings to provide individual fairness guarantees.

2.2 Adversarial Attack of GNNs

Due to their intuitive nature, early adversarial attacks mainly focused on image data [31]. [45] first introduced adversarial attacks to graphs with Nettack, a gray-box greedy attack. Subsequently, [4] proposed RL-S2V, a reinforcement-learning-based black-box attack that degrades GNN utility. [12] first explicitly targeted group fairness in node classification through adversarial edge injection, increasing fairness disparity while preserving accuracy. [41] proposed G-FairAttack, which uses a general proxy loss and non-gradient constrained optimization for both poisoning and evasion attacks. However, it requires full-graph traversal and surrogate-model retraining in each iteration, leading to high overhead and poor scalability on large graphs; it also perturbs existing node connections, which may be infeasible under restricted permissions. NIFA [22], the SOTA method, shows that injecting adversarial nodes can stealthily disrupt GNN group fairness without modifying the original graph. However, NIFA is limited to poisoning attacks, requires more model knowledge, and cannot perform evasion attacks under stricter constraints. Existing methods also have not directly addressed fairness attacks on multi-class datasets with multi-valued sensitive attributes.

3 Preliminaries

3.1 Notation

We consider a typical node classification task on graph data. The graph is defined as $G = (V, E, X, Y)$, where V is the set of nodes, E is the set of edges, with $|V| = n$ and $|E| = m$. $X \in \mathbb{R}^{n \times d}$ denotes the set of node feature matrices, where d is the dimensionality of each node's feature vector. $Y \in \mathbb{R}^n$ represents the ground-truth label vector for all nodes. The adjacency matrix of the graph is given by $A \in \{0, 1\}^{n \times n}$, where for any node pair $\{u, v\} \in V, A_{uv} = 1$ if $e_{uv} \in E$, and $A_{uv} = 0$ otherwise. Let $\mathcal{M}(\theta; \cdot)$ denote a GNN model parameterized by trainable parameters θ. The goal of the node classification task is to learn a mapping $\mathcal{M}(\theta; A, V) \rightarrow \tilde{Y}$, where $\tilde{Y} \in \mathbb{R}^{n \times |\mathcal{Y}|}$

is the predicted probability distribution over labels for each node. A well-trained model $\mathcal{M}(\theta;\cdot)$ aims to assign each node a predicted label that matches its ground-truth label as accurately as possible.

3.2 Fairness Metrics

Fairness metrics quantify the extent to which a model's predictions are equitable. Following prior work [20, 3, 41, 22], we first focus on group fairness under a binary sensitive attribute, although UFA can be generalized to multi-sensitive settings. Let each node $v \in V$ in G have a sensitive attribute $S \in \{0,1\}$. Denote by $\tilde{Y} \in \{0,1\}$ the binary prediction of the classifier, and by $Y \in \{0,1\}$ the ground-truth label. We consider two standard group fairness notions:

- **Equal Opportunity** [9] requires equal true positive rates across sensitive groups:

$$P(\tilde{Y}=1 \mid S=1, Y=1) = P(\tilde{Y}=1 \mid S=0, Y=1)$$

- **Demographic Parity** [6] requires that the overall positive prediction rates be identical across sensitive groups:

$$P(\tilde{Y}=1 \mid S=1) = P(\tilde{Y}=1 \mid S=0)$$

We further generalize these binary group fairness notions to the multi-class and multi-sensitive-group setting by computing pairwise disparities across sensitive groups for each class and averaging them over all classes. The DP loss is defined as the average pairwise disparity of $P(\tilde{Y}=i \mid S=s)$ across sensitive groups, and the EO loss is defined as the average pairwise disparity of $P(\tilde{Y}=i \mid S=s, Y=i)$ across sensitive groups:

$$\Delta_{DP}^{(i)} = \frac{1}{|\mathcal{B}_i|} \sum_{(s,s')\in\mathcal{B}_i} \left| P(\tilde{Y}=i \mid S=s) - P(\tilde{Y}=i \mid S=s') \right|,$$

$$\Delta_{EO}^{(i)} = \frac{1}{|\mathcal{C}_i|} \sum_{(s,s')\in\mathcal{C}_i} \left| P(\tilde{Y}=i \mid S=s, Y=i) - P(\tilde{Y}=i \mid S=s', Y=i) \right|.$$

Here, s and s' denote two distinct values of the sensitive attribute, $\mathcal{B}_i$ and $\mathcal{C}_i$ denote the sets of valid sensitive-group pairs for class i. We then aggregate the class-wise disparities by averaging them over all classes. A lower fairness loss indicates better fairness of GNNs:

$$\Delta_{DP} = \frac{1}{|\mathcal{Y}|} \sum_{i\in\mathcal{Y}} \Delta_{DP}^{(i)}, \quad \Delta_{EO} = \frac{1}{|\mathcal{Y}|} \sum_{i\in\mathcal{Y}} \Delta_{EO}^{(i)}.$$

3.3 Problem Formulation

In fairness attacks on GNNs, attackers aim to induce bias in the victim GNNs' predictions by slightly perturbing the input graph data within certain feasible spaces. Given the graph G, UFA injects a set of malicious nodes $V_{adv} = V \cup V'$, $X_{adv} = X \cup X'$, $X' \in [X_{min}, X_{max}]^{b\times d}$, and $|V'| = b$, where the feature X' of malicious nodes V' are optimized by algorithm and constrained within ϵ. b is the attack budget, representing the maximum number of malicious nodes that can be injected. UFA does not tamper with the label set Y of train set V^{train} and test set V^{test}. Let $\mathcal{L}$ be a training loss, following prior studies [41, 22], we define the fairness poisoning and evasion attack as:

Problem 1. Fairness Poisoning Attack: The attacker generates the poisoned data G_{adv} and its adjacency matrix A_{adv}. During training, the model trains on G_{adv} and learns the poisoned parameters θ_{adv}. During the inference phase, using the test nodes set V^{test}, the attacker aims to maximize the fairness loss $\mathcal{L}_{\Delta_{EO},\Delta_{DP}}$, while constraining the cross-entropy loss on test nodes to remain within a small tolerance ϕ compared to that of the clean model:

$$\theta_{adv} = \arg\min_{\theta^*} \mathcal{L}(\mathcal{M}(\theta^*; A_{adv}, V^{train}), Y), \quad \max_{G_{adv}} \mathcal{L}_{\Delta_{EO},\Delta_{DP}}(\mathcal{M}(\theta_{adv}; A_{adv}, V^{test}), Y, S)$$
$$s.t.\ \mathcal{L}_{CE}(\mathcal{M}(\theta_{adv}; A_{adv}, V^{test})) < \mathcal{L}_{CE}(\theta; A, V^{test}) + \phi$$

Problem 2. Fairness Evasion Attack: In the training phase, the victim model $\mathcal{M}(\theta;\cdot)$ is trained on the original data G with adjacency matrix A, yielding clean model parameters θ. During the inference

phase, the attacker uses poisoned data G_{adv} with adjacency matrix A_{adv} and aims to maximize the fairness loss $\mathcal{L}_{\Delta_{EO},\Delta_{DP}}$, while preserve model utility:

$$\theta = \arg\min_{\theta^*} \mathcal{L}(\mathcal{M}(\theta^*; A, V^{train}), Y), \quad \max_{G_{adv}} \mathcal{L}_{\Delta_{EO},\Delta_{DP}}(\mathcal{M}(\theta; A_{adv}, V^{test}), Y, S)$$
$$s.t.\ \mathcal{L}_{CE}(\theta; A_{adv}, V^{test}) < \mathcal{L}_{CE}(\theta; A, V^{test}) + \phi$$

Notably, UFA can be simultaneously applied to both fairness poisoning attacks and fairness evasion attacks. The poisoned graph G_{adv} generated in the two attack settings is identical throughout this paper. The details of the attack settings will be shown in Appendix A.

4 Methodology

4.1 Overview of UFA

The UFA framework, illustrated in Figure 1, comprises two stages. (A) In the injection stage, UFA leverages DropConnect and repeated sampling to train a GNN and identify nodes near the decision boundary as targets for injecting a small number ($< 0.2\%$) of optimizable malicious nodes. (B) In the feature construction stage, UFA employs a bi-level optimization: the first level trains a shadow model, while the second level updates the injected nodes' features by exploiting shared saliency map across GNNs, thereby amplifying inter-group disparities and degrading fairness.

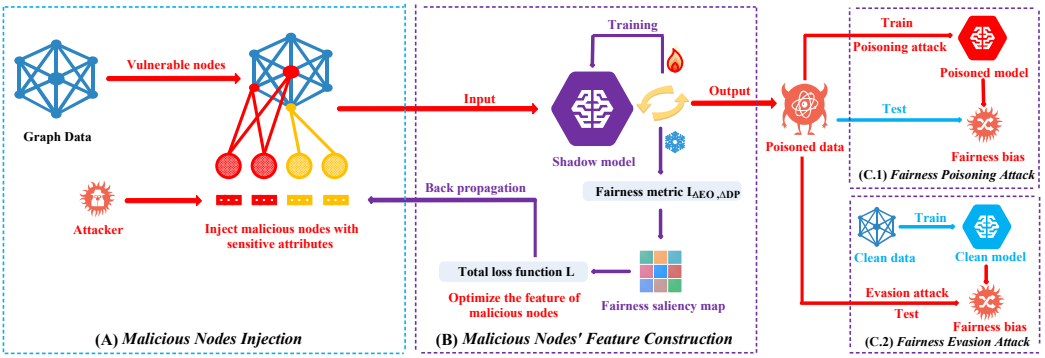


Figure 1: Framework of the UFA Algorithm

4.2 Malicious Nodes Injection

In this phase, UFA identifies the subset of nodes most susceptible to being driven toward the decision boundary and uses them as targets for injecting optimizable malicious nodes. The number of malicious nodes is constrained to less than 0.2% of the total node count. We first train a GCN model with the DropConnect [10] mechanism $\mathcal{M}_D$ and perform multiple forward passes during inference to obtain a simple variational Bayesian approximation of the model parameters θ_D [23]. This enables estimation of the mean predictive probability distribution for each node, improving prediction stability and reducing variance induced by stochastic weight dropping. The effects of different $\mathcal{M}_D$ backbone models are reported in Appendix J.2. The training loss is defined as follows:

$$\mathcal{L}_{\theta_D} = \mathcal{L}_{CE}(\mathcal{M}_D(\theta; G)) + ||\theta||_2^2$$

$\mathcal{L}_{CE}$ represents the cross-entropy loss, and $||\cdot||_2^2$ is the ℓ_2-norm regularization term. $\mathcal{M}_D$ is trained on the original graph G, where DropConnect is activated during sampling to generate sub-networks and produce an average predictive probability distribution $\mathcal{P}_\mathcal{D}$ for each labeled node:

$$\mathcal{P}_\mathcal{D} = \frac{1}{T_s}\sum_{1}^{T_s} \text{softmax}(\mathcal{M}_D(\theta_D; \rho, G))$$

where T_s denotes the sampling iterations, and ρ is the DropConnect rate, which sets a proportion of the weight matrix connections (i.e., parameter values θ_D) to 0 with probability ρ. This process

enables Monte Carlo sampling of the probability distribution over the labeled nodes. Then, UFA leverages $p \in \mathcal{P}_{\mathcal{D}}$ to identify the set of nodes most susceptible to being driven toward the decision boundary. Specifically, the algorithm employs two standard classification uncertainty measures: **entropy** and **class margin**. It computes their combination to assess a node's relative position to decision boundary in the prediction space, thereby yielding uncertainty score $\mathcal{Q}$ for node:

$$\mathcal{Q} = (1 - (p^{(1)} - p^{(2)}) - \sum\nolimits_c p^{(c)} \log p^{(c)})_{nor}$$

where the first term is the class margin, measuring the node's local distance to the decision boundary. The second term is the node's entropy, which captures the global flatness of its predictive distribution. Specifically, $p^{(c)}$ denotes the probability assigned to class c (used to compute entropy over all classes), $p^{(1)}$ is the probability of the node's most likely class, $p^{(2)}$ is the probability of its second-most likely class, and $(\cdot)_{nor}$ is min-max normalization. We compare the effects of different attack-node selection strategies in Appendix J.1. A detailed procedure is shown in Algorithm 2 in Appendix C.

4.3 Malicious Nodes' Feature Construction

In this phase, UFA aims to construct features X' for malicious nodes V' via a shadow model, such that the poisoned graph induces fairness bias across sensitive-attribute groups in the victim GNN. Since UFA attacks the shared fairness-saliency map of GNNs, the resulting poisoned data can generalize across different GNN architectures, enabling both fairness poisoning and fairness evasion attacks. Specifically, UFA introduces a fairness saliency map attack that targets the message-passing universality of GNNs, i.e., the common vulnerability regions reflected by similar fairness saliency map across different GNN variants on identical inputs. Let $\mathcal{S}$ denote the set of sensitive groups and $\mathcal{Y}$ denote the set of class labels. For node v, let z_v denote the prediction vector produced by the shadow GNN. We use $\mathcal{L}_{\Delta_{EO},\Delta_{DP}} = \mathcal{L}_{\Delta_{EO}} + \mathcal{L}_{\Delta_{DP}}$ to measure the fairness group disparity of G:

$$\mathcal{L}_{\Delta_{DP}} = \frac{1}{|\mathcal{S}|d} \sum_{s \in \mathcal{S}} \left\| \frac{1}{|V_s|} \sum_{v \in V_s} z_v - \frac{1}{|\mathcal{S}|} \sum_{s' \in \mathcal{S}} \frac{1}{|V_{s'}|} \sum_{v \in V_{s'}} z_v \right\|_2^2,$$

$$\mathcal{L}_{\Delta_{EO}} = \frac{1}{|\mathcal{Y}|} \sum_{c \in \mathcal{Y}} \frac{1}{|\mathcal{S}_c|d} \sum_{s \in \mathcal{S}_c} \left\| \frac{1}{|V_s^{(c)}|} \sum_{v \in V_s^{(c)}} z_v - \frac{1}{|\mathcal{S}_c|} \sum_{s' \in \mathcal{S}_c} \frac{1}{|V_{s'}^{(c)}|} \sum_{v \in V_{s'}^{(c)}} z_v \right\|_2^2$$

where V_s denotes the set of nodes whose sensitive attribute is s, $V_s^{(c)}$ denotes the set of nodes whose sensitive attribute is s and label is c, $\mathcal{S}$ denotes the set of sensitive groups, $\mathcal{S}_c$ denotes the set of sensitive groups that contain at least one node with label c, and d is the dimension of the prediction vector z_v. Based on this fairness loss, the fairness saliency map for sensitive group s is defined as:

$$\mathcal{H}_s = \frac{1}{|V_s|} \sum_{v \in V_s} \frac{\partial \mathcal{L}_{\Delta_{EO},\Delta_{DP}}}{\partial X_v}, \quad s \in \mathcal{S}$$

Here, X_v denotes the input feature vector of node v. The fairness saliency map $\mathcal{H}_s$ captures how input features of nodes in sensitive group s influence the model's fairness-oriented loss.

Lemma 4.1. *For the same graph input, if the objective functions of two models exhibit similar local behavior within a neighborhood of the input and are differentiable with respect to the input features, then their input-gradient-based saliency maps are close.*

The proof of **lemma 4.1** is provided in the Appendix B. It provides a sufficient condition: when two GNNs exhibit similar local responses around the same graph input under the fairness objective, their input-gradient-based fairness saliency maps are close. Therefore, the lemma formalizes why attacking saliency maps can transfer across architectures once local response similarity holds. We empirically verify this condition in Appendix G. After obtaining the fairness saliency maps, UFA maximizes the dispersion among the saliency maps of different sensitive groups to induce biased outcomes in the model's predictions. UFA does not explicitly construct the Hessian and it computes the derivatives through automatic differentiation. The fairness attack loss is defined as

$$\mathcal{L}_f = -\frac{1}{|\mathcal{S}|R} \sum_{s \in \mathcal{S}} \left\| \mathcal{H}_s - \bar{\mathcal{H}} \right\|_2^2, \quad \bar{\mathcal{H}} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \mathcal{H}_s,$$

where $\mathcal{S}$ denotes the set of sensitive groups, R is the dimension of $\mathcal{H}$, and $\mathcal{L}_f$ denotes the fairness attack loss. Minimizing $\mathcal{L}_f$ is equivalent to maximizing the dispersion of fairness saliency maps across sensitive groups, thereby exacerbating the model's unfairness.

To ensure that the poisoning attack does not significantly compromise the victim model's utility performance, which is essential for maintaining attack stealthiness, we introduce a utility preservation loss. In practice, we use the standard cross-entropy loss on labeled training nodes: $\mathcal{L}_U = -\frac{1}{|V_{\text{tr}}|}\sum_{v\in V_{\text{tr}}}\sum_{c\in\mathcal{Y}} y_v^{(c)} \log z_v^{(c)}$, where V_{tr} denotes the set of labeled training nodes, $y_v^{(c)}$ is the one-hot ground-truth label of node v, and $z_v^{(c)}$ denotes the predicted probability of the GNN for class c. The utility constraint defined in the problem formulation is implemented through a penalty-based approach, whereby $\mathcal{L}_U$ is incorporated into the optimization objective for the malicious node features. Therefore, in the current implementation, ϕ is not treated as a separate, predefined, dataset-specific hyperparameter. Subsequently, we encourage the injected malicious nodes to form feature-divergent subgroups, thereby reinforcing group-wise polarization in the feature space:

$$\mathcal{L}_{CF} = \frac{1}{|\mathcal{S}'|d}\sum_{s\in\mathcal{S}'}\left\|\frac{1}{|V_s'|}\sum_{v\in V_s'} X_v' - \frac{1}{|\mathcal{S}'|}\sum_{s'\in\mathcal{S}'}\frac{1}{|V_{s'}'|}\sum_{v\in V_{s'}'} X_v'\right\|_2^2,$$

V_s' denotes the injected-node subset assigned to the candidate pool of sensitive group $s \in \mathcal{S}$, d is the feature dimension. Finally, the total objective for optimizing the malicious node features is formulated as $\mathcal{L} = \alpha \cdot \mathcal{L}_f - \mathcal{L}_{CF} + \mathcal{L}_U$, where α is a hyperparameter.

4.4 Put Them Together

In this section, we design an algorithm to effectively optimize $\mathcal{L}$ for constructing malicious node features X'. Since the attack is conducted in a gray-box setting, the attacker has no access to the victim model's internal information. To address this, we propose a bi-level multi-step optimization framework, where training a shadow model $\mathcal{M}_q(\theta_q;\cdot)$ constitutes the first-level optimization problem, and optimizing malicious node features forms the second-level problem. By attacking the shared saliency patterns of GNNs to maximize disparity across sensitive groups, the malicious node features optimized by the shadow model can transfer to GNN models with different architectures.

Algorithm 1 Universal Fairness attack (UFA)

Input: Graph $G = (V, E, X, Y)$;
Parameter: Shadow model $\mathcal{M}_q(\theta_q;\cdot)$; Loss function of shadow GNN and feature construction $\mathcal{L}_q$ and $\mathcal{L}$; Train epoch of shadow model and feature construction T_q and T_f; Epoch of bi-level iteration T_e; Learning rate of shadow model and feature construction ℓ_q and ℓ_f;
Output: Poisoned graph G_{adv};

1: Initializing the parameter θ_q of the model $\mathcal{M}_q$;
2: Select the target node set $\mathcal{J}$ of G by **Algorithm 2** and connect them to randomly chosen target nodes to get unoptimized G_{adv}^*. Malicious nodes don't participate in the evaluation.
3: **for** 0, 1, ..., T_e **do**
4: **for** 0, 1, ..., T_q **do**
5: $\theta_q \leftarrow \theta_q - \ell_q \cdot \nabla_{\theta_q}\mathcal{L}_q(G_{adv}^*)$;
6: **end for**
7: **for** 0, 1, ..., T_f **do**
8: $\mathcal{Z} = \text{softmax}(\mathcal{M}_q(\theta_q; G_{adv}^*))$, get probability distribution vector over the labeled nodes;
9: $X' \leftarrow X' - \ell_f \cdot \nabla_{X'}\mathcal{L}(\mathcal{Z})$
10: **end for**
11: Clamp X' between the min and max of X
12: **end for**
13: **return** G_{adv}

As shown in Algorithm 1, line 2 injects malicious nodes into G and generates the poisoned graph to be optimized G_{adv}^*, where the optimization target is the feature matrix of the malicious nodes $X' \in \mathbb{R}^{b\times d}$. The epoch of iterations in the bi-layer iteration loop is T_e (line 3). Lines 4–6 constitute the first-level, which trains a shadow model $\mathcal{M}_q(\theta_q;\cdot)$ and produces prediction probability vectors $\mathcal{Z}$. Lines 7–10 define the second-level optimization loop, where feature $X' \in \mathbb{R}$ is iteratively optimized.

To prevent malicious nodes from being filtered out by anomaly detection, $X' \in \mathbb{R}$ are constrained within the range of clean features X (line 11).

5 Experiments

We evaluate UFA against SOTA methods on real-world datasets of varying scales, assessing its effectiveness in both fairness poisoning and evasion attacks and its robustness across diverse GNN architectures, including representative fairness-aware GNNs. The Appendix provides further ablation studies, computational efficiency analysis, structural impact evaluation, and potential defense strategies, further validating the superiority and practicality of UFA over existing approaches. The code is available at: `https://github.com/pikalooo/ufa`

Table 1: Attack performance of our method and SOTA baselines on fairness poisoning attacks against GNNs across various datasets. The results are reported in percentage (%), where timeout error is indicated by -. The best results are highlighted in bold.

Victim	Attack	Pokec-Z			Pokec-N			DBLP		
		ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
GCN	Clean	71.30	4.61	5.97	71.11	2.13	1.37	95.98	1.45	3.33
	FA-GNN	71.03	5.29	6.38	71.41	3.31	2.06	94.61	4.49	6.11
	G-FairAttack	-	-	-	-	-	-	95.48	5.48	4.41
	FATE	-	-	-	-	-	-	95.83	4.81	3.23
	NIFA	70.72	16.23	18.10	70.20	9.11	9.29	93.04	19.49	13.07
	UFA	71.25	**16.87**	**18.94**	70.29	**10.82**	**10.66**	93.47	**20.71**	**14.34**
GraphSAGE	Clean	70.63	3.11	5.19	69.01	2.17	1.32	96.31	3.42	3.79
	FA-GNN	71.41	3.94	4.95	70.78	2.09	1.44	94.78	5.50	4.34
	G-FairAttack	-	-	-	-	-	-	96.48	2.25	3.48
	FATE	-	-	-	-	-	-	95.51	3.81	4.49
	NIFA	70.32	3.91	5.28	68.85	2.05	1.53	93.34	16.22	9.80
	UFA	69.38	**4.19**	**5.89**	69.02	**3.74**	**2.33**	92.60	**17.47**	**11.08**
APPNP	Clean	69.84	5.00	6.86	68.40	5.22	4.91	96.47	2.23	3.86
	FA-GNN	69.17	6.30	7.31	68.97	7.08	6.79	95.48	5.48	4.41
	G-FairAttack	-	-	-	-	-	-	93.15	6.51	7.67
	FATE	-	-	-	-	-	-	95.57	5.38	4.71
	NIFA	69.36	16.60	18.11	68.15	11.81	12.05	91.08	**24.20**	16.90
	UFA	69.19	**21.08**	**22.48**	67.13	**14.63**	**14.71**	91.20	23.85	**17.07**
SGC	Clean	67.43	4.18	4.78	67.17	3.39	2.92	96.71	2.32	3.67
	FA-GNN	67.24	5.07	7.27	65.95	5.18	5.59	93.43	3.65	4.29
	G-FairAttack	-	-	-	-	-	-	95.95	2.34	3.41
	FATE	-	-	-	-	-	-	95.90	4.14	5.45
	NIFA	66.57	16.51	17.82	65.40	11.40	12.54	91.38	22.36	12.32
	UFA	66.50	**20.02**	**21.32**	65.72	**12.30**	**12.63**	91.04	**23.04**	**15.23**
GAT	Clean	68.33	4.72	3.48	69.10	2.35	2.59	93.89	6.55	2.26
	FA-GNN	68.41	3.60	3.88	67.87	2.55	2.73	93.72	6.70	1.88
	G-FairAttack	-	-	-	-	-	-	91.81	6.06	5.44
	FATE	-	-	-	-	-	-	92.30	6.89	3.03
	NIFA	68.46	8.35	**10.19**	67.69	3.30	2.99	92.92	7.96	3.35
	UFA	68.25	**8.48**	8.95	68.85	**4.60**	**4.95**	92.96	**8.06**	**4.30**
GIN	Clean	68.71	1.32	1.69	69.15	3.03	4.27	96.58	2.69	3.45
	FA-GNN	68.53	2.80	3.12	67.33	3.51	5.52	93.09	5.25	3.59
	G-FairAttack	-	-	-	-	-	-	94.09	10.52	8.37
	FATE	-	-	-	-	-	-	93.88	9.94	8.25
	NIFA	68.37	2.82	3.35	68.02	5.37	8.16	93.12	15.96	9.85
	UFA	68.24	**5.78**	**5.45**	68.13	**6.75**	**8.53**	93.84	**18.47**	**12.87**
FairGNN	Clean	69.63	2.76	2.04	68.57	1.63	2.01	89.05	5.46	7.78
	FA-GNN	69.11	3.37	4.13	68.01	2.32	3.17	87.43	7.91	8.22
	G-FairAttack	-	-	-	-	-	-	87.09	6.89	5.90
	FATE	-	-	-	-	-	-	88.52	7.55	5.90
	NIFA	68.92	**4.20**	**6.24**	68.34	4.32	**4.07**	90.35	19.13	12.18
	UFA	69.42	3.90	5.49	68.27	**4.33**	4.05	90.55	**19.17**	**12.94**
FairVGNN	Clean	68.39	2.02	3.56	70.87	2.20	1.83	88.84	6.22	8.03
	FA-GNN	68.26	3.92	4.39	68.96	2.40	1.94	90.68	8.25	7.95
	G-FairAttack	-	-	-	-	-	-	86.93	8.90	6.82
	FATE	-	-	-	-	-	-	87.52	7.66	6.52
	NIFA	70.78	6.23	8.18	69.66	3.12	2.41	90.70	**18.34**	11.65
	UFA	71.67	**8.98**	**10.89**	69.68	**7.20**	**6.83**	90.16	16.68	**11.90**
FairSIN	Clean	70.93	1.34	2.90	69.86	0.18	0.46	93.72	7.97	4.37
	FA-GNN	71.53	1.40	2.05	70.64	1.19	1.11	89.84	9.72	7.02
	G-FairAttack	-	-	-	-	-	-	87.94	8.75	7.94
	FATE	-	-	-	-	-	-	88.01	6.55	5.59
	NIFA	70.03	0.98	2.44	67.50	1.21	1.51	90.45	12.72	11.34
	UFA	71.20	**3.09**	**4.22**	70.05	**2.81**	**3.13**	93.72	**15.54**	**16.88**

5.1 Experimental Settings

Datasets: Since existing studies primarily focus on binary classification with binary sensitive groups, we strictly follow the dataset settings used in prior work [3, 12, 22] and conduct experiments on three real-world datasets: DBLP [3], Pokec-Z, and Pokec-N [32]. We further evaluate UFA on

AMiner-S [5] and AMiner-L [5], corresponding to binary classification with multi-valued sensitive groups and multi-class classification with multi-valued sensitive groups, respectively. In each dataset, only a subset of nodes is labeled; these labeled nodes are randomly split into 50% for training, 25% for validation, and 25% for testing. Dataset statistics are summarized in Appendix H.

Baselines and Victim Models: We compare our algorithm against SOTA fairness attack method, including **FA-GNN** [12], **G-FairAttack** [41], **FATE** [13], **NIFA** [22]. We faithfully follow the original settings as specified in the papers. Further details are provided in Appendix D. We compare the algorithms on different GNNs as victim models, including classical GNNs (**GCN** [16], **GraphSAGE** [8], **APPNP** [7], **SGC** [37], **GAT** [34], **GIN** [38]) and fairness-aware GNNs (**FairGNN** [3], **FairVGNN** [36], **FairSIN** [40]). The details of these GNNs are shown in Appendix E.

Evaluation metrics and parameter settings: We adopt Δ_{EO} and Δ_{DP} introduced in Section 3.2 as fairness evaluation measures; larger values correspond to greater fairness bias. As the utility-loss metric, we use the overall classification accuracy Acc [41, 22]; a higher drop in Acc indicates a stronger perturbation to model utility and a more readily detectable attack. All experiments are independently repeated 20 times, and the performance improvements are statistically significant according to a t-test ($p < 0.05$). Further details on hyperparameters are provided in Appendix H.

Table 2: Attack performance of our method and SOTA baselines on fairness evasion attacks against GNNs across various datasets. The results are reported in percentage (%), where timeout error is indicated by -. The best results are highlighted in bold.

Victim	Attack	Pokec-Z			Pokec-N			DBLP		
		ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
GCN	Clean	71.31	5.12	7.08	70.77	2.36	1.55	96.12	1.81	3.28
	G-FairAttack	-	-	-	-	-	-	94.17	6.28	9.20
	FA-GNN	71.09	10.33	12.03	69.46	11.43	11.05	91.26	13.86	10.61
	UFA	70.62	**22.91**	**24.53**	69.80	**18.93**	**18.55**	89.52	**27.64**	**20.02**
GraphSAGE	Clean	70.56	2.95	3.59	68.92	2.98	1.77	96.34	2.46	3.14
	G-FairAttack	-	-	-	-	-	-	88.34	**23.32**	**27.35**
	FA-GNN	69.62	10.30	11.96	67.21	10.68	10.37	87.84	13.20	11.95
	UFA	69.24	**17.60**	**19.20**	67.69	**22.52**	**22.18**	88.33	20.28	27.27
APPNP	Clean	69.38	5.43	7.20	68.66	5.22	4.915	96.84	1.69	3.23
	G-FairAttack	-	-	-	-	-	-	90.30	19.83	22.15
	FA-GNN	69.31	10.04	11.53	67.25	12.62	12.31	86.48	12.99	12.38
	UFA	68.79	**27.97**	**29.01**	66.81	**27.03**	**26.76**	88.97	**30.41**	**23.88**
SGC	Clean	68.04	4.22	6.03	65.95	4.01	3.50	96.66	2.76	4.34
	G-FairAttack	-	-	-	-	-	-	90.45	21.21	23.27
	FA-GNN	66.85	14.21	15.42	65.14	15.20	14.90	86.63	15.33	18.95
	UFA	66.91	**26.19**	**27.11**	65.79	**27.69**	**27.42**	88.18	**23.24**	**29.66**
GAT	Clean	68.90	4.79	6.04	68.60	1.48	1.96	96.23	3.14	2.80
	G-FairAttack	-	-	-	-	-	-	91.23	12.75	12.48
	FA-GNN	69.88	6.99	8.61	68.70	4.15	5.71	94.15	9.30	6.32
	UFA	68.10	**32.72**	**33.77**	65.83	**27.75**	**27.56**	90.95	**21.98**	**15.51**
GIN	Clean	69.13	2.99	4.31	68.10	2.97	2.58	94.75	5.93	2.75
	G-FairAttack	-	-	-	-	-	-	92.07	11.38	16.02
	FA-GNN	68.98	6.03	6.65	68.92	5.46	4.71	96.38	6.54	5.38
	UFA	67.90	**17.50**	**19.10**	67.39	**10.56**	**10.90**	91.98	**15.93**	**21.21**
FairGNN	Clean	68.78	3.36	1.72	68.34	1.91	1.52	88.44	5.38	6.94
	G-FairAttack	-	-	-	-	-	-	86.94	7.14	9.29
	FA-GNN	67.61	15.13	16.67	66.95	18.82	18.54	86.41	7.94	6.19
	UFA	67.03	**29.55**	**30.6**	68.05	**29.68**	**29.52**	85.93	**20.66**	**17.20**
FairVGNN	Clean	68.55	2.11	3.32	71.59	2.27	1.38	88.94	5.63	8.58
	G-FairAttack	-	-	-	-	-	-	88.19	7.78	10.10
	FA-GNN	71.40	6.66	8.36	70.95	10.09	9.68	87.66	8.42	6.50
	UFA	71.71	**17.38**	**18.95**	71.45	**14.49**	**14.12**	85.18	**22.59**	**18.41**
FairSIN	Clean	70.58	2.37	1.88	69.55	2.32	2.03	92.20	4.70	8.31
	G-FairAttack	-	-	-	-	-	-	89.70	19.45	**21.45**
	FA-GNN	70.81	5.41	6.77	68.00	9.12	7.85	87.66	9.42	11.50
	UFA	69.56	**14.79**	**15.87**	69.23	**17.65**	**17.42**	87.68	**26.80**	21.14

5.2 Attacking Binary-Classification Datasets with Binary Sensitive Attributes

Table 1 reports the results of fairness poisoning attacks. For standard GNNs, including GCN, GraphSAGE, APPNP, SGC, GAT, and GIN, UFA consistently achieves stronger attack performance than SOTA baselines across datasets, with higher Δ_{EO} and Δ_{DP} values, which increase by 18.16% and 19.18% on average, respectively. Meanwhile, UFA has only a minor impact on utility, causing a slight decrease in Acc. Moreover, our method remains effective against fairness-aware GNNs, including FairGNN, FairVGNN, and FairSIN, yielding average increases of 27.64% in Δ_{EO} and 27.78% in Δ_{DP}. This indicates that fairness-oriented GNNs remain vulnerable to UFA. Due to poor scalability, G-FairAttack and FATE times out on large graphs such as Pokec-N and Pokec-Z, exceeding 72 hours. The runtime results in Appendix F further confirm that UFA is the most

computationally efficient method. As shown in Table 2, the results of fairness evasion attacks are consistent with those of fairness poisoning attacks. Since NIFA is not applicable to fairness evasion attacks, we use FA-GNN as the baseline. The results show that UFA outperforms the baselines in most settings baselines on both traditional and fairness-aware GNNs, achieving average increases of 83.36% and 61.68% in Δ_{EO} and Δ_{DP}, respectively, for traditional GNNs, with comparable improvements for fairness-aware GNNs.

Table 3: Attack performance of UFA on fairness poisoning attacks against GNNs across AMiner-S and AMiner-L datasets.

Victim	Attack	AMiner-S			AMiner-L		
		ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
GCN	Clean	95.49	3.80	9.64	95.54	5.77	10.55
	UFA	93.53	14.84	13.91	95.36	9.83	12.73
GraphSAGE	Clean	95.19	5.33	11.09	95.13	6.13	8.96
	UFA	95.10	10.50	13.64	94.64	10.09	11.60
APPNP	Clean	95.77	3.03	9.07	94.97	7.47	8.95
	UFA	94.57	12.41	13.54	94.21	10.85	12.70
SGC	Clean	94.92	3.67	9.09	94.50	6.35	9.41
	UFA	93.55	13.96	13.61	93.73	11.71	12.09
GAT	Clean	95.02	4.72	10.17	95.05	6.59	9.09
	UFA	93.91	11.93	15.20	95.12	9.44	12.61
GIN	Clean	95.27	4.32	11.19	94.50	6.20	8.95
	UFA	93.45	15.44	18.08	93.25	12.93	12.32

5.3 Attacking Multi-Class Datasets with Non-Binary Sensitive Attribute

Unlike prior methods, we explicitly implement fairness attacks against multi-class tasks with non-binary sensitive attributes for the first time, without modifying the algorithmic design. As shown in Table 3, AMiner-S represents binary classification with non-binary sensitive attributes, whereas AMiner-L represents the more complex setting of multi-class classification with non-binary sensitive attributes. The results show that UFA can readily attack multi-class tasks or non-binary sensitive attributes, demonstrating broader generality than traditional settings limited to binary classification with binary sensitive attributes. Fairness evasion results on these datasets are reported in Appendix I.

6 Conclusion

This work reveals the fragility of fairness in GNNs under adversarial manipulation. We propose UFA, a universal fairness attack framework that leverages malicious node injection to conduct both poisoning and evasion attacks with minimal utility loss. By identifying boundary-sensitive nodes and manipulating shared representational mechanisms in GNNs, UFA amplifies outcome disparities across sensitive groups. Extensive experiments on real-world datasets show that injecting fewer than 0.2% adversarial nodes significantly undermines the fairness of both standard and fairness-aware GNNs. These findings highlight the pressing need for more robust, fairness-preserving GNN architectures.

Acknowledgments and Disclosure of Funding

The work of He Huang is supported by the National Natural Science Foundation of China (NSFC) under Grant 62625206 and 62332013. The work of Shuang Cui was supported in part by the NSFC under Grant No. 62502333, and the Basic Research Program of Jiangsu under Grant No. BK20250786. The work of Yu-E Sun is supported by NSFC under Grant 62672340.

References

[1] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[2] Sirui Chen, Jiawei Chen, Sheng Zhou, Bohao Wang, Shen Han, Chanfei Su, Yuqing Yuan, and Can Wang. Sigformer: Sign-aware graph transformer for recommendation. In *Proceedings of*

the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pages 1274–1284, 2024.

[3] Enyan Dai and Suhang Wang. Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 680–688, 2021.

[4] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International Conference on Machine Learning (ICML)*, pages 1115–1124. PMLR, 2018.

[5] Yushun Dong, Jing Ma, Song Wang, Chen Chen, and Jundong Li. Fairness in graph mining: A survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2023.

[6] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.

[7] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations (ICLR)*, 2019.

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

[9] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems (NeurIPS)*, 29, 2016.

[10] Arman Hasanzadeh, Ehsan Hajiramezanali, Shahin Boluki, Mingyuan Zhou, Nick Duffield, Krishna Narayanan, and Xiaoning Qian. Bayesian graph neural networks with adaptive connection sampling. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

[11] Zhichao Hou, Ruiqi Feng, Tyler Derr, and Xiaorui Liu. Robust graph neural networks via unbiased aggregation. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:110097–110130, 2024.

[12] Hussain Hussain, Meng Cao, Sandipan Sikdar, Denis Helic, Elisabeth Lex, Markus Strohmaier, and Roman Kern. Adversarial inter-group link injection degrades the fairness of graph neural networks. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 975–980. IEEE, 2022.

[13] J, Y Kang, R Xia, J Maciejewski, H Luo, and Tong. Deceptive fairness attacks on graphs via meta learning. In *Proceedings of the Twelfth International Conference on Learning Represen tations (ICLR)*, 2024.

[14] Mingxuan Ju, Yujie Fan, Chuxu Zhang, and Yanfang 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 (AAAI)*, volume 37, pages 4383–4390, 2023.

[15] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[17] Oyku Deniz Kose and Yanning Shen. Fast&fair: Training acceleration and bias mitigation for gnns. *Transactions on Machine Learning Research*, 2023.

[18] Yuni Lai, Yulin Zhu, Bailin Pan, and Kai Zhou. Node-aware bi-smoothing: Certified robustness against graph injection attacks. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 2958–2976. IEEE, 2024.

[19] I Lawrence and Kuei Lin. A concordance correlation coefficient to evaluate reproducibility. *Biometrics*, pages 255–268, 1989.

[20] Hongyi Ling, Zhimeng Jiang, Youzhi Luo, Shuiwang Ji, and Na Zou. Learning fair graph representations via automated data augmentations. In *International Conference on Learning Representations (ICLR)*, 2023.

[21] Zihan Luo, Hong Huang, Jianxun Lian, Xiran Song, Xing Xie, and Hai Jin. Cross-links matter for link prediction: rethinking the debiased gnn from a data perspective. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:79594–79612, 2023.

[22] Zihan Luo, Hong Huang, Yongkang Zhou, Jiping Zhang, Nuo Chen, and Hai Jin. Are your models still fair? fairness attacks on graph neural networks via node injections. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[23] Aryan Mobiny, Pengyu Yuan, Supratik K Moulik, Naveen Garg, Carol C Wu, and Hien Van Nguyen. Dropconnect is effective in modeling uncertainty of bayesian deep networks. *Scientific reports*, 11(1):5458, 2021.

[24] Dino Pedreshi, Salvatore Ruggieri, and Franco Turini. Discrimination-aware data mining. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 560–568, 2008.

[25] Tahleen Rahman, Bartlomiej Surma, Michael Backes, and Yang Zhang. Fairwalk: Towards fair graph embedding. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[26] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision, (ICCV) Venice, Italy, October 22-29, 2017*, pages 618–626. IEEE Computer Society, 2017.

[27] Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daume III, and Tudor Dumitras. When does machine learning {FAIL}? generalized transferability for evasion and poisoning attacks. In *27th USENIX Security Symposium (USENIX Security)*, pages 1299–1316, 2018.

[28] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *Proceedings of the Web Conference 2020*, pages 673–683, 2020.

[29] Maciej Sypetkowski, Frederik Wenkel, Farimah Poursafaei, Nia Dickson, Karush Suri, Philip Fradkin, and Dominique Beaini. On the scalability of gnns for molecular graphs. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:19870–19906, 2024.

[30] Karolina Szczepankiewicz, Adam Popowicz, Kamil Charkiewicz, Katarzyna Naecz-Charkiewicz, Michał Szczepankiewicz, Sławomir Lasota, Paweł Zawistowski, and Krystian Radlak. Ground truth based comparison of saliency maps algorithms. *Scientific Reports*, 13(1):16887, 2023.

[31] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[32] Lubos Takac and Michal Zabovsky. Data analysis in public social networks. In *International scientific conference and international workshop present day trends of innovations*, volume 1, 2012.

[33] Shuchang Tao, Qi Cao, Huawei Shen, Junjie Huang, Yunfan Wu, and Xueqi Cheng. Single node injection attack against graph neural networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM)*, pages 1794–1803, 2021.

[34] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[35] Jiakai Wang, Aishan Liu, Zixin Yin, Shunchang Liu, Shiyu Tang, and Xianglong Liu. Dual attention suppression attack: Generate adversarial camouflage in physical world. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 8565–8574, 2021.

[36] Yu Wang, Yuying Zhao, Yushun Dong, Huiyuan Chen, Jundong Li, and Tyler Derr. Improving fairness in graph neural networks via mitigating sensitive attribute leakage. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining (KDD)*, pages 1938–1948, 2022.

[37] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning (ICML)*, pages 6861–6871. Pmlr, 2019.

[38] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

[39] Paiheng Xu, Yuhang Zhou, Bang An, Wei Ai, and Furong Huang. Gfairhint: Improving individual fairness for graph neural networks via fairness hint. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 19(3):1–22, 2025.

[40] Cheng Yang, Jixi Liu, Yunhe Yan, and Chuan Shi. Fairsin: Achieving fairness in graph neural networks through sensitive information neutralization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, pages 9241–9249, 2024.

[41] Binchi Zhang, Yushun Dong, Chen Chen, Yada Zhu, Minnan Luo, and Jundong Li. Adversarial attacks on fairness of graph neural networks. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[42] Haotian Zhang and Xu Ma. Misleading attention and classification: an adversarial attack to fool object detection models in the real world. *Computers & Security (CS)*, 122:102876, 2022.

[43] Juzheng Zhang, Lanning Wei, Zhen Xu, and Quanming Yao. Heuristic learning with graph neural networks: A unified framework for link prediction. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 4223–4231, 2024.

[44] Yifei Zhang, Hao Zhu, Zixing Song, Piotr Koniusz, Irwin King, et al. Mitigating the popularity bias of graph collaborative filtering: A dimensional collapse perspective. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:67533–67550, 2023.

[45] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD (KDD)*, pages 2847–2856, 2018.

A Attack Settings

Attacker's knowledge: UFA operationalizes fairness attacks in both evasion attacks and poisoning attacks scenarios. To ensure real-world feasibility [14, 33, 22], UFA is formulated as a gray-box attack: the attacker has access only to the graph data (node features X, adjacency matrix A, labels Y, and sensitive attributes S), whereas the victim model's architecture and parameters remain hidden. Furthermore, as shown in Table 6, only a subset of nodes in the dataset have labels and sensitive attributes. UFA aims to induce group fairness disparities under this partially observed information setting. No feedback or information from the test set is accessible for attack optimization or parameter tuning. Consequently, all attacks are performed without any knowledge of the evaluation data, which preserves the integrity of the test assessment.

Attacker's capability: UFA implements fairness attacks by injecting malicious nodes into the graph data. Operating in a gray-box setting, UFA first trains a shadow model and employs a bi-level, multi-step optimization procedure to refine the features of the injected nodes iteratively. During this process, it directly targets the model's universally vulnerable fairness saliency map. Compared to existing gray-box fairness attacks [28, 22], this approach naturally endows the adversarial data crafted on the shadow model with cross-architecture transferability: divergent GNN variants tend to exhibit similar fairness saliency map patterns when evaluated on the same fairness criteria, allowing the same G_{adv} to compromise multiple victim models effectively.

Compared with methods that directly modify graph structure or node features [12, 41], UFA requires neither any discretization nor iterative traversal of all edges or nodes, thereby achieving substantial time savings. Moreover, node injection is generally more feasible in real-world settings, since an attacker typically cannot alter existing users' data or relationships. Unlike other injection attacks [22], UFA does not tamper with the training set or the ground-truth label set, and it can simultaneously realize both a fairness evasion attack and a fairness poisoning attack. In addition, Appendix J.5 considers a weaker threat model in which the attacker can observe the sensitive attributes of only a subset of nodes. This setting is used to evaluate the attack effectiveness of UPA under more restrictive information-access conditions.

B Proof of Lemma 4.1.

Lemma 4.1 can be formalized as follows:

Let G be a fixed graph structure, and $x_0 \in \mathbb{R}^p$ denote the given input feature vector. Suppose that two GNN models $\mathcal{M}_1$ and $\mathcal{M}_2$ induce two scalar objective functions on the same graph input: $f_1, f_2 : \mathbb{R}^p \to \mathbb{R}$. Define the input-gradient-based saliency map of two models as $\mathcal{H}^{(i)} := \nabla_x f_i(x_0), i = 1, 2$. Let

$$\mathcal{U}(x) := f_1(x) - f_2(x).$$

Assume that $\mathcal{U}$ is differentiable at x_0, and that there exist constants $\epsilon \geq 0$ and $r > 0$ such that, for any perturbation d satisfying $||d||_2 \leq r$,

$$|(f_1(x_0 + d) - f_1(x_0)) - (f_2(x_0 + d) - f_2(x_0))| \leq \epsilon ||d||_2.$$

Then,

$$||\mathcal{H}^{(1)} - \mathcal{H}^{(2)}||_2 \leq \epsilon.$$

Furthermore, if there exists a constant $\gamma > 0$ such that

$$\min\{||\mathcal{H}^{(1)}(x_0)||_2, ||\mathcal{H}^{(2)}(x_0)||_2\} \geq \gamma,$$

then the normalized saliency maps satisfy

$$||\frac{\mathcal{H}^{(1)}(x_0)}{||\mathcal{H}^{(1)}(x_0)||_2} - \frac{\mathcal{H}^{(2)}(x_0)}{||\mathcal{H}^{(2)}(x_0)||_2}||_2 \leq \frac{2\epsilon}{\gamma}.$$

Moreover, if θ denotes the angle between saliency maps, then

$$\cos\theta \geq 1 - \frac{2\epsilon^2}{\gamma^2}.$$

Proof. By the local response similarity assumption, for any $||d||_2 \leq r$, we have

$$|\mathcal{U}(x_0 + d) - \mathcal{U}(x_0)| \leq \epsilon ||d||_2.$$

Take an arbitrary unit vector $w \in \mathbb{R}^p$, let $d = t \cdot w, 0 < t \leq r$. Then,

$$|\frac{\mathcal{U}(x_0 + t \cdot w) - \mathcal{U}(x_0)}{t}| \leq \epsilon.$$

Since $\mathcal{U}$ is differentiable at x_0, taking the limit $t \to 0^+$ yields

$$|\nabla \mathcal{U}(x_0)^\top w| \leq \epsilon.$$

Since the above inequality holds for every unit vector w, we obtain

$$||\nabla \mathcal{U}(x_0)||_2 = \sup_{||w||_2=1} |\nabla \mathcal{U}(x_0)^\top w| \leq \epsilon$$

Moreover,

$$\nabla \mathcal{U}(x_0) = \nabla f_1(x_0) - \nabla f_2(x_0) = \mathcal{H}^{(1)}(x_0) - \mathcal{H}^{(2)}(x_0).$$

Therefore,

$$||\mathcal{H}^{(1)}(x_0) - \mathcal{H}^{(2)}(x_0)||_2 \leq \epsilon.$$

This proves the closeness of the unnormalized input-gradient-based saliency maps.

Next, we prove the closeness of the normalized saliency directions. Let

$$a := \mathcal{H}^{(1)}(x_0), \quad b := \mathcal{H}^{(2)}(x_0).$$

By the preceding result, $||a - b||_2 \leq \epsilon$. Assume that $||a||_2 \geq \gamma$, $||b||_2 \geq \gamma$. Then

$$\begin{aligned}
||\frac{a}{||a||_2} - \frac{b}{||b||_2}||_2 &= ||\frac{a-b}{||a||_2} + b \cdot (\frac{1}{||a||_2} - \frac{1}{||b||_2})||_2 \\
&\leq \frac{||a-b||_2}{||a||_2} + ||b||_2 \cdot |\frac{1}{||a||_2} - \frac{1}{||b||_2}| \\
&\leq \frac{||a-b||_2}{||a||_2} + \frac{|||b||_2 - ||a||_2|}{||a||_2} \\
&\leq 2 \cdot \frac{||a-b||_2}{||a||_2} \\
&\leq \frac{2}{\gamma} \cdot ||a-b||_2 \\
&\leq \frac{2 \cdot \epsilon}{\gamma}
\end{aligned}$$

Hence,

$$||\frac{\mathcal{H}^{(1)}(x_0)}{||\mathcal{H}^{(1)}(x_0)||_2} - \frac{\mathcal{H}^{(2)}(x_0)}{||\mathcal{H}^{(2)}(x_0)||_2}||_2 \leq \frac{2\epsilon}{\gamma}$$

Finally, let θ denotes the angle between a and b. Using the identity

$$||\frac{a}{||a||_2} - \frac{b}{||b||_2}||_2^2 = 2 - 2\cos\theta,$$

we have

$$2 - 2\cos\theta \leq 4 \cdot \frac{\epsilon^2}{\gamma^2},$$

Rearranging gives

$$\cos\theta \geq 1 - \frac{2\epsilon^2}{\gamma^2}.$$

□

Moreover, in UFA, both $\mathcal{L}_{\Delta_{DP}}$ and $\mathcal{L}_{\Delta_{EO}}$ consist only of group-wise averaging, finite summation, subtraction, and squared Euclidean norms over the prediction vectors. Hence, they are smooth quadratic functions of $\{z_v\}$. Therefore, whenever the GNN prediction map $x \mapsto M(G, x)$ is differentiable at x, the composite objective $f(x)$ is differentiable by the chain rule. For piecewise smooth GNNs, such as those using ReLU activations, this differentiability holds almost everywhere. Thus, our local response similarity result applies directly to the UFA fairness saliency objective.

C Pseudocode

Due to space limitations, we provide the pseudocode for the building injection target node set here. As shown in Algorithm 2, for each sensitive attribute group, UFA selects the top-k nodes as the injection target node set $\mathcal{J}$ by uncertainty score $\mathcal{Q}$. Finally, following prior work [40, 21, 22], the algorithm uniformly and randomly assigns sensitive attributes to the malicious nodes $|V'| = b$ and connects each malicious node to randomly chosen target nodes $\mathcal{J}$ within the corresponding group, completing the malicious node injection process.

Algorithm 2 Building injection target node set

Input: Graph $G = (V, E, X, Y)$;
Parameter: GNN with DropConnect $\mathcal{M}_D(\theta_D; \cdot)$; Loss function $\mathcal{L}_{\theta_D}$; Train epoch T_D; Number of sampling T_S; Learning rate ℓ_D; DropConnect rate ρ;
Output: Target node set $|\mathcal{J}| = k * |V|$;

Initializing the parameter θ_D of the model $\mathcal{M}_D$;
for 0, 1, ..., T_D **do**
    $\theta_D \leftarrow \theta_D - \ell_D \cdot \nabla_{\theta_D} \mathcal{L}_{\theta_D}$;
end for
$\mathcal{P}_{\mathcal{D}} = \frac{1}{T_s} \sum_{i=1}^{T_s} \mathrm{softmax}(\mathcal{M}_D(\theta_D; \rho, G))$;
for $p \in \mathcal{P}_{\mathcal{D}}$ **do**
    $\mathcal{Q} \leftarrow 1 - (p^{(1)} - p^{(2)}) - \sum_c p^{(c)} \log p^{(c)}$
end for
while $|\mathcal{J}| < k$ **do**
    Select a node with the largest values in $\mathcal{Q}$ and add it into $\mathcal{J}$;
end while
return $\mathcal{J}$

D Additional Descriptions on Baselines

In this section, we give an introduction to the baselines used in our experiment.

- **FA-GNN [12]:** FA-GNN is among the earliest works to investigate adversarial attacks on GNN fairness. Under a predefined attack budget, it perturbs the graph structure by adding edges between nodes belonging to different label classes and sensitive attribute groups, thereby amplifying inter-group disparities. FA-GNN supports both fairness evasion and fairness poisoning attack settings.
- **G-FairAttack [41]:** G-FairAttack proposes a surrogate fairness loss that serves as a unified upper bound for multiple group fairness metrics. Based on this loss, it greedily flips selected edges to progressively increase fairness violations, enabling effective fairness evasion and poisoning attacks through structural perturbations.
- **FATE [13]:** FATE introduces a general meta-learning framework for fairness attacks on graph learning models by formulating the attack process as a bi-level optimization problem. The framework is model-agnostic and can flexibly adapt to different fairness definitions, GNN architectures, and graph manipulation strategies.
- **NIFA [22]:** NIFA is a state-of-the-art node injection-based fairness attack method for GNNs. It identifies vulnerable nodes via uncertainty maximization and homophily-aware principles, and further optimizes the features of injected nodes to exacerbate group-level fairness disparities. NIFA primarily focuses on fairness poisoning attacks and does not address fairness evasion scenarios.

E Additional Descriptions on Victim Models

In this section, we give an introduction to the victim models used in our experiment.

Classical classification GNNs:

- **GCN [16]:** GCN is a seminal graph neural network that employs a first-order approximation of spectral graph convolutions. It propagates and aggregates feature information from immediate neighbors through linear transformations and normalized adjacency matrices, enabling effective semi-supervised node representation learning.
- **GraphSAGE [8]:** GraphSAGE is an inductive graph representation learning framework that generates node embeddings by sampling and aggregating features from a node's local neighborhood using learnable aggregation functions (e.g., mean, LSTM, and pooling). This design allows the model to generalize to unseen nodes and graphs.
- **APPNP [7]:** APPNP is a personalized PageRank-based propagation method that decouples feature transformation from neighborhood propagation. By repeatedly mixing initial node features with propagated messages through a teleport probability, APPNP enables deep information diffusion while mitigating the oversmoothing problem.
- **SGC [37]:** SGC (Simplified Graph Convolution) removes nonlinear activations and collapses multiple weight matrices in standard GCNs, reducing the model to a single linear transformation preceded by K-hop neighborhood feature aggregation, thereby significantly simplifying model training and inference.
- **GAT [34]:** GAT introduces masked self-attention mechanisms into graph neural networks, allowing nodes to assign adaptive, learnable importance weights to their neighbors. This enables more flexible and discriminative aggregation of neighborhood information when computing node embeddings.
- **GIN [38]:** GIN designs a graph neural network architecture inspired by the Weisfeiler-Lehman graph isomorphism test, using a sum-based aggregation scheme followed by multilayer perceptrons to update node representations. This enables GIN to achieve strong expressive power in distinguishing different graph structures and learning discriminative graph-level embeddings.

Fairness-aware GNNs:

- **FairGNN [3]:** FairGNN is one of the earliest fairness-aware graph neural networks that incorporates an adversarial regularization strategy and a sensitive-attribute discriminator into the training process. By jointly optimizing node classification and sensitive attribute prediction, FairGNN mitigates bias amplification during message passing, even when only a limited portion of sensitive attribute labels is available.
- **FairVGNN [36]:** FairVGNN proposes a multi-view fairness learning framework that identifies and suppresses feature views highly correlated with sensitive attributes. By adaptively masking or reweighting sensitive-related views during aggregation, FairVGNN reduces protected attribute leakage and improves both group-level and individual-level fairness.
- **FairSIN [40]:** FairSIN introduces a fairness-enhanced semantic interaction network that integrates a structural debiasing module into a GNN backbone. Through contrastive learning across sensitive groups, FairSIN aligns node representations to reduce group-wise disparities, leading to improved performance under multiple fairness metrics, such as statistical parity and equalized odds.

F Efficiency Analysis

In this section, we compare the runtime efficiency of UFA with state-of-the-art attack methods, as shown in Table 4. For G-FairAttack, it is only able to complete within 19.7 hours on the DBLP dataset, while its runtime exceeds 72 hours on Pokec-N and Pokec-Z, rendering it impractical for large-scale graphs. UFA$_m$ denotes the total computation time of the saliency maps, and UFA$_d$ denotes the total computation time of the node injection. UFA runs slightly faster than NIFA, despite requiring

additional time for computing the saliency map. This is because UFA targets the most sensitive feature dimensions with respect to the GNN output, allowing it to achieve stronger fairness degradation with smaller perturbations. As a result, it requires fewer optimization iterations to construct effective malicious node features.

Table 4: Runtime comparison of SOTA baselines and UFA across different datasets, where timeout error is indicated by -. The best results are highlighted in bold.

	Pokec-Z	Pokec-N	DBLP
FA-GNN	107.33 s	89.35 s	48.03 s
G-FairAttack	-	-	19.7 h
NIFA	86.19 s	75.08 s	53.77 s
UFA$_m$	21.91 s	20.01 s	17.95 s
UFA$_d$	28.74 s	29.23 s	23.92 s

Notably, NIFA is incapable of performing fairness evasion attacks. In contrast, UFA can leverage the same adversarially poisoned data to simultaneously achieve both poisoning attacks and evasion attacks, without requiring repeated training. This enables UFA to handle more complex gray-box scenarios effectively.

G Analyzing Fairness Saliency Maps of Different Models

For Lemma 4.1, due to the structural dependencies and complex topological properties inherent in graph data, which are absent in grid-based image data, it is not feasible to visually compare the differences in fairness saliency maps using simple visualization methods. Instead, we adopt statistical approaches [30] to quantify the differences in fairness saliency maps generated by different models on the same input data. Specifically, we employ the following commonly used similarity metrics:

- **Pearson Correlation Coefficient:** If two saliency maps exhibit an overall linear relationship in their numerical values—that is, the sensitivity scores from one model are approximately a linear scaling or translation of those from another—then the Pearson coefficient will approach 1. A high $Pearson$ value (> 0.8) indicates a strong consistency in identifying which features are important or unimportant.

$$Pearson(x, y) = \frac{\sum_{i=1}^{d}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{d}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{d}(y_i - \bar{y})^2}}$$

  where $\bar{x} = \frac{1}{d}\sum_{i=1}^{d} x_i$ and $\bar{y} = \frac{1}{d}\sum_{i=1}^{d} y_i$

- **Top-k Jaccard Index:** We extract the index sets corresponding to the Top-k entries with the highest absolute values from each saliency map and compute the Jaccard index between them. A high $Jaccard$ score (> 0.8) indicates that both models focus their attention on the same positions, which directly suggests a high level of agreement on the most critical dimensions.

$$Jaccard(x, y) = \frac{|I_x \cap I_y|}{|I_x \cup I_y|}$$

  where I represents the position with the largest absolute value k.

- **Cosine Similarity:** This metric evaluates whether the two high-dimensional vectors share a similar direction. A high $Cosine$ similarity (> 0.8) indicates that the two saliency maps exhibit highly consistent distributional patterns.

$$Cosine(x, y) = \frac{x^\top y}{||x||_2 \cdot ||y||_2}$$

- **Concordance Correlation Coefficient (CCC) [19]:** This metric assesses not only the linear relationship (covariance) between two sets of values but also penalizes differences in their means and variances. A high CCC (> 0.8) indicates that the two saliency maps

are not only consistent in their variation trends but also exhibit similar average sensitivity levels and distributional spread, thereby validating their overall agreement from multiple perspectives.

$$CCC(x, y) = \frac{\frac{2}{d}\sum_{i=1}^{d}(x_i - \bar{x})(y_i - \bar{y})}{var(x) + var(y) + (\bar{x} - \bar{y})^2}$$

where $var(x) = \frac{1}{d}\sum_{i=1}^{d}(x_i - \bar{x})^2$.

As shown in Table 5, we use GCN as the baseline and compare it with GraphSAGE, APPNP, SGC, and GAT on the aforementioned similarity metrics. The experiments are conducted on the Pokec-Z dataset, while similar trends are observed on the other two datasets. The results indicate that, given the same input, the fairness saliency maps generated by these structurally different models are highly consistent. In particular, APPNP and SGC exhibit especially high similarity, which can be attributed to their shared architectural characteristics—both are propagation-dominated GNNs with decoupled transformation and diffusion processes. In contrast, GCN and GraphSAGE adopt stacked nonlinear aggregation mechanisms. It can be observed that the Pearson correlation coefficient and the cosine similarity are nearly identical. This is because the positive and negative values of the model gradients are approximately balanced, rendering the Pearson correlation coefficient almost equivalent to the cosine similarity in this context.

Table 5: Similarity metrics of different models on the Pokec-Z dataset (Reference: GCN)

	$Pearson$	$Jaccard$	$Cosine$	CCC
GCN	1	1	1	1
GraphSAGE	0.9569	0.8583	0.9569	0.9426
APPNP	0.8864	0.8129	0.8865	0.8504
SGC	0.8865	0.8127	0.8867	0.8508
GAT	0.8408	0.8031	0.8409	0.8146

For fixed-propagation or propagation-dominated GNNs such as GCN, SGC, and APPNP, the input gradient admits a common form $P^\top GW^\top$, where P is a graph propagation operator. Since these models share the same graph topology and use propagation operators derived from the same normalized adjacency matrix, their fairness gradients may concentrate on similar feature dimensions and neighborhoods. GAT does not strictly belong to this class because its attention coefficients are input-dependent. However, under a local linearization around a fixed graph input, the attention matrix can be viewed as locally fixed up to higher-order attention-derivative terms, leading to a similar gradient decomposition. Consequently, UFA targets the shared vulnerable regions of different models by attacking their structurally similar fairness saliency maps. This enables the malicious data optimized on a shadow model to generalize effectively to other architectures, achieving efficient and accurate fairness attacks with minimal perturbations.

H Datasets and Parameter Setting Details

A summary of the datasets is provided in Table 6. All experiments adopt a two-layer GCN as the shadow model with a hidden dimension of 128 and a dropout rate of 0.5. We set the number of DropConnect GCN sampling repetitions to $T_s = 20$, with optimization steps of $T_D = 400$, $\rho = 0.5$, and a learning rate of $\ell_d = 0.001$. The first-level and second-level optimization steps are set to $T_q = 50$ and $T_f = 50$. Epoch of bi-level iteration $T_e = 10$, and the proportion of high-scoring candidate nodes is $k = 0.4$. The shadow model's learning rate is $\ell_q = 0.001$, while the learning rate for the injected node features is $\ell_f = 0.1$. All optimizers are set to Adam [15]. All experiments were carried out on a Linux server with Intel Xeon W-5317 CPU and NVIDIA A100 GPU.

For the victim models, we adopt standard configurations provided by the Deep Graph Library. Specifically, we implement GCN with a two-layer convolutional architecture; GraphSAGE with two layers of mean-pooling SAGE convolutions; and APPNP with one fully connected layer followed by one APPNP layer, where the teleport probability $\alpha = 0.2$ and the number of propagation steps $k = 1$. GAT is implemented with two GATConv layers using a single attention head per layer. A ReLU activation is applied after the first layer, and the final layer outputs node-level predictions. For

Table 6: Overview of the Datasets

	Pokec-Z	Pokec-N	DBLP	AMiner-S	AMiner-L
Nodes	67,796	66,569	20,111	39,424	129,726
Edges	617,958	517,047	57,508	144,344	1,311,804
Label	2	2	2	2	3
Sensitive	2	2	2	7	7
Labeled nodes	10,262	8,797	3,196	6,826	43,469
Feature dim	276	265	2530	5694	5694

SGC, we use a single SGConv layer with $k = 1$. All of the above models are trained without dropout for 500 epochs using a learning rate of 0.01. For the following models, we set the dropout rate to 0.5. For FairGNN, we use GCN as the backbone model, with a learning rate of 0.001, trained for 1000 epochs, and set the regularization parameters $\beta = 0.01$ and $\alpha = 2$. For FairVGNN, the generator is trained with a learning rate of 0.01 for a total of 500 iterations. Both the discriminator and classifier are trained for 5 steps in each iteration. The cut-off threshold is set to $c = 0.1$, the classification loss weight to 0.5, the adversarial loss weight to 1, and the view generation loss weight to 0.1. For FairSIN, the weights of the SP/EO regularization, and Gradient Reversal Layer are set to 1, 5, and 1, respectively.

Table 7: Hyper-parameter settings of UFA

	Pokec-Z	Pokec-N	DBLP	AMiner-S	AMiner-L
α	$1e11$	$1e11$	$1e5$	$1e7$	$1e9$
b	100	80	30	50	200
b_e	50	50	25	50	50

For all baseline algorithms, we ensure that the same number of nodes and edges are modified. For NIFA, we strictly follow the hyperparameter settings provided in the original paper. For G-FairAttack, we adopt the original hyperparameters for both poisoning and evasion attacks. To enable it to run on the DBLP dataset within a reasonable time, we set the candidate edge ratio to 0.0005, the learning rate to 0.01, and the number of edge update iterations to 1000. For UFA, we select the top-k nodes with the highest uncertainty scores. The number of injected malicious nodes b, the number of connected edges b_e, and the fairness saliency map attack parameters α on different datasets are summarized in Table 7.

I Fairness Evasion Attacks on AMiner-S and AMiner-L Datasets

In this subsection, we present the performance of fairness evasion attacks on binary-classification datasets with multi-valued sensitive attributes and on multi-class datasets. As shown in Table 8, our method remains effective in these more general settings.

Table 8: Attack performance of UFA on fairness evasion attacks against GNNs across AMiner-S and AMiner-L datasets.

Victim	Attack	AMiner-S			AMiner-L		
		ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
GCN	Clean	95.36	3.99	10.84	95.57	6.14	10.04
	UFA	92.80	19.15	15.44	91.22	21.64	22.25
GraphSAGE	Clean	95.17	6.56	11.95	94.81	5.62	9.25
	UFA	92.47	19.16	14.97	90.67	21.22	21.72
APPNP	Clean	94.69	3.10	10.01	94.84	7.01	9.17
	UFA	92.58	19.52	15.44	90.35	20.92	21.43
SGC	Clean	94.22	3.39	9.91	94.51	6.81	9.60
	UFA	91.98	19.59	15.45	90.52	21.07	21.71
GAT	Clean	94.76	5.42	11.47	95.18	6.60	9.04
	UFA	94.13	13.29	14.64	94.92	11.50	12.64
GIN	Clean	95.52	3.95	11.44	94.59	5.34	8.51
	UFA	90.88	19.26	15.31	91.13	21.03	21.70

J Ablation Study and Hyperparameter Analysis

J.1 Ablation Study of Attacking

In this subsection, we conduct ablation studies by isolating different components of UFA to examine their individual contributions. The following settings are considered: **(1)** U denotes the original UFA configuration without modification; **(2)** U_{var} adopts the target node selection strategy proposed in NIFA [22], which selects nodes based on the variance of multiple model sampling rounds; **(3)** U_r randomly selects target nodes for malicious injection; **(4)** U_c sets the parameter $\alpha = 0$ in UFA, which controls the fairness saliency map attack, to isolate the effect of the proposed fairness saliency map attack. **(5)** U_N denotes a variant of UFA in which the node feature optimization component is replaced with a direct optimization of a fairness objective similar to that adopted in NIFA [22]. **(6)** U_o denotes a variant of UFA in which the bi-level alternating optimization process is replaced by training the shadow model only once, followed by directly optimizing the node features. As shown in Table 9, we perform both poisoning and evasion attacks using GCN as the victim model across three datasets.

Table 9: Ablation study of each setting in UFA. The victim model is GCN.

		Pokec-Z			Pokec-N			DBLP		
		ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
Poison	Clean	71.39	4.75	6.75	70.85	2.80	1.22	95.93	1.35	3.52
	U_r	71.31	11.66	13.47	70.23	8.59	8.11	94.72	12.26	5.78
	U_{var}	70.79	14.18	16.00	70.08	9.40	9.28	92.61	20.41	13.67
	U_c	71.06	4.93	7.04	70.52	3.60	1.79	95.64	6.83	3.76
	U_N	71.14	16.37	17.19	70.73	9.79	8.99	93.12	17.23	11.65
	U_o	71.21	16.74	18.55	70.89	10.94	10.21	92.93	19.51	14.20
	U	71.25	16.87	18.94	70.29	10.82	10.66	93.47	20.71	14.34
Evasion	Clean	71.58	5.21	7.25	71.10	2.30	1.31	95.95	2.25	4.01
	U_r	71.31	17.87	19.55	69.58	15.85	16.32	90.14	23.82	17.37
	U_{var}	70.04	22.87	24.38	70.02	16.55	16.97	90.21	23.01	17.46
	U_c	71.50	5.62	7.54	70.97	2.60	1.62	93.55	11.52	6.32
	U_N	70.25	19.44	20.69	70.91	10.59	10.15	91.71	18.01	10.66
	U_o	71.33	21.52	24.09	70.04	18.54	18.22	90.67	26.24	20.61
	U	70.62	22.91	24.53	69.80	18.93	18.55	89.52	27.64	20.02

The results indicate that removing UFA's proposed method (U_r) for identifying nodes most likely to cross the decision boundary leads to a consistent decline in fairness attack performance across all three datasets. Similarly, replacing this strategy with the one proposed in NIFA (U_{var}) also results in degraded attack performance. This is because NIFA selects target nodes based on the variance across multiple model sampling rounds, which reflects model instability or sensitivity to structural perturbations rather than providing a direct measure of proximity to the decision boundary. In contrast, UFA leverages entropy and inter-class margin as direct indicators of a sample's closeness to the decision boundary in the prediction space. Despite removing the bi-level optimization process (U_o), one-shot UFA consistently degrades the fairness of different victim models. These results demonstrate that the saliency-map-based attack objective does not rely on the NIFA-style alternating optimization procedure and remains effective under a simplified attack pipeline. In contrast, when the saliency-map objective is replaced with a fairness objective similar to that adopted in NIFA (U_N), the resulting method performs worse than the saliency-map-based approach, particularly in the evasion-attack setting. Furthermore, when the fairness saliency map attack is disabled (U_c), the attack effectiveness drops significantly, confirming the crucial role of the proposed feature optimization strategy in enabling effective fairness attacks.

J.2 Ablation Study of $\mathcal{M}_D$

In this subsection, we compare the impact of different DropConnect backbone models $\mathcal{M}_D$ on attack performance. $\mathcal{M}_D$ is used to identify nodes that are most likely to cross the decision boundary as the targets for attack injection. As shown in Table 10, we select GCN, SGC, and APPNP as backbone models, and poisoning attacks on the GCN model. The results indicate that the target selection

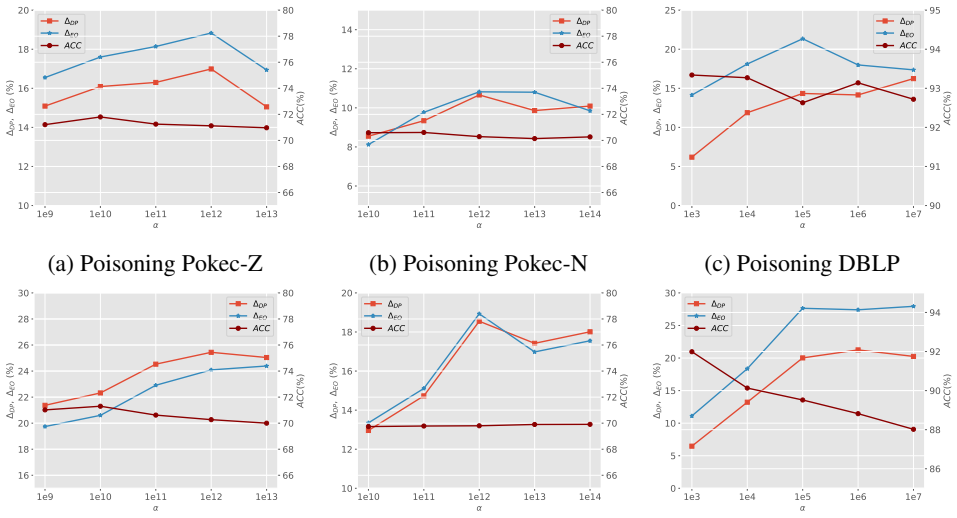
process does not strictly depend on the specific architecture of $\mathcal{M}_D$; all these models achieve strong attack performance.

Table 10: Ablation Study of $\mathcal{M}_D$

	Pokec-Z			Pokec-N			DBLP		
$\mathcal{M}_D$	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
$\mathcal{M}_D$-GCN	71.25	16.87	18.94	70.29	10.82	10.66	93.47	20.71	14.34
$\mathcal{M}_D$-SGC	70.46	16.21	18.87	71.01	11.02	10.74	92.55	19.48	14.51
$\mathcal{M}_D$-APPNP	70.99	16.89	18.85	70.43	10.91	10.30	93.47	20.71	13.92

J.3 The impact of α

We investigate the impact of different values of α on the attack performance of UFA. As defined in Section 4.3, α controls the strength of UFA's attack on the fairness saliency map. The detailed results are shown in Figure 2, where GCN is adopted as the victim model and both poisoning (Figure 2 a, b, and c) and evasion attacks (Figure 2 d, e, and f) are conducted across three datasets. It can be observed that increasing α does not necessarily lead to improved attack performance Δ_{EO} and Δ_{DP}. This may be attributed to the fact that excessively large values of α can induce numerical oscillations in the optimizer, potentially causing it to skip over optimal regions. Additionally, a slight decrease in ACC is observed, which is expected, as an overly large α may overshadow the utility loss $\mathcal{L}_{CE}$ in the loss function $\mathcal{L}$. Therefore, by conducting a range search for α, we conclude that the value adopted in this paper is reasonable.



(a) Poisoning Pokec-Z (b) Poisoning Pokec-N (c) Poisoning DBLP

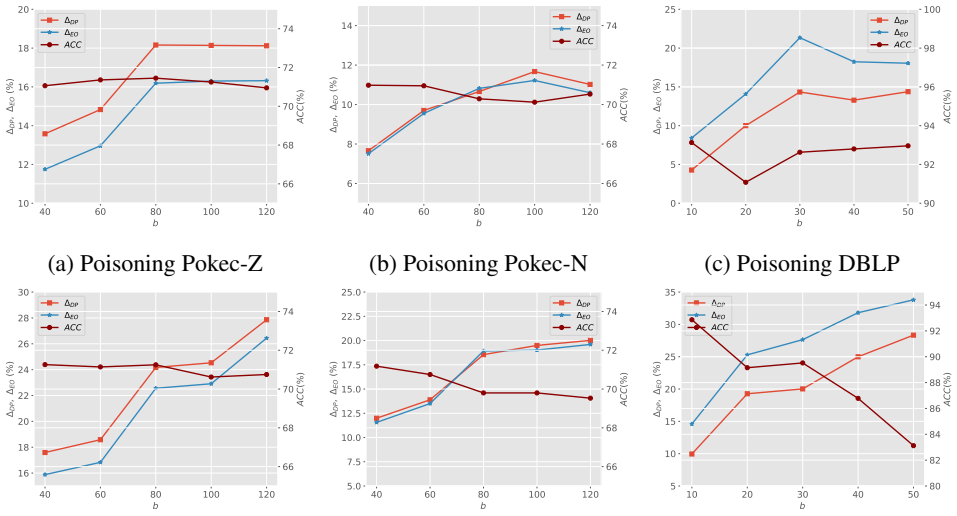
(d) Evasion Pokec-Z (e) Evasion Pokec-N (f) Evasion DBLP

Figure 2: The impact of different values of α on the attack performance (%) of UFA. GCN is used as the victim model, and fairness poisoning and fairness evasion attacks are conducted on three datasets, respectively. The x-axis denotes the varying values of α, while the left y-axis represents Δ_{EO} and Δ_{DP}, and the right y-axis corresponds to ACC.

J.4 The impact of b

We investigate the impact of different values of b on the attack performance of UFA. Here, b denotes the budget, i.e., the number of malicious nodes to be injected, which directly influences the attack performance. As illustrated in Figure 3, we adopt GCN as the victim model and conduct both poisoning (a, b, and c) and evasion attacks (d, e, and f) across three datasets. It can be observed that the attack performance Δ_{EO} and Δ_{DP} of UFA improve progressively with increasing b. This is intuitive, as a larger number of malicious nodes introduces more adversarial features into the model's message propagation process. Meanwhile, ACC exhibits a slight decrease, which can be attributed

to the greater structural perturbation caused by injecting additional malicious nodes. Therefore, based on a range search over b, the chosen value of b in this work is deemed appropriate.



(a) Poisoning Pokec-Z (b) Poisoning Pokec-N (c) Poisoning DBLP

(d) Evasion Pokec-Z (e) Evasion Pokec-N (f) Evasion DBLP

Figure 3: The impact of different values of b on the attack performance (%) of UFA. GCN is used as the victim model, and fairness poisoning and fairness evasion attacks are conducted on three datasets, respectively. The x-axis denotes the varying values of b, while the left y-axis represents Δ_{EO} and Δ_{DP}, and the right y-axis corresponds to ACC.

J.5 The impact of k and Partial sensitive attribute knowledge

We investigate the impact of different values of k on the attack performance of UFA. k is the proportion of high-scoring candidate nodes. We use a unified default value of $k = 40\%$ in all experiments. As shown in Table 11, we evaluate $k = \{10\%, 30\%, 40\%, 60\%, 80\%\}$ for poisoning attacks against GCN on three dataset. Attack effectiveness first increases and then decreases with k: a small k overly restricts the candidate set and concentrates injected edges in limited regions, whereas a large k includes less vulnerable nodes farther from the decision boundary. Importantly, UFA achieves comparable performance for $k \in [0.3, 0.6]$, suggesting that it does not require fine-grained tuning. We will add this analysis to the revised paper.

Table 11: The impact of different values of k on the poisoning attack performance (%) of UFA.

	Pokec-Z			Pokec-N			DBLP		
	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
$k = 10\%$	71.38	11.72	13.80	71.08	7.66	7.68	95.03	8.08	3.56
$k = 30\%$	71.47	15.40	17.33	70.41	10.13	9.79	93.87	19.54	13.41
$k = 40\%$	71.25	16.87	18.94	70.29	10.82	10.66	93.47	20.71	14.34
$k = 60\%$	71.83	15.99	16.83	70.37	10.79	10.38	92.41	18.96	12.45
$k = 80\%$	70.74	11.71	13.67	70.22	8.04	7.91	93.52	14.71	8.68

In addition, we also consider a more restrictive gray-box setting, where the attacker can access only partial node sensitive attributes. In this setting, the defender still has access to the complete graph data. As shown in Table 12, we randomly mask the sensitive attributes of 10%, 20%, 30%, and 40% of nodes in each graph dataset, and evaluate the poisoning attack performance against GCN on three datasets. According to the design of UFA, missing sensitive attributes do not affect the vulnerable-node scoring mechanism based on class margins and prediction entropy. However, they influence the group fairness loss, group-specific saliency maps, and the assignment of injected nodes to sensitive groups. Consequently, the attack performance slightly decreases as the availability of sensitive attributes decreases. Nevertheless, UFA remains highly effective even when 40% of

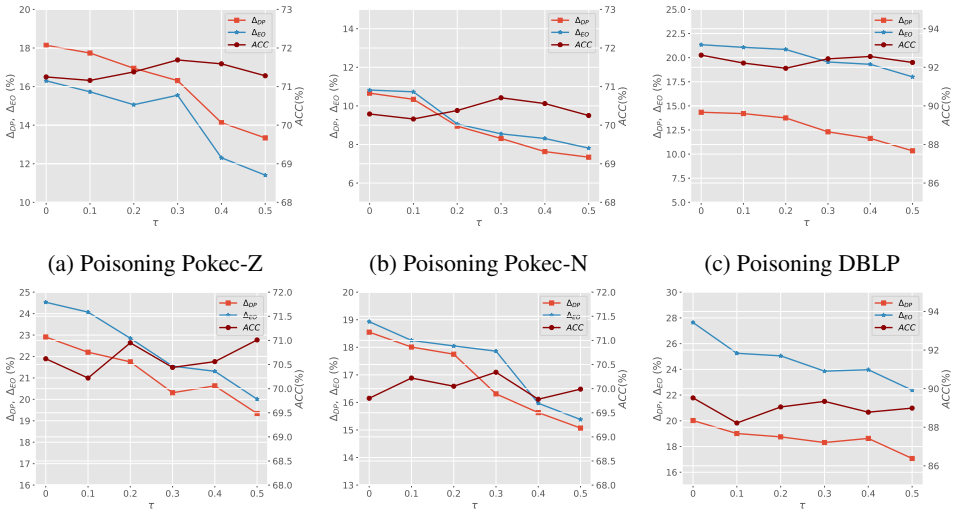
the sensitive attributes are masked, demonstrating its broader applicability under more restrictive gray-box scenarios with incomplete sensitive-attribute information.

Table 12: Fairness poisoning attack GCN on a more restrictive gray-box setting.

	Pokec-Z			Pokec-N			DBLP		
	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
10%	71.30	16.30	18.19	70.97	9.20	9.41	92.66	12.95	16.47
20%	71.29	15.65	17.52	70.06	8.86	9.09	92.01	12.13	14.89
30%	71.48	13.42	15.91	70.14	8.57	8.64	93.37	11.87	13.18
40%	71.44	12.88	15.67	70.60	8.22	8.48	93.97	10.00	12.37

K Defenses against Fairness Attacks on GNNs

For defending against UFA, we analyze the strategies separately for poisoning attacks and evasion attacks. For fairness poisoning attacks, one can monitor the fairness saliency map of the model during the training phase to identify features that contribute most to fairness bias. By isolating nodes associated with these features, it is possible to effectively block the influence of injected malicious nodes. For fairness evasion attacks, during the inference phase, a pre-trained detection model can be employed to identify nodes that are likely to be pushed across the decision boundary. Selected nodes can then be filtered to mitigate the impact of such attacks.



(a) Poisoning Pokec-Z (b) Poisoning Pokec-N (c) Poisoning DBLP

(d) Evasion Pokec-Z (e) Evasion Pokec-N (f) Evasion DBLP

Figure 4: The impact of different defense parameter τ on the attack performance (%) of UFA. GCN is used as the victim model, and fairness poisoning and fairness evasion attacks are conducted on three datasets, respectively. The x-axis denotes the varying values of τ, while the left y-axis represents Δ_{EO} and Δ_{DP}, and the right y-axis corresponds to ACC.

To simulate the aforementioned defense strategy, we artificially mask a subset of nodes that are most susceptible to being pushed across the decision boundary. We define the defense parameter τ as the proportion of the injection target node set $\mathcal{J}$ randomly masked. We use GCN as the victim model and conduct both poisoning and evasion attacks across three datasets under different defense parameter settings. As shown in Figure 4 (a, b, and c), for UFA's fairness poisoning attacks, increasing the masking ratio progressively reduces the attack performance. Although this indicates a partial defensive effect, the fairness metrics still fall short compared to models trained on clean datasets. A similar issue is observed for UFA's fairness evasion attacks, as illustrated in Figure 4 (d, e, and f), suggesting that UFA exhibits strong robustness against such defensive mechanisms.

Furthermore, we evaluate the defense performance of Bi-Smoothing [18], a defense method specifically designed against node-injection attacks, with the dropout rate set to 0.25. We also include

Table 13: Fairness poisoning attacks on Bi-Smoothing and RUNG.

		Pokec-Z			Pokec-N			DBLP		
		ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
Bi-Smoothing	Clean	70.63	2.36	4.26	71.30	2.94	2.19	96.12	3.82	6.25
	UFA	71.13	13.09	14.32	70.51	7.42	7.17	93.61	11.56	18.45
RUNG	Clean	70.47	2.61	4.70	70.94	2.87	2.28	96.36	2.30	0.54
	UFA	70.62	11.13	13.16	70.65	8.44	8.05	94.67	8.80	13.89

RUNG [11] as a defense baseline, whose original paper specifically investigates robustness against node-injection attacks. RUNG iteratively reweights graph edges, preserving information from similar neighbors while reducing the weights of or removing suspicious neighboring nodes with significant representation discrepancies. As shown in Table 13, the results indicate that UFA remains effective even under this targeted defense setting. Therefore, a more direct and effective approach is to conduct real-time analysis of the fairness of model outputs, rather than focusing solely on traditional utility performance. This allows early identification of data vulnerabilities to prevent more significant damage. These findings highlight the need for future research to develop more efficient and adaptive defense strategies.

L Structural Analysis of Poisoned Graph

To prevent the poisoned graph from being detected by defenders, we aim to minimize the difference between the graph with injected malicious nodes and the original graph. Therefore, to measure the impact of UFA on the original graph data, we use change rates Δ to evaluate the differences between the two graphs across several metrics. Specifically, we consider the following metrics:

- **Average Degree:** The average degree of all nodes in the graph, representing the average number of connections per node, reflects the overall connectivity and density of the graph.
- **Global Clustering Coefficient:** The ratio of the number of triangles to the number of connected triplets in the graph, measuring the tendency of nodes to form triangles (tightly connected communities) within the network.
- **Degree Assortativity:** which measures whether connected nodes have similar degrees (i.e., high-degree nodes tend to connect with high-degree nodes, and low-degree nodes with low-degree nodes).
- **Mean Node Feature Vector:** The mean of all node feature vectors in the graph, describing the overall central tendency or average representation of node features.
- **Gini Coefficient:** Measures the disparity in the distribution of node feature values.

As shown in Table 14, the hyperparameters for generating the poisoned graph are consistent with those used in the experiments. We observe that the difference between the poisoned graph and the original graph is minimal—less than 2%—making it difficult for defenders to detect malicious data through graph structure or node feature analysis. This demonstrates that UFA can achieve fairness attacks with minimal perturbations, exhibiting excellent stealthiness.

Table 14: Comparison between the poisoned graph generated by UFA and the clean graph across different datasets in terms of structural and statistical metrics.

		Average Degree		Global Clustering Coefficient		Degree Assortativity		Mean Node Feature Vector		Gini Coefficient	
		Value	Δ	Value	Δ	Value	Δ	Value	Δ	Value	Δ
Pokec-Z	Clean	19.22	0	0.0746	0	0.2108	0	45.74	0	0.3437	0
	Poisoned	19.35	0.63%	0.0741	0.72%	0.2106	0.11%	45.77	0.07%	0.3438	0.02%
Pokec-N	Clean	16.53	0	0.0712	0	0.1978	0	44.66	0	0.3490	0
	Poisoned	16.64	0.66%	0.0706	0.79%	0.1967	0.55%	44.68	0.06%	0.3491	0.03%
DBLP	Clean	6.71	0	0.2639	0	0.1335	0	0.9071	0	0.5716	0
	Poisoned	6.78	0.98%	0.2598	1.46%	0.1317	1.24%	0.9087	0.17%	0.5722	0.10%
AMiner-S	Clean	3.66	0	0.1836	0	0.2050	0	14.10	0	0.6352	0
	Poisoned	3.73	1.91%	0.1844	0.43%	0.1966	1.04%	14.19	0.91%	0.6374	0.35%
AMiner-L	Clean	10.11	0	0.1759	0	0.2040	0	14.38	0	0.6386	0
	Poisoned	10.18	0.68%	0.1744	0.85%	0.2019	1.87%	14.45	0.59%	0.6402	0.24%

M Broader Impact

This work studies fairness attacks on graph neural networks through malicious node injection. The positive societal impact of this work is that it exposes the fragility of fairness in GNN-based systems and highlights the need for more robust fairness-preserving graph learning methods, especially in high-stakes applications where biased predictions may affect different demographic groups. By revealing such vulnerabilities, this study can help researchers and practitioners design better defense mechanisms and conduct more careful fairness auditing before deployment. However, this work also has potential negative societal impacts. Since UFA is an attack framework, it could be misused by malicious actors to intentionally amplify group-level unfairness or manipulate graph-based decision systems. To mitigate this risk, we present the method as a controlled robustness study, evaluate it on benchmark datasets, and discuss potential defense strategies, including monitoring fairness saliency maps and detecting vulnerable nodes during training or inference. The goal of this work is to raise awareness of fairness vulnerabilities and motivate future defenses rather than facilitate harmful deployment. Similarly, we provide possible defense strategies in the Appendix K

N Limitations

Although UFA demonstrates strong effectiveness in both fairness poisoning and evasion settings, it still has several limitations. First, while UFA is designed to preserve model utility, certain dataset–victim model combinations may exhibit slightly more noticeable accuracy degradation, which could reduce attack stealthiness. Second, UFA is formulated under a gray-box threat model where the attacker has access to graph data, including node features, graph structure, labels, and sensitive attributes; extending UFA to stricter black-box settings with less side information remains an important direction. Third, the transferability of UFA relies on shared fairness saliency patterns across GNN architectures, and its effectiveness may weaken when victim models exhibit substantially different local responses or incorporate stronger robustness mechanisms. Finally, our evaluation focuses on node classification and group fairness metrics on five benchmark datasets, and future work should investigate broader graph learning tasks, additional fairness notions, and more diverse real-world deployment scenarios.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Please refer to the abstract and Section 1

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please refer to Appendix N for our discussions on the limitations.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Please refer to Appendix B

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Please refer to Section 5

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The URL of our code and data is released in the Section 5. We also provide the corresponding code in the supplementary materials.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Please refer to Appendix H for experimental details.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: All experiments are independently repeated 20 times, and the performance improvements are statistically significant according to a t-test ($p < 0.05$).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to Appendix H and Appendix F for sufficient information on the computer resources and running time.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research in the paper does not violate the NeurIPS Code of Ethics

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Please refer to Appendix M

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: Since UFA is an adversarial attack framework, we acknowledge its potential dual-use risk. To mitigate this risk, we position the work as a benchmark and diagnostic tool for revealing fairness vulnerabilities in GNNs, rather than as a deployment-oriented attack system. All experiments are conducted on public benchmark datasets in controlled offline settings, without targeting real users or deployed systems. We also discuss potential defense directions in Appendix K, including detection and robust fairness-aware training strategies, to help practitioners improve the resilience of GNN models against such attacks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All models and baselines from existing assets are properly cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We thoroughly discuss the details in this paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.