CROSS-ENCODER FAIRNESS DEGRADATION IN TEXT-ATTRIBUTED GRAPHS VIA ADVERSARIAL NODE INJECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Neural Networks (GNNs) have emerged as the dominant paradigm for modeling graph-structured data. Despite their wide adoption, recent studies reveal that GNNs are vulnerable to adversarial perturbations, particularly fairness attacks. These attacks can induce discriminatory outcomes and cause severe societal consequences in human-centered applications. Consequently, existing research has investigated fairness attacks targeting GNNs on traditional graphs to expose potential risks and ultimately guide the design of more robust systems. However, to the best of our knowledge, no existing fairness attack methods target GNNs processing Text-Attributed Graphs (TAGs), even though TAGs are ubiquitous and critically important in semantically rich domains like social networks. As a result, the community currently lacks a fundamental understanding of the fairness vulnerabilities of GNNs when processing TAGs. To bridge this gap, we propose a Universal Fairness Poisoning Attack against GNNs on TAG (UFPAT), which induces fairness-bias in the victim GNN by injecting malicious nodes. UFPAT achieves the first attack of generalizing across different text encoders. It leverages Large Language Models to refine the textual features of malicious nodes, ensuring high readability for real-world feasibility. We benchmark UFPAT against SOTA methods on three real-world TAG datasets. Results show that UFPAT is the only method capable of launching fairness poisoning attacks targeting GNNs on TAGs while generating readable text. It generalizes cross-text-encoders and attacks both standard and fairness-aware GNNs. This work reveals key fairness vulnerabilities in GNNs for TAGs and offers insights for improving robustness.

1 INTRODUCTION

Benefiting from their strong capacity to encode graph-structured representations, GNNs can efficiently process graph data Bruna et al. (2013) and have achieved notable success in recommender systems Chen et al. (2024), molecular simulation Sypetkowski et al. (2024), and social networks Zhang et al. (2024b). However, studies have shown that GNNs are vulnerable to adversarial attacks. Such attacks may primarily target model utility by degrading overall predictive performance, such as reducing classification accuracy Zügner et al. (2018). More critically, GNNs can also suffer from fairness attacks Wang et al. (2025), where adversaries manipulate graph structure or node attributes to induce systematically biased predictions against specific demographic groups, such as nationality, gender, or race. Unlike utility-oriented attacks, which mainly affect aggregate performance, fairness attacks selectively amplify disparities across sensitive groups, causing discriminatory outcomes even when overall accuracy remains largely unaffected. This makes fairness attacks dangerous, as they can remain concealed under seemingly acceptable utility while harming vulnerable populations.

With advances in NLP, graph learning has evolved from traditional graphs to text-attributed graphs (TAGs). Although TAGs have attracted considerable attention, their security remains less explored than that of traditional graphs. Compared with conventional graph data, TAGs differ substantially in both the semantic expressiveness of node features and attack constraints. **(1)** In traditional graphs, node features are typically generated by early representation learning methods such as GloVe Pennington et al. (2014) and BoW Salton et al. (1975). These fixed vectors have limited semantic capacity and thus struggle to fully characterize the complex semantics of nodes. In contrast, each node in

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

a TAG is usually associated with rich natural language text, which is encoded into high-dimensional node representations through a text encoder, substantially enhancing feature expressiveness. This enables graph neural networks to more naturally integrate textual semantics with graph structure during message passing, leading to stronger semantic understanding Yang et al. (2021); Yan et al. (2023). Moreover, **(2)** traditional graphs and TAGs also differ fundamentally in robustness and attack modeling. In traditional graphs, node features can often be directly manipulated, and attackers need not consider whether injected malicious node features are interpretable or readable. However, in TAGs, node features are tightly coupled with their textual content, making direct optimization of embedded representations difficult. Attackers must instead influence node features indirectly by generating or modifying natural language text. Meanwhile, the generated text must preserve semantic coherence and readability to avoid human inspection or automatic detection. Therefore, TAGs provide richer semantic information while imposing stricter constraints on feature perturbation and malicious node injection. Although recent studies Lei et al. (2024); Lyu et al. (2025); Fan et al. (2025) show that GNNs on TAGs can be vulnerable to attacks, these efforts still have substantial limitations, leaving several unresolved issues:

- $\mathcal{Q}1$: **Absence of fairness attacks**: Existing attacks on GNNs for TAGs focus exclusively on utility degradation (e.g., reducing accuracy), completely neglecting fairness attack scenarios. Unlike utility attacks, fairness attacks induce biased predictions toward specific node groups while maintaining overall utility Hussain et al. (2022). As previously discussed, fairness attacks can inflict more severe and insidious harm than utility attacks, as discriminatory outcomes may remain undetected under seemingly normal performance metrics, yet cause significant damage to vulnerable populations.
- $\mathcal{Q}2$: **Challenge of cross-encoder transferability**: Existing attack methods on TAGs lack cross-encoder capability, which significantly limits their applicability in real-world scenarios Lei et al. (2024). In practice, adversaries typically do not have access to the victim model's training pipeline, and distinct language models produce vastly different feature embeddings. Existing attacks, which often overfit specific textual embeddings, struggle to remain effective when transferred to models relying on different encoders. Meanwhile, most existing fairness attack methods Zhang et al. (2024a); J et al. (2024); Luo et al. (2024b) developed for conventional graphs require carefully crafting the features of malicious nodes. Such high-dimensional feature optimization makes it difficult to preserve the readability and semantic coherence of the corresponding node texts, **which constitutes a fundamental distinction between conventional graphs and TAGs**.
- $\mathcal{Q}3$: **High dependency on textual feature engineering**: Existing attack strategies impose substantial computational burdens due to intricate manipulation of textual features. These methods often require carefully crafted prompts and complex evaluation metrics tailored to LLMs Lyu et al. (2025), or training of specialized language models Fan et al. (2025). Such requirements increase implementation costs and limit scalability. Efficient attack frameworks capable of effective poisoning without expensive text generation remain lacking.

1.1 Technical Overview

To address the challenges above, we propose a Universal Fairness Poisoning Attack against GNNs on TAG (UFPAT). Specifically, for $\mathcal{Q}$**1**, we design a malicious node injection strategy that injects a small fraction of malicious nodes (less than 1% of the total) through perturbations at both the structural and feature levels. Furthermore, to address the multiple constraints associated with fairness issues, UFPAT proposes a unified objective function that integrates fairness metrics (Demographic Parity and Equal Opportunity Dwork et al. (2012); Hardt et al. (2016)), utility constraints, and sparsity constraints into a cohesive optimization framework, aiming to degrade GNN fairness while minimizing the impact on model utility.

To circumvent the challenges in $\mathcal{Q}$**2** and $\mathcal{Q}$**3**, UFPAT features an innovative end-to-end framework integrating LLM text generation with Relaxed Bernoulli soft-edge bi-level optimization. The process begins by using an LLM to generate readable, semantically meaningful text features for malicious nodes. We then parameterize the discrete injection edges using a Relaxed Bernoulli distribution and employ bi-level optimization to alternately train the shadow model and refine the edge weights, which are finally discretized to yield the poisoned graph. A key advantage of this framework is its strong dependence on adjacency structure but weak dependence on the specific text features of malicious nodes, which act merely as semantic camouflage. This design cleverly bypasses the variations

caused by different text encoders, ensuring cross-encoder transferability. This weak dependency allows for the use of lightweight generation methods, thereby reducing computational overhead and implementation costs. Our contributions can be summarized as follows:

- To the best of our knowledge, UFPAT is the first method to perform fairness attacks against GNNs on TAGs. By injecting a small number of malicious nodes with human-readable textual features, UFPAT successfully induces biased and unfair predictions while minimizing utility degradation of the victim GNN.
- UFPAT achieves cross-text-encoder transferability for the first time in TAG adversarial learning. Unlike prior methods limited by specific embeddings, our poisoned data is optimized under one encoder and seamlessly transfers to victim GNNs with different encoders, improving practical feasibility in real-world scenarios.
- We conducted extensive experiments on three real-world TAG datasets, directly comparing UFPAT against SOTA methods. Results demonstrate that UFPAT is the only method capable of performing fairness poisoning attacks while simultaneously ensuring cross-text-encoder transferability and generating human-readable text.

2 Related Work

2.1 Adversarial Attack of GNNs based on TAG

In TAGs, node attributes are associated with semantically rich text Yan et al. (2023); He et al. (2023), enabling Graph Language Models to integrate textual content and graph structure in their data processing pipelines Yang et al. (2021). Existing studies show that GNNs on TAGs are vulnerable to adversarial attacks Liu et al. (2025); however, these works focus exclusively on utility degradation, such as accuracy reduction. Lei et al. (2024) pioneered utility attacks on GNNs for TAGs by proposing a word-frequency-based text node injection method, balancing attack performance and interpretability. Nevertheless, it is highly sensitive to text encoders and lacks cross-encoder generalization. Lyu et al. (2025) introduced a black-box universal attack framework for TAGs, using LLMs to generate interpretable text-level node attributes. Although practically feasible, this approach relies heavily on heuristic LLM-based text generation and incurs high costs. Fan et al. (2025) proposed an interpretable multi-dimensional graph attack that jointly perturbs graph structure and textual features. Despite these advances, existing attacks on GNNs for TAGs still face critical limitations. First, they lack cross-text-encoder transferability, becoming ineffective when the victim model uses an encoder different from the shadow model. Second, they depend heavily on textual feature engineering, requiring carefully crafted text generation with substantial overhead. Most importantly, they target only utility degradation while overlooking important fairness attack scenarios.

In addition, although neither Tao et al. (2021) nor Sun et al. (2020) is designed for TAG attacks, they are algorithmically closest to UFPAT. Tao et al. (2021) formulates node injection attacks as a sequential decision-making process and uses reinforcement learning to progressively select injected-node connections and labels. Sun et al. (2020) further models node injection as the joint optimization of injected node features and discrete injected edges, employing a Gumbel-Top-k approximation to select edges under a fixed budget. In contrast, UFPAT directly combines Relaxed Bernoulli with STE for edge-wise differentiable parameterized optimization over candidate injected edges. This makes UFPAT more suitable for end-to-end training than RL-based sequential search and Top-k-based global edge selection, while facilitating the joint modeling of attack effectiveness, sparsity, and stealthiness constraints.

2.2 Fairness-aware GNNs

Rahman et al. (2019) was the first to explore fairness from a graph data perspective, introducing FairWalk to mitigate bias via pre-training structural modifications. Following this, Dai & Wang (2021) identified that historical data biases and message-passing mechanisms contribute to discrimination in GNNs, proposing FairGNN to suppress such bias through adversarial training and sensitive attribute discriminators. Improving upon this, Wang et al. (2022) developed FairVGNN, which masks features highly correlated with sensitive attributes to forestall bias generation during message propagation. Meanwhile, FairSIN Yang et al. (2024) employed a neutralization-based paradigm, incorporating fairness-enhancing components and non-sensitive information into node representations before message passing. Recently, attention has turned to fairness in TAGs. The only study

Luo et al. (2025) introduced FairTAG, which alleviates language model overfitting to homophily in downstream fine-tuning and then optimizes connectivity preferences at subgraph level for specific anchor nodes. This work marks the first realization of fairness awareness specifically for TAGs.

3 PRELIMINARY

We consider the standard node classification task on graph data. A Text-Attributed Graph (TAG) is formally defined as $\mathcal{G} = (V, E, T, Y)$, where V denotes the set of nodes and E denotes the set of edges, $|V| = n$ and $|E| = m$. Each node $v_i \in V$ is associated with a semantic text feature $t_i \in T$ and a ground-truth label $y_i \in Y$. The adjacency matrix of $\mathcal{G}$ is given by $\mathcal{A} \in \{0,1\}^{n\times n}$, where for any node pair $\{u, v\} \in V, \mathcal{A}_{uv} = 1$ if $e_{uv} \in E$, and $\mathcal{A}_{uv} = 0$ otherwise. Let $\mathbf{M}(\cdot, \theta)$ be a GNN model parameterized by θ. The objective of the TAG-based node classification task is to embed T into a feature matrix X using a pre-trained text encoder $\mathcal{C}(\cdot)$, and to train θ such that the predicted probability distribution $\hat{Y}$ approximates the ground-truth.

3.1 DEFINITION OF FAIRNESS METRICS

Fairness metrics are utilized to quantify the degree of fairness in model predictions. Following prior studies Ling et al. (2023); Dai & Wang (2021); Zhang et al. (2024a); Luo et al. (2024b), this paper focuses on group fairness in binary classification with binary sensitive attributes. Let each node $v_i \in V$ in $\mathcal{G}$ be associated with a sensitive attribute $s_i \in \{0, 1\}$, and let $S = (s_1, \ldots, s_n)$ denote the sensitive attribute vector of all nodes. Let $\tilde{y}$ denote the binary prediction of the classifier. Two widely used group fairness metrics are considered:

- **Equal Opportunity (EO)** Hardt et al. (2016) requires the true positive rates to be equal across sensitive groups:
$$P(\tilde{y} = 1 \mid s = 1, y = 1) = P(\tilde{y} = 1 \mid s = 0, y = 1).$$
- **Demographic Parity (DP)** Dwork et al. (2012) requires the overall positive prediction rates to be identical across groups:
$$P(\tilde{y} = 1 \mid s = 1) = P(\tilde{y} = 1 \mid s = 0).$$

Based on the above definitions, we quantify fairness disparities using the following metrics, where smaller values indicate better group fairness:

$$\Delta_{EO} = |P(\tilde{y}=1|s=1, y=1) - P(\tilde{y}=1|s=0, y=1)|$$
$$\Delta_{DP} = |P(\tilde{y}=1|s=1) - P(\tilde{y}=1|s=0)|$$

3.2 PROBLEM FORMULATION

In a fairness poisoning attack on TAG-based GNNs, the attacker maliciously modifies the training data under specific constraints to induce significantly increased prediction bias. Specifically, for a given $\mathcal{G}$, UFPAT injects a set of malicious nodes $v^* \in V^*$ with semantically meaningful text features T^*, where $|V^*| < b$ denotes the attack budget. To preserve semantic plausibility, the injected texts are constrained to have high cosine similarity with embeddings of existing nodes. Under strict access permission constraints, UFPAT does not modify existing labels, features, topological structures, or train/test splits of $\mathcal{G}$. Moreover, V^* is excluded from both loss computation and fairness evaluation to avoid artificially inflating performance metrics. For an original GNN $\mathcal{M}(\cdot)$ with parameters θ, we define the fairness poisoning attack on TAG as follows:

$$V^X_{train}, V^X_{test} = \mathcal{C}(V^T_{train}, V^T_{test}), \quad \theta^* = \arg\min_{\theta} \mathcal{L}_o(\mathcal{M}(\theta; \mathcal{A}^*, G^*, V^X_{train}), Y),$$
$$\max_{\mathcal{G}^*} \mathcal{L}_{\Delta_{EO}, \Delta_{DP}}(\mathcal{M}(\theta^*; \mathcal{A}^*, V^X_{test}), Y, S), \quad s.t.\ \mathcal{L}_{CE}(\theta^*) < \mathcal{L}_{CE}(\theta) + \epsilon$$

where $\mathcal{G}^*$ denotes the poisoned TAG, $\mathcal{A}^*$ is its adjacency matrix, and $\mathcal{L}_o$ denotes the training loss. The attacker may have access to only partial information about the node features and distribution of V^T_{train}. During training, text features V^T_{train} and V^T_{test} are embedded into feature vectors V^X_{train} and V^X_{test} via the pre-trained encoder $\mathcal{C}(\cdot)$, after which the GNN optimizes the poisoned parameters θ^* based on $\mathcal{G}^*$. During inference, maximizing the fairness loss $\mathcal{L}_{\Delta_{EO}, \Delta_{DP}}$ on the nodes exacerbates fairness bias, while minimizing the utility loss $\mathcal{L}_{CE}$ with respect to the original labels keeps accuracy close to that before the fairness attack. The attack setting will be shown in Appendix A.

4 METHODOLOGY

The UFPAT framework, illustrated in Figure 1, consists of two stages. In the malicious node textual feature generation stage, UFPAT randomly assigns each malicious node to a category group and employs an LLM to generate textual features via contrastive sampling of positive and negative examples. In the malicious node connection modification stage, a dual-layer, multi-step structural optimization framework integrating relaxed Bernoulli sampling and the straight-through estimator is used to manipulate connections between malicious and original nodes, thereby amplifying inter-group disparities and undermining fairness.

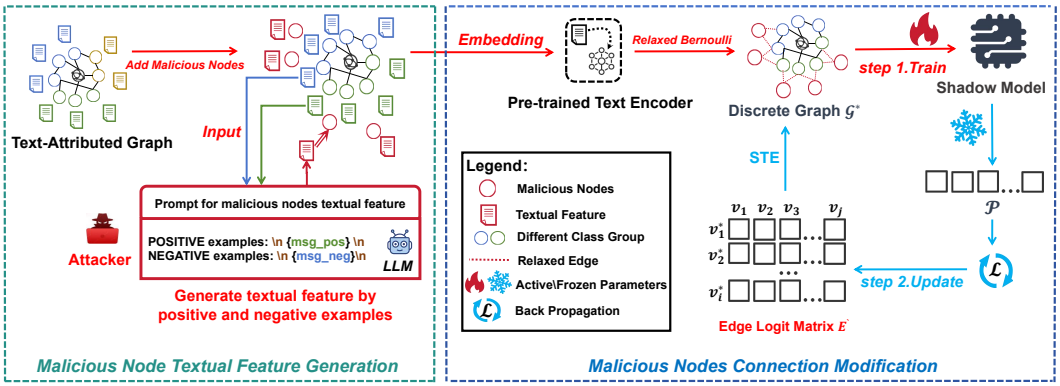


Figure 1: Framework of the UFPAT Algorithm

4.1 MALICIOUS NODES TEXTUAL FEATURE GENERATION

In this phase, UFPAT injects malicious nodes $v^* \in V^*$ into semantic regions corresponding to different classes to generate textual features T^* with semantic information. First, UFPAT randomly and uniformly distributes b malicious nodes across each class group $y \in Y$ in $\mathcal{G}$. As UFPAT aims to degrade model fairness on clean nodes, these malicious nodes act solely as sources of structural and feature-level perturbations, and are thus assigned labels $y^* = -1$. During training and evaluation, a masking mechanism excludes these nodes from loss computation and fairness assessment, preventing them from directly influencing performance metrics.

The algorithm then invokes LLM API to generate semantically meaningful textual features t^*. Specifically, for each malicious node v^*, UFPAT randomly samples h node textual features from the semantic region corresponding to the class of v^* to form a positive sample set $\mathcal{H}_\mathbf{p}$, while sampling textual features from h random nodes belonging to a different class to constitute a negative set $\mathcal{H}_\mathbf{n}$. Subsequently, the LLM is conditioned on a constrained text generation task: to generate text that aligns as closely as possible with the positive set in terms of semantics and topic, while maximizing the semantic divergence from the negative set. The length and syntactic structure of the generated text are constrained to resemble the statistical properties of the provided exemplars. This design effectively constrains the semantic space of the generated text, enabling it to maintain consistency. The specific **Prompt** design is illustrated in Figure 2.

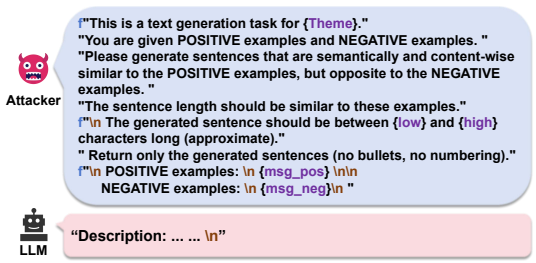


Figure 2: The **Prompt** for malicious nodes textual feature

The process generates c candidate textual features $\{\hat{t}\}^c$ for each malicious node. To select the text exhibiting the strongest semantic consistency, the algorithm employs an embedding-based similarity metric to evaluate the candidate sentences. UFPAT utilizes a pre-trained ***sentence-transformers*** model $\mathcal{C}(\cdot)$ to encode both the candidate texts and the positive example texts into semantic vectors. It then calculates the average cosine similarity between each candidate and the positive set as a

scoring function:

$$\mathcal{S}_{cos}(\hat{t}, \mathcal{H}_{\mathbf{p}}) = \frac{1}{|\mathcal{H}_{\mathbf{p}}|} \sum_{\mathbf{p} \in \mathcal{H}_{\mathbf{p}}} \frac{\mathcal{C}(\hat{t})^{\top} \mathcal{C}(\mathbf{p})}{\|\mathcal{C}(\hat{t})\|_2 \, \|\mathcal{C}(\mathbf{p})\|_2}$$

where $\mathbf{p}$ denotes a positive example text in the set $\mathcal{H}_{\mathbf{p}}$, The algorithm selects the candidate text $\hat{t}$ with the highest score $\mathcal{S}_{cos}$ as the textual feature t^* for each v^*. The detailed procedure is shown in Algorithm 2 in Appendix B.

4.2 Malicious Nodes Connection Modification

In this phase, UFPAT leverages a shadow model to establish connectivity between V^* and $\mathcal{G}$, aiming to execute a fairness poisoning attack that induces bias across sensitive attribute groups in the victim model. Specifically, we reformulate the originally discrete, multi-constrained edge selection problem into a differentiable optimization of injected edge weights within a continuous relaxed space. This enables the end-to-end joint optimization of the connection structure during the graph message-passing process.

First, while strictly preserving the original graph structure, UFPAT introduces learnable edge existence variables for potential connections between malicious node and original nodes. By adopting node-wise Relaxed Bernoulli parameterization, the discrete selection of injected edges is continuously relaxed into differentiable optimization variables:

$$\tilde{e}_{ij} = \mathrm{Sigmoid}\left(\frac{e'_{ij} + \log u_{ij} - \log(1 - u_{ij})}{\rho}\right)$$

where e'_{ij} denotes the learnable logit parameter for the injected edge between the i-th malicious node and the j-th original node; $u_{ij} = \mathrm{Uniform}(0, 1)$ represents a random variable sampled from a uniform distribution; and ρ is the temperature parameter controlling the degree of relaxation. $\tilde{e}_{ij}$ represents the continuous edge weight obtained via Relaxed Bernoulli sampling, serving as the optimization variable to approximate the discrete edge existence variable.

Then, we employ the Straight-Through Estimator (STE) to discretize the continuously relaxed edge weights. During the forward propagation, hard edge selections are obtained via a thresholding operation. During the backward propagation, gradients are propagated through the continuous relaxation variables, thereby enabling the end-to-end optimization of the discrete graph structure:

$$\hat{e}_{ij} = \left(\mathbb{I}[\tilde{e}_{ij} > \delta] - \tilde{e}_{ij}\right)^{\mathrm{stop}} + \tilde{e}_{ij}$$

where $\tilde{e}_{ij}$ represents the continuous edge weight obtained via Relaxed Bernoulli sampling, serving as the optimization variable for backpropagation, $\hat{e}_{ij}$ denotes the hard edge employed during the forward pass. $\mathbb{I}[\tilde{e}_{ij} > \delta]$ is the indicator function that returns 1 if $\tilde{e}_{ij} > \delta$, and 0 otherwise. $(\cdot)^{\mathrm{stop}}$ denotes the stop-gradient operator. It is employed to detach the gradient history from specific terms, while the re-addition of $\tilde{e}_{ij}$ ensures that gradients are effectively propagated through the continuous variables during optimization.

We employ a unified loss function to jointly optimize the fairness loss, utility constraints, and sparsity constraints. This formulation aims to degrade GNN fairness while minimizing the adverse impact on model utility to the extent possible. Specifically, the fairness loss $\mathcal{L}_{fair} = \mathcal{L}_{\Delta_{EO}} + \mathcal{L}_{\Delta_{DP}}$ quantifies the statistical discrepancy in prediction outcomes across distinct sensitive groups:

$$\mathcal{L}_{\Delta_{EO}} = || \sum_{c \in Y} \left(\frac{1}{|V_{S_0}^{(c)}|} \sum_{v \in V_{S_0}^{(c)}} \tilde{y}_v^{(c)} - \frac{1}{|V_{S_1}^{(c)}|} \sum_{v \in V_{S_1}^{(c)}} \tilde{y}_v^{(c)} \right) ||_2^2,$$

$$\mathcal{L}_{\Delta_{DP}} = || \frac{1}{|V_{S_0}|} \sum_{v \in V_{S_0}} \tilde{y}_v - \frac{1}{|V_{S_1}|} \sum_{v \in V_{S_1}} \tilde{y}_v ||_2^2$$

where $S = \{0, 1\}$ is sensitive attribute, $\tilde{y}_v$ denotes the prediction probability of the GNN for node v, and $\tilde{y}_v^{(c)}$ represents the predicted probability of node v for class c. Y denotes the set of all classes. $V_{s_i}^{(c)}$ refers to the set of nodes whose sensitive attribute is $i \in \{1, 0\}$ and label is c. To ensure the poisoning attack does not significantly compromise the victim model's utility performance – a

requirement for maintaining attack stealthiness – we design the utility preservation loss as follows:

$$\mathcal{L}_U = -\frac{1}{|V|}\sum_{v\in V} y_v \cdot \log(\tilde{y}_v)$$

where y_v denotes the true label of node v, and $\tilde{y}_v$ denotes the prediction probability of the GNN for node v. Additionally, we incorporate a sparsity regularization term that approximates the expected number of injected edges, thereby imposing a soft constraint on the attack budget. This term is defined as: $\mathcal{L}_{\text{sparse}} = \sum_{i,j} \tilde{e}_{ij}$, where $\tilde{e}_{ij}$ denotes the continuous relaxation of the edge existence variable, representing the probability that an edge (i, j) is injected in the attacked graph $\mathcal{G}^*$. Ultimately, the overall loss function is formulated as: $\mathcal{L} = -\mathcal{L}_{fair} + \beta \cdot \mathcal{L}_U + \alpha \cdot \mathcal{L}_{\text{sparse}}$, where α and β are hyperparameter. By jointly optimizing the fairness loss, utility loss, and the sparsity regularization term, UFPAT achieves a cross-text-encoder fairness attack against TAG.

4.3 Overall Design of UFPAT

In the gray-box setting, the attacker has no access to the victim model's text encoder, parameters, or gradients. To address this, we use a bi-level multi-step optimization framework, where the first level trains a shadow model $\mathcal{M}_s(\cdot)$, and the second level optimizes connections between malicious and original nodes. For node injection attacks, we further adopt a node-wise Relaxed Bernoulli formulation with the STE, enabling hard edge selection in the forward pass and continuous relaxation for gradient propagation in the backward pass, thereby achieving end-to-end optimization of the discrete graph.

Algorithm 1 UFPAT

Input: TAG $\mathcal{G} = (V, E, T, Y)$;
Parameter: Shadow model $\mathcal{M}_s(\theta_s; \cdot)$; Loss function of shadow model and connection modification $\mathcal{L}_s$, $\mathcal{L}$; Train epoch of shadow model and connection modification T_s, T_c; Epoch of bi-level iteration T_e; Learning rate of shadow model and connection modification ℓ_s, ℓ_c; Parameter ρ, δ;
Output: Poisoned TAG $\mathcal{G}^*$;

1: Generate malicious node set V^* of $\mathcal{G}^*$ by **Algorithm 2**.
2: Initialize a learnable edge logit matrix $E' \in \mathbb{R}^{b\times n}$, where e'_{ij} represents the logit associated with a potential injected edge between $v_i^* \in V^*$ and $v_j \in V$.
3: **for** 0, 1, ..., T_e **do**
4: Sample continuously relaxed edge weights $\tilde{E} = \{\tilde{e}_{ij}\}$ from a Relaxed Bernoulli distribution parameterized by E':
$\tilde{e}_{ij} = \text{Sigmoid}\Big(\frac{e'_{ij}+\log u_{ij}-\log(1-u_{ij})}{\rho}\Big)$,
5: Obtain hard injected edges $\hat{E}$ via STE:
$\hat{e}_{ij} = \big(\mathbb{I}[\tilde{e}_{ij} > \delta] - \tilde{e}_{ij}\big)^{\text{stop}} + \tilde{e}_{ij}$.
6: Construct discrete graph $\mathcal{G}^*$ with V^* and $\hat{E}$.
7: **for** 0, 1, ..., T_s **do**
8: $\theta_s \leftarrow \theta_s - \ell_s \cdot \nabla_{\theta_s}\mathcal{L}_s(\mathcal{G}^*)$;
9: **end for**
10: **for** 0, 1, ..., T_c **do**
11: $\mathcal{P} = \text{softmax}(\mathcal{M}_s(\theta_s; \mathcal{G}^*))$.
12: $E' \leftarrow E' - \ell_c \cdot \nabla_{E'}\mathcal{L}(\mathcal{P})$;
13: **end for**
14: **end for**
15: Deterministically discretizing the learned edge logits E': $\hat{e}_{ij} = \mathbb{I}[\text{Sigmoid}(e'_{ij}) > \delta]$.
16: **return** $\mathcal{G}^*$

As shown in Algorithm 1, Lines 1-2 generate the set of malicious nodes $|V^*| = b$ and initialize a learnable parameter vector E' for injected edge logits. Using the Relaxed Bernoulli distribution and STE, the discrete and multi-constrained edge selection problem is reformulated as a differentiable optimization problem in a continuous relaxed space (Lines 4–6). The STE ensures discrete injected edge structures during the forward pass while allowing gradient backpropagation to the constant parameter space during the backward pass. Lines 7–9 train the shadow model on the poisoned graph, and Lines 10–13 update the injected edge logits based on the shadow model's prediction

probability vector $\mathcal{P}$. Finally, discretizing the learned edge logits produces the final set of injected edges, constructing poisoned graph $\mathcal{G}^*$.

5 EXPERIMENTS

In this section, we evaluate UFPAT against SOTA baselines on real-world datasets of varying scales. We comprehensively assess its fairness poisoning effectiveness, robustness across diverse GNN architectures, and cross-encoder transferability. We further provide supplementary experiments, including cross-text-encoder generalization, ablation studies, analyses of different LLMs as semantic generators, LLM cost analysis, and readability assessment of generated malicious texts. These results collectively demonstrate the superiority of UFPAT over existing approaches. Finally, we discuss potential future defense strategies. The code is available at an anonymous link: `https://anonymous.4open.science/r/ufpat-6B86`

5.1 EXPERIMENTAL SETTINGS

Datasets: We strictly follow the dataset settings adopted in prior work Hussain et al. (2022); Zhang et al. (2024a); Luo et al. (2024b) and conduct experiments on three commonly used real-world datasets: **PubMed** Yang et al. (2016), **Pokec-Z**, and **Pokec-N** Takac & Zabovsky (2012). These datasets integrate text-annotated data from multiple domains and sensitive attributes. In each dataset, only a subset of nodes is labeled; we randomly split labeled nodes into 50% for training, 25% for validation, 25% for testing. Appendix H considers a weaker threat model in which the attacker can observe the class labels, sensitive attributes, and textual features of only a subset of nodes.

Baselines: Since UFPAT is the first method to perform fairness attacks on GNNs on TAGs, we adopt two categories of baselines for a fair and comprehensive comparison: classical fairness attacks and utility attacks on TAGs. The classical fairness attack baselines include **FA-GNN** Hussain et al. (2022), **G-FairAttack** Zhang et al. (2024a), **FATE** J et al. (2024), and **NIFA** Luo et al. (2024b). The TAG utility attack baselines include **WTGIA** Lei et al. (2024), and **ATAG-LLM** Lyu et al. (2025); we adapt these methods by replacing their objective functions with fairness objectives. For baseline classical fairness attacks, we utilize encoder to transform text features into node embeddings, and we impose no semantic constraints on the features of the malicious nodes.

Victim Models: We compare the algorithms on different GNNs as victim models, including classical GNNs (**GCN** Kipf & Welling (2017), **GraphSAGE** Hamilton et al. (2017), **GAT** Velickovic et al. (2018), and **GIN** Xu et al. (2019)) and fairness-aware GNNs (**FairGNN** Dai & Wang (2021), **FairVGNN** Wang et al. (2022)), and **FairSIN** Yang et al. (2024)). And a SOTA fairness GNN model based on TAG, **FairTAG** Luo et al. (2025). For the GNNs not based on TAG, we refer to the method in Luo et al. (2024a), directly encoding attributes using a pre-trained text encoder and inputting the embedding vectors into the GNN.

LLM and Text Encoder: In UFPAT, the LLM used for generating malicious node text features is the **GPT-5 mini** OpenAI (2025) API. For raw textual attributes across all datasets, we use high-quality pre-trained ***sentence-transformers*** models fine-tuned for sentence embedding. Specifically, **Paraphrase-MPNet-base-v2** Song et al. (2020) is adopted as the primary text encoder to generate feature vectors for all experiments. The ablation studies for the encoder and LMM will be provided in Appendix I and L.

Evaluation Metrics and Parameter Settings: We adopt the fairness metrics Δ_{EO} and Δ_{DP} from Section 3.2, where larger values indicate greater fairness bias. As the utility-loss metric, we use the overall classification accuracy ACC Zhang et al. (2024a); Luo et al. (2024b). A larger deviation between the algorithm's ACC and clean accuracy indicates stronger utility degradation, making the attack more detectable. All experiments are independently repeated 20 times, and the performance improvements are statistically significant according to a t-test ($p < 0.05$). Appendix C provides hyperparameter details.

5.2 EFFECTIVENESS OF ATTACK

Table 1 reports the results of fairness poisoning attacks on classical GNNs. Compared with SOTA baselines, our method consistently achieves superior attack performance across datasets. Specifically, it induces substantial increases in Δ_{EO} and Δ_{DP} compared with existing fairness poisoning schemes. The suboptimal performance of traditional baselines, including NIFA, FATE, G-

Table 1: Attack performance of our method and SOTA baselines on fairness poisoning attacks against classical 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			PubMed		
		ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
GCN	Clean	71.01	5.27	7.46	70.42	1.53	0.93	92.68	3.36	4.12
	FA-GNN	70.02	5.63	7.81	70.15	2.46	2.17	90.59	3.84	4.91
	G-FairAttack	-	-	-	-	-	-	91.99	4.22	5.35
	FATE	-	-	-	-	-	-	91.44	3.48	4.69
	NIFA	70.64	8.84	10.86	69.29	3.37	3.14	90.51	6.61	9.10
	WTGIA	70.54	4.97	6.53	68.98	3.01	3.15	90.37	5.85	4.67
	ATAG-LLM	69.92	4.25	4.47	67.05	2.89	3.62	90.47	4.26	5.63
	Ours	70.33	**12.85**	**14.71**	69.60	**6.51**	**6.83**	89.02	**11.32**	**10.60**
GraphSAGE	Clean	70.56	3.60	5.46	67.90	2.70	2.12	93.88	1.31	3.06
	FA-GNN	69.60	3.46	5.15	66.32	3.84	3.05	93.62	1.44	2.38
	G-FairAttack	-	-	-	-	-	-	92.05	1.38	2.66
	FATE	-	-	-	-	-	-	92.42	1.53	2.58
	NIFA	69.73	6.92	8.78	67.11	2.91	2.83	93.07	1.49	3.09
	WTGIA	70.52	4.86	3.47	66.14	2.15	2.26	93.77	2.46	3.02
	ATAG-LLM	68.34	5.27	5.83	66.22	2.81	3.68	93.74	2.83	4.15
	Ours	69.93	**9.31**	**11.25**	66.14	**3.99**	**3.63**	93.44	**3.16**	**4.83**
GAT	Clean	70.37	4.03	5.66	65.97	2.91	2.90	91.73	3.49	3.71
	FA-GNN	69.01	4.90	5.77	65.35	3.23	3.61	89.67	3.14	3.90
	G-FairAttack	-	-	-	-	-	-	91.02	3.84	3.98
	FATE	-	-	-	-	-	-	91.15	3.50	4.37
	NIFA	69.46	7.65	9.11	63.94	4.54	3.33	91.15	3.53	4.14
	WTGIA	70.44	4.52	5.43	63.88	3.70	3.24	91.74	3.42	3.30
	ATAG-LLM	67.56	3.70	3.94	63.55	3.37	4.89	91.07	3.59	4.83
	Ours	68.72	**8.04**	**9.74**	62.24	**6.57**	**6.22**	90.02	**5.93**	**7.41**
GIN	Clean	65.08	2.88	1.71	63.20	2.73	2.32	92.06	2.81	3.26
	FA-GNN	63.77	2.59	2.60	62.28	3.01	2.47	91.59	2.12	3.01
	G-FairAttack	-	-	-	-	-	-	92.89	3.08	4.68
	FATE	-	-	-	-	-	-	91.20	2.65	2.68
	NIFA	64.51	3.84	3.43	62.76	3.33	3.59	92.85	2.94	4.42
	WTGIA	64.81	3.49	3.19	62.43	3.08	4.11	91.83	3.87	4.12
	ATAG-LLM	64.02	3.76	4.59	61.80	3.45	4.06	91.27	3.63	4.29
	Ours	64.77	**11.23**	**11.93**	62.66	**8.21**	**8.20**	90.57	**7.21**	**8.28**

Table 2: Attack performance of our method and SOTA baselines on fairness poisoning attacks against fairness-aware 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			PubMed		
		ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
FairGNN	Clean	71.64	5.32	7.37	70.42	1.77	1.25	92.25	3.14	4.94
	FA-GNN	70.32	6.16	7.11	70.20	2.12	2.56	91.22	3.89	4.46
	G-FairAttack	-	-	-	-	-	-	91.75	3.39	4.75
	FATE	-	-	-	-	-	-	91.53	4.30	5.19
	NIFA	70.10	8.28	9.06	70.56	2.40	3.66	90.77	4.56	5.85
	WTGIA	70.53	6.30	7.19	70.44	3.07	2.18	90.51	3.83	3.20
	ATAG-LLM	70.99	5.16	6.17	69.99	3.88	1.56	90.42	3.19	5.87
	Ours	70.47	**12.71**	**13.67**	69.35	**6.79**	**7.58**	90.05	**10.37**	**9.19**
FairVGNN	Clean	71.84	5.21	7.22	70.74	2.66	1.54	92.34	3.75	4.35
	FA-GNN	70.16	5.62	6.81	70.10	2.75	1.46	90.75	4.15	5.80
	G-FairAttack	-	-	-	-	-	-	90.83	5.74	5.30
	FATE	-	-	-	-	-	-	91.76	5.14	6.50
	NIFA	69.22	7.86	9.81	69.19	3.69	2.51	90.86	5.50	8.23
	WTGIA	69.04	4.11	7.10	70.98	2.29	1.20	91.80	3.11	4.49
	ATAG-LLM	69.42	5.96	6.92	70.17	1.74	1.38	92.27	3.68	4.72
	Ours	70.15	**11.85**	**11.40**	70.60	**5.24**	**4.40**	90.53	**10.71**	**12.16**
FairSIN	Clean	70.97	4.37	6.22	71.38	1.67	1.08	93.19	2.21	4.02
	FA-GNN	70.56	4.73	6.63	69.68	2.04	3.13	91.57	3.54	3.66
	G-FairAttack	-	-	-	-	-	-	90.33	3.83	4.64
	FATE	-	-	-	-	-	-	90.72	3.56	3.85
	NIFA	70.32	7.06	8.59	69.26	3.82	3.31	91.07	5.34	**9.74**
	WTGIA	70.88	4.00	6.40	71.32	1.66	2.95	92.54	2.99	5.01
	ATAG-LLM	71.29	4.91	6.62	70.25	2.89	2.81	91.86	3.24	4.40
	Ours	70.97	**9.15**	**9.41**	69.34	**5.15**	**6.23**	89.17	**8.44**	7.53
FairTAG	clean	71.50	4.31	7.51	70.53	2.88	1.24	91.17	2.90	3.50
	WTGIA	70.46	4.34	8.19	70.11	3.44	4.64	91.12	4.86	3.94
	ATAG-LLM	71.33	5.56	6.39	70.76	4.16	4.45	91.68	4.39	3.70
	Ours	70.19	**11.13**	**12.52**	69.54	**5.75**	**5.84**	90.25	**8.67**	**8.52**

FairAttack, and FA-GNN, stems from fundamental limitations in their attack assumptions. These methods typically optimize edge perturbations or low-dimensional features of injected nodes, while neglecting the intricate coupling between graph structure and high-dimensional features generated by language models. Such strategies are effectively mitigated by the semantic robustness of pre-trained encoders and the message-passing mechanism of GNNs, where high-dimensional textual attributes dominate aggregation and dilute the attack signal. Consequently, adversarial perturbations are suppressed by both the semantic consistency of language models and the regularization effects of GNNs. Due to scalability issues, FATE and G-FairAttack both took over 48 hours to run on two large graph. Moreover, TAG-specific attacks, such as WTGIA and ATAG-LLM, fail to induce unfairness. Table 2 reports the results of fairness poisoning attacks against fairness-aware GNN models. Compared with SOTA baselines, our method again achieves superior attack performance. Whether the target is a non-TAG-based fairness-aware GNN, such as FairGNN, FairVGNN, and FairSIN, or a TAG-based fairness-aware GNN trained through an end-to-end unified framework, such as FairTAG, our method successfully performs fairness poisoning.

AI USE STATEMENT

Generative AI tools were used to polish the writing of the manuscript and to provide limited assistance with software implementation. All AI-assisted text and code were reviewed, edited, and verified by the authors.

REFERENCES

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

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)*, pp. 1274–1284, 2024.

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)*, pp. 680–688, 2021.

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*, pp. 214–226, 2012.

Bowen Fan, Zhilin Guo, Xunkai Li, Yihan Zhou, Bing Zhou, Zhenjun Li, Rong-Hua Li, and Guoren Wang. Unveiling the vulnerability of graph-llms: An interpretable multi-dimensional adversarial attack on tags. *arXiv preprint arXiv:2510.12233*, 2025.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Bryan Hooi, et al. Explanations as features: Llm-based features for text-attributed graphs. *arXiv preprint arXiv:2305.19523*, 2(4):8, 2023.

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)*, pp. 975–980. IEEE, 2022.

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.

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

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

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)*, pp. 2958–2976. IEEE, 2024.

Runlin Lei, Yuwei Hu, Yuchen Ren, and Zhewei Wei. Intruding with words: Towards understanding graph injection attacks at the text level. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:49214–49251, 2024.

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.

Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S Yu, et al. Graph foundation models: Concepts, opportunities and challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2025.

Ruilin Luo, Tianle Gu, Lin Wang, Yunfeng Zhou, Songtao Jiang, Lei Wang, and Yujiu Yang. Fair text-attributed graph representation learning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 14330–14353, 2025.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic gnns are strong baselines: Reassessing gnns for node classification. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:97650–97669, 2024a.

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)*, 2024b.

Yuefei Lyu, Chaozhuo Li, Xi Zhang, and Tianle Zhang. Navigating the black box: Leveraging llms for effective text-level graph injection attacks. *arXiv preprint arXiv:2506.13276*, 2025.

OpenAI. Introducing gpt-5. `https://openai.com/index/introducing-gpt-5/`, 2025.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

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

Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:16857–16867, 2020.

Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. Non-target-specific node injection attacks on graph neural networks: A hierarchical reinforcement learning approach. In *Proc. WWW*, volume 3, 2020.

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.

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.

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*, pp. 1794–1803, 2021.

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

Yifan Wang, Hourun Li, Ling Yue, Zhiping Xiao, Jia Yang, Changling Zhou, Wei Ju, Ming Zhang, and Xiao Luo. Dance: Dual unbiased expansion with group-acquired alignment for out-of-distribution graph fairness learning. In *Forty-second International Conference on Machine Learning (ICML)*, 2025.

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)*, pp. 1938–1948, 2022.

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

Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, et al. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. *Advances in Neural Information Processing Systems (NeurIPS)*, 36: 17238–17264, 2023.

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, pp. 9241–9249, 2024.

Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. Graphformers: Gnn-nested transformers for representation learning on textual graph. *Advances in Neural Information Processing Systems (NeurIPS)*, 34: 28798–28810, 2021.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning (ICML)*, pp. 40–48, 2016.

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)*, 2024a.

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)*, pp. 4223–4231, 2024b.

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

APPENDIX

A ATTACK SETTINGS

Threat Model: To ensure real-world feasibility, UFPAT is formulated as a gray-box attack. In this setting, the attacker has access to the graph structure $\mathcal{A}$, text features T, ground-truth labels Y, and sensitive attributes S. The attacker is aware only of the node set utilized during the training phase and remains oblivious to the specific partitioning of test nodes. Furthermore, the architecture $\mathcal{M}$, parameters θ, and text embedder $\mathbf{C}$ of the victim model remain concealed from the attacker. This setup constitutes a realistic and conservative threat model assumption. In addition, Appendix H considers a weaker threat model in which the attacker can observe the class labels, sensitive attributes, and textual features of only a subset of nodes. This setting is used to evaluate the attack effectiveness of UFPAT under more restrictive information-access conditions.

Attacker's capability: The attacker can generate malicious nodes and their associated textual features, and can control which existing nodes the malicious nodes connect to. To ensure a fair comparison, these malicious nodes are masked during evaluation and are excluded from the computation of all evaluation metrics. They participate only in graph propagation through their structural connections and node features. The attacker is allowed to configure a shadow model but has no knowledge of the architecture or parameters of either the text encoder or the target model.

B PSEUDOCODE

Due to space limitations, we provide the pseudocode for the malicious node textual feature generation in Algorithm 2. For each $v^* \in V^*$, the algorithm first assigns a semantic region (Line 3). Then, using v^*'s positive and negative sample sets, the LLM API is invoked to generate c candidate textual features (Lines 4-6). Finally, the algorithm calculates the semantic relevance scores for candidate textual features and selects the highest-scoring one as the textual feature for v^* (Lines 7-8).

Algorithm 2 Malicious Node Textual Feature Generation

Input: TAG $\mathcal{G} = (V, E, T, Y)$;
Parameter: LLM API $LLM(\cdot)$; Pre-trained sentence-transformers model $\mathcal{C}(\cdot)$; Attack budget b; Temperature parameter of diversity τ; Number of candidate c; Number of samples h; Prompt;
Output: Malicious nodes set v^* with textual features T^*;

1: Initializing Malicious nodes $|V^*| = b$;
2: **for** $v^* \in V^*$ **do**
3: Randomly assign v^* to semantic region corresponding $y \in Y$, $y^* = -1$;
4: Randomly select h textual features $\mathbf{p}$ from semantic region where v^* resides, add $\mathbf{p}$ to $\mathcal{H}_{\mathbf{p}}$;
5: Randomly select h textual features $\mathbf{n}$ from semantic region with different from v^*, add $\mathbf{n}$ to $\mathcal{H}_{\mathbf{n}}$;
6: $\{\hat{t}_i\}_{i=1}^{c} \leftarrow LLM(\mathrm{Prompt}, \mathcal{H}_{\mathbf{p}}, \mathcal{H}_{\mathbf{n}}, \tau)$
7: $\mathcal{S}_{cos}(\hat{t}, \mathcal{H}_{\mathbf{p}}) = \frac{1}{|\mathcal{H}_{\mathbf{p}}|} \sum_{\mathbf{p} \in \mathcal{H}_{\mathbf{p}}} \frac{\mathcal{C}(\hat{t})^\top \mathcal{C}(\mathbf{p})}{\|\mathcal{C}(\hat{t})\|_2 \, \|\mathcal{C}(\mathbf{p})\|_2}$;
8: $t^* = \arg\max_{\hat{t}} \mathcal{S}_{cos}(\hat{t}, \mathcal{H}_{\mathbf{p}})$;
9: **end for**
10: **return** V^*

C Parameter Setting Details

In this section, we give an introduction to the parameter setting details used in our experiment. The number of positive and negative samples is set to $h = 5$, the number of candidate textual features is $c = 5$, and LLM exploration temperature is set to $\tau = 0.25$. All experiments used a two-layer Weighted GCN as a shadow model with a hidden dimension of 128. We set $\rho = 0.8$ and $\delta = 0.5$. We set the learning rate of the shadow model and the connection modification to $\ell_s = 0.01$ and $\ell_c = 0.0001$. We set the train epoch of the shadow model and connection modification $T_s = 50$, $T_c = 50$, and the epoch of bi-level iteration $T_e = 10$. All optimizers were set to Adam Kingma (2014). For the Pokec-Z, Pokec-N, and PubMed dataset, we set the sparsity constraint parameter to $\alpha = 1e^{-7}$, $\alpha = 5e^{-8}$, and $\alpha = 1e^{-7}$, respectively, the utility constraint to $\beta = 0.1$, $\beta = 0.001$, and $\beta = 0.001$ and the attack budget to $b = 10$, $b = 20$, and $b = 75$, respectively. We obtain the above parameters through grid search by jointly considering accuracy and fairness metrics. All experiments are conducted on a Linux server with Intel Xeon Gold 5317 CPU @ 3.00GHz and NVIDIA A100-SXM4-40GB GPU, with consistent settings across all experiments (unless otherwise stated).

For the victim models, we adopt the standard implementations and default configurations provided by the Deep Graph Library (DGL). Specifically, GCN is implemented with a two-layer graph convolutional architecture, while GraphSAGE employs two layers of mean-aggregation SAGE convolutions. GAT is implemented with two GATConv layers using a single attention head per layer, while GIN employs one layer of convolutions. All these models are trained without dropout for 500 epochs using a learning rate of 0.01. A ReLU activation is applied after the first layer, and the final layer outputs node-level predictions. Mean aggregation is used to merge attention head outputs when applicable. For fairness-aware models, a dropout rate of 0.5 is applied. FairGNN adopts GCN as the backbone architecture and is trained with a learning rate of 0.001 for 1000 epochs. The regularization parameters are set to $\beta = 0.01$ and $\alpha = 2$. For FairVGNN, the generator is trained with a learning rate of 0.01 for a total of 500 iterations. In each iteration, both the discriminator and classifier are updated for 5 steps. The cut-off threshold is set to $c = 0.1$, while the weights for the classification loss, adversarial loss, and view generation loss are set to 0.5, 1, and 0.1, respectively. For FairSIN, the weights of the Demographic Parity regularization, equalized odds regularization, and the gradient reversal layer are set to 1, 5, and 1, respectively. FairTAG follows a three-stage training pipeline: oracle training, reference learning with fairness-aware edge selection and distillation, and final anchor node preference optimization. The oracle and reference language models are fine-tuned using a learning rate of 2×10^{-5} and batch size 32. Fairness score edge dropping is applied with $\gamma = 0.2$. Knowledge distillation is performed via KL divergence on unselected edges. The final model is optimized using DPO with $\beta = 0.1$, $\lambda = 0.1$, learning rate 1×10^{-5}, batch size 16, and 2 epochs.

D Dataset Details

Pokec Takac & Zabovsky (2012) is the most popular online social network in Slovakia. Datasets contains anonymized data of the whole network. Profile data contains gender, age, hobbies, interests, education, etc. We use the **Pokec-Z** and **Pokec-N** datasets processed by prior work Hussain et al. (2022), where the label is the working field, the sensitive attribute is region, and the textual content corresponds to personal information. **PubMed** Yang et al. (2016) is a citation network dataset commonly used in graph neural network research. In this dataset, nodes represent biomedical papers related to diabetes, and edges denote citation relationships between papers. The labels correspond to research categories, the sensitive attribute is the country of publication, and the textual content is the paper's abstract. Detailed dataset statistics are provided in Table 3.

Table 3: Overview of the Datasets

	Pokec-Z	Pokec-N	PubMed
Nodes	67,796	66,569	19,717
Edges	617,958	517,047	44,338
Labeled nodes	10,262	8,797	9,291
Avg. Character Length	441.29	408.43	1590.52

E Additional Descriptions on Baselines

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

Classical fairness attack methods:

- **FA-GNN Hussain et al. (2022)**: This is the first work to target GNNs with fairness adversarial attacks. Under a specified attack budget, the method randomly creates edges between nodes belonging to different label classes and different sensitive attribute groups. FA-GNN can be applied to both fairness evasion and fairness poisoning attacks.
- **G-FairAttack Zhang et al. (2024a)**: It formulates a novel surrogate fairness loss that serves as a common upper bound for diverse fairness metrics, and then sequentially greedily flips targeted edges to generate an optimally poisoned graph. This approach can be applied to both fairness evasion and fairness poisoning attacks.
- **FATE J et al. (2024)**: This work proposes a meta-learning framework by formulating a bi-level optimization problem. The framework exhibits broad applicability, capable of adapting to different fairness definitions, various graph learning models, and arbitrary operation strategies.
- **NIFA Luo et al. (2024b)**: This is the state-of-the-art node injection GNNs fairness attack algorithm. It introduces an uncertainty-maximization and a homophily-increase principle, and then optimizes the features of injected nodes to ensure the effectiveness of fairness attacks further. This method is limited to executing fairness poisoning attacks.

TAG utility attack methods:

- **WTGIA Lei et al. (2024)**: This study represents the first realization of a utility attack specifically targeting text-attributed nodes. Deviating from the direct injection of vectors, this study approach generates and injects human-readable text content, thereby pioneering utility poisoning attacks against Text-Attributed Graphs.
- **ATAG-LLM Fan et al. (2025)**: This study focuses on text-level graph injection attacks executed under a strict black-box setting. In this approach, node text content is directly generated rather than being derived from uninterpretable node embeddings, thereby successfully realizing utility attacks against Text-Attributed Graphs.

F Additional Descriptions on Victim Models

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

Classical classification GNNs:

- **GCN Kipf & Welling (2017)**: This is the seminal graph convolutional network that applies a first-order approximation of spectral graph convolutions, propagating and aggregating feature information from immediate neighbors to learn node embeddings in a semi-supervised setting.
- **GraphSAGE Hamilton et al. (2017)**: This is an inductive graph representation learning framework that samples and aggregates features from a node's local neighborhood using learnable aggregator functions, enabling generalization to unseen nodes and graphs.
- **GAT Velickovic et al. (2018)**: This is the graph attention network that leverages masked self-attention mechanisms to assign adaptive, learnable weights to neighboring nodes, enabling the model to selectively aggregate information from different neighbors when computing node embeddings.
- **GIN Xu et al. (2019):** 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 and fairness-aware GNN based on TGA:

- **FairGNN Dai & Wang (2021)**: This is the first fairness-aware GNN that incorporates an adversarial regularizer and a sensitive-attribute discriminator into the training loop, thereby mitigating bias amplification during message passing even when only limited sensitive attribute labels are available.
- **FairVGNN Wang et al. (2022)**: This is a multi-view fairness framework that learns to mask or reweight feature views most correlated with sensitive attributes, preventing leakage of protected information during aggregation and improving both group and individual fairness metrics.
- **FairSIN Yang et al. (2024)**: This is a fairness-enhanced semantic interaction network that integrates a structural debiasing module into a GNN backbone, aligning node representations across sensitive groups via contrastive learning to promote statistical parity and equalized odds.
- **FairTAG Luo et al. (2025)**: This work presents the first fairness-aware GNN algorithm specifically tailored for Text-Attributed Graph. It first mitigates the language model's overfitting to homophily during downstream fine-tuning, and subsequently optimizes the connection preferences of specific anchor nodes at the subgraph level.

Table 4: Runtime (s) and GPU Memory Consumption (MB) of UFPAT

	Pokec-Z	Pokec-N	PubMed
UFPAT-t (s)	131.70	255.63	1075.38
UFPAT-r (s)	56.75	80.75	70.84
UFPAT-p (MB)	5665.48	5542.68	1807.92

G Runtime and GPU Memory Consumption

This section presents the time and GPU memory overhead of the UFPAT algorithm, both of which are related to the number of injected nodes and the size of the graph. Regarding the runtime of UFPAT, UFPAT-t denotes the overall runtime of the algorithm, whereas UFPAT-r denotes the local runtime after excluding the waiting time associated with LLM interactions. As shown in the first and second rows of Table 4, we observe that the local runtime of UFPAT is extremely fast, while a substantial portion of the time is spent in the LLM interaction stage. This is reasonable, as interactions with LLMs are affected by factors such as network latency and the internal computation process of the LLM. Regarding the memory overhead of UFPAT, UFPAT-p denotes the peak increase in actual tensor GPU memory consumption. As shown in the third row of Table 4, UFPAT incurs very low GPU memory usage and can be readily executed on general-purpose devices.

Table 5: The attack performance of UFPAT (random masking of nodes with different proportions) on GCN during fair poisoning attacks on different datasets. The results are reported as a percentage (%).

	Pokec-Z			Pokec-N			Pubmed		
	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
Clean	71.06	5.78	7.97	70.23	1.29	1.17	92.67	3.19	3.90
10%	70.45	12.21	14.11	69.09	6.43	6.07	89.57	11.30	10.32
20%	70.87	11.82	13.66	69.69	6.28	5.92	89.93	10.62	8.29
30%	69.96	11.52	12.91	68.93	5.96	5.86	88.58	9.17	8.75
40%	70.28	10.75	12.43	69.52	5.15	4.97	88.46	8.56	8.13

H Partial Sensitive Attribute Knowledge

As stated in the attack setting described in Appendix A, this assumption is adopted primarily to follow the standard threat model used in existing gray-box fairness attacks and to ensure fair comparisons with state-of-the-art baselines. UFPAT follows this setting so that the evaluation can focus on the effectiveness of the proposed attack itself. In addition, we randomly masked the sensitive attributes of 10%, 20%, 30%, and 40% of the selected nodes and evaluated the poisoning-attack performance against GCN. By design, missing sensitive attributes do not affect the generation of malicious textual content. However, they influence the group fairness loss, group-specific saliency maps, and the assignment of injected nodes to sensitive groups. Consequently, as shown in Table 5, the attack effectiveness decreases slightly as the availability of sensitive attributes declines. Nevertheless, UFPAT can still substantially degrade model fairness even when 40% of the sensitive attributes are masked, demonstrating its broader applicability in scenarios where sensitive-attribute information is incomplete.

I The Generalization of cross-text-encoders

In this section, we evaluate the transferability of UFPAT's fairness poisoning attack across different text encoder configurations. Specifically, during the generation of poisoned data, UFPAT utilizes the pre-trained Paraphrase-MPNet-base-v2 model to embed node text information into continuous feature vectors. Conversely, when assessing cross-encoder transferability, we employ distinct pre-trained text encoders to re-embed the features of all nodes within the poisoned graph. As shown in Table 6, we select two comparative pre-trained text encoders (all-MiniLM-L6-v2, E5-base-v2) that exhibit significant differences in model scale and Transformer architecture, with Paraphrase-MPNet-base-v2 serving as the baseline. In all experimental setups, a GCN is employed as the victim model. Results indicate that despite a marginal decline in fairness poisoning performance upon switching text encoders, UFPAT maintains stable and significant attack effectiveness. This suggests that the structural poisoning induced by UFPAT exhibits low dependence on specific text encoders and demonstrates robust encoder-agnostic transferability. Consequently, the method proves highly feasible in low-privilege real-world scenarios where the attacker neither knows nor controls the victim model's text encoder configuration.

Table 6: Attack performance of UFPAT (use Paraphrase-MPNet-base-v2) on fairness poisoning attack against GCN using different text encoders across various datasets. The results are reported in percentage (%).

Encoder		Pokec-Z			Pokec-N			PubMed		
		ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
all-MiniLM-L6-v2	Clean	71.27	4.36	7.78	71.39	2.07	1.57	93.17	2.90	4.66
	UFPAT	70.82	13.21	14.20	69.57	7.24	6.65	88.93	11.57	10.61
Paraphrase-MPNet-base-v2	Clean	71.01	5.27	7.46	70.42	1.53	0.93	92.68	3.36	4.42
	UFPAT	70.33	12.85	14.71	69.60	6.51	6.83	89.02	11.32	10.60
E5-base-v2	Clean	71.40	5.23	7.34	70.27	1.88	1.03	92.73	3.85	4.12
	UFPAT	70.19	12.32	14.36	69.60	6.26	7.11	89.34	11.42	11.08

J Ablation Study and Hyperparameter Analysis

In this section, we analyze the impact of the attack budget b, the sparsity regularization term parameter α, and the utility constraint parameter β within UFPAT on attack performance, as these two parameters directly influence the utility performance of the victim model and the effectiveness of the fairness attack. The objective of UFPAT is to achieve fairness poisoning while minimizing the impact on model utility. Furthermore, to demonstrate the effectiveness of our strategies, we compare the edge modification method for malicious nodes in our algorithm against other approaches, as well as the selection of fairness loss $\mathcal{L}_{\Delta_{EO}}$ and $\mathcal{L}_{\Delta_{DP}}$.

J.1 Ablation study

In this subsection, we conduct an ablation study on UFPAT to examine the individual contributions of its components. We consider the following settings: (1) UFPAT denotes the original, unmodified configuration; (2) P_{random} connects the generated malicious nodes to existing graph nodes randomly; (3) P_{sc} involves connecting nodes that differ in both class and sensitive attributes without injecting new nodes, to demonstrate the effectiveness of the proposed injection method; (4) P_{EO} excludes the Equal Opportunity component from the fairness loss; and (5) P_{DP} excludes the Demographic Parity component. (6) P_{LLM} excludes the LLM component and uses random text. As reported in Table 7, we execute fairness poisoning attacks across three datasets using GCN as the victim model.

Table 7: Ablation study of each UFPAT setting. The best results Δ_{EO} and Δ_{DP} (%) are highlighted in bold. The victim model is GCN.

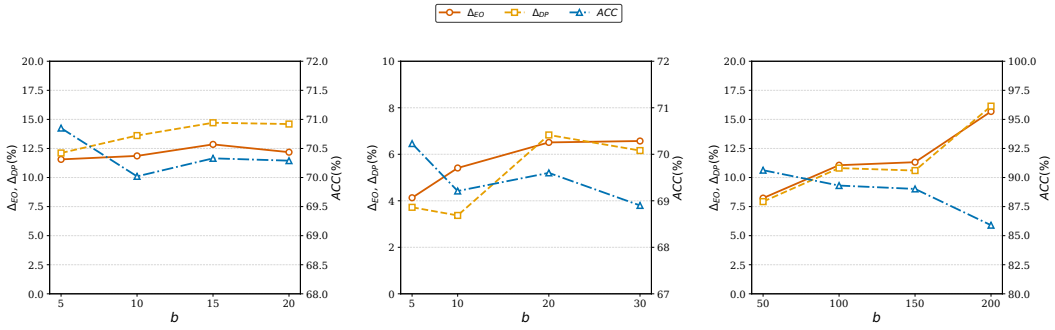
	Pokec-Z			Pokec-N			PubMed		
	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
Clean	71.01	5.27	7.46	70.42	1.53	0.93	92.68	3.36	4.12
P_{random}	71.25	5.33	7.18	70.35	1.80	1.24	92.64	3.07	4.56
P_{sc}	70.92	4.76	7.54	70.51	2.11	1.09	92.25	3.25	4.37
P_{LLM}	70.27	11.84	14.50	69.83	6.57	6.90	89.89	10.92	10.55
P_{EO}	70.84	6.16	8.82	70.65	1.69	1.45	91.47	2.83	3.98
P_{DP}	70.19	9.96	10.28	70.22	2.44	3.92	88.47	5.41	7.30
UFPAT	70.33	12.85	14.71	69.60	6.51	6.83	89.02	11.32	10.60

The results indicate that removing the malicious node edge modification strategies proposed in UFPAT (P_{random} and P_{sc}) leads to a significant degradation in fairness attack performance across all three datasets, demonstrating the effectiveness of the malicious node edge modification mechanism. Similarly, when certain components of the fairness loss are masked (P_{EO} or P_{DP}), the attack's effectiveness is substantially reduced, confirming the critical role of the proposed feature optimization strategy in enabling effective fairness attacks. Moreover, the attack remains effective when the LLM module is removed (P_{LLM}). This result validates a key innovation of UFPAT: its attack effectiveness does not depend on the textual content of the injected nodes. The LLM is introduced primarily to generate human-readable features for malicious nodes, thereby enhancing their stealthiness.

J.2 The utility–fairness trade-off curve of attack budget b

We investigated the impact of different values of b on the attack performance. As defined in subsection 4.2, b controls the number of malicious nodes injected by UFPAT, directly influencing the attack's intensity and model performance. The detailed results are shown in Figure 3, where GCN is used as the victim model, and fairness poisoning attacks are conducted on three datasets. It can be observed that increasing b does not necessarily lead to a continuous improvement in attack performance (Δ_{EO} and Δ_{DP}), and in some cases, performance stagnation or even slight degradation is observed. This phenomenon may be attributed to the fact that a larger b introduces more significant structural perturbations, thereby significantly enlarging the attack search space. The injected structural modifications may no longer focus on the optimal attack patterns and may even cause a dilution of the attack's effectiveness. Moreover, as b increases, the ACC exhibits a slight downward trend, which may be due to the stronger perturbations caused by the excessive injection of malicious

nodes into the original graph structure. Based on the above analysis and experiments exploring different ranges of b, the value of b selected in this study strikes a reasonable balance between attack effectiveness and model performance.

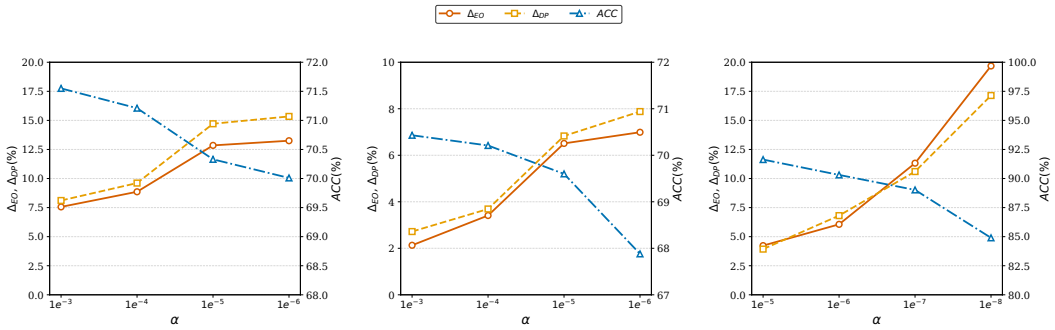


(a) Pokec-Z (b) Pokec-N (c) PubMed

Figure 3: The impact of different values of b on the attack performance (%) of UFPAT. GCN is used as the victim model, and fairness poisoning attacks are conducted on three datasets. 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.3 The utility–fairness trade-off curve of sparsity constraint α

We further investigated the impact of different values of α on the attack performance. As defined in subsection 4.3, α controls the strength of the sparsity constraint in UFPAT, directly affecting the attack's intensity. The detailed results are shown in Figure 4, where GCN is used as the victim model, and fairness poisoning attacks are conducted on three datasets. It can be observed that decreasing α significantly improves the attack performance (Δ_{EO} and Δ_{DP}), but simultaneously leads to a substantial decrease in model accuracy (ACC), with a drop exceeding 10%. This is reasonable, as a smaller α reduces the sparsity constraint, resulting in more edges being generated and injected, thereby enhancing the attack's effectiveness. However, the significant decline in ACC represents an unreasonable trade-off, as the notable decrease in model utility could lead to the model being flagged as abnormal, exposing the fairness poisoning attack. Based on the trade-off between attack performance and model utility, the α value selected in this study is deemed reasonable.



(a) Pokec-Z (b) Pokec-N (c) PubMed

Figure 4: The impact of different values of α on the attack performance (%) of UFPAT. GCN is used as the victim model, and fairness poisoning attacks are conducted on three datasets. 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 utility–fairness trade-off curve of utility constraint β

We further investigate the impact of different values of β on attack performance. According to the definition in Section 4.3, β controls the strength of the utility constraint in UFPAT, thereby in-

directly regulating the attack intensity and its impact on model utility. The detailed experimental results are shown in Figure 5, where GCN is used as the victim model and fairness poisoning attacks are conducted on the three datasets. As observed from the results, when β decreases, the attack-effectiveness metrics Δ_{EO} and Δ_{DP} improve slightly, while the model accuracy ACC decreases. This phenomenon is reasonable: a smaller β weakens the utility constraint, causing the attack process to place less emphasis on preserving the overall utility of the model, and thus enhancing the attack effectiveness to some extent. However, a substantial drop in ACC indicates an unreasonable trade-off between fairness degradation and utility preservation. Significant performance degradation may cause the model's behavior to be regarded as abnormal, thereby increasing the risk that the fairness poisoning attack is detected or exposed. Therefore, by jointly considering the balance between attack effectiveness and model utility, the value of β adopted in this paper is reasonable and appropriate, as it can effectively compromise model fairness while avoiding excessively obvious utility loss.

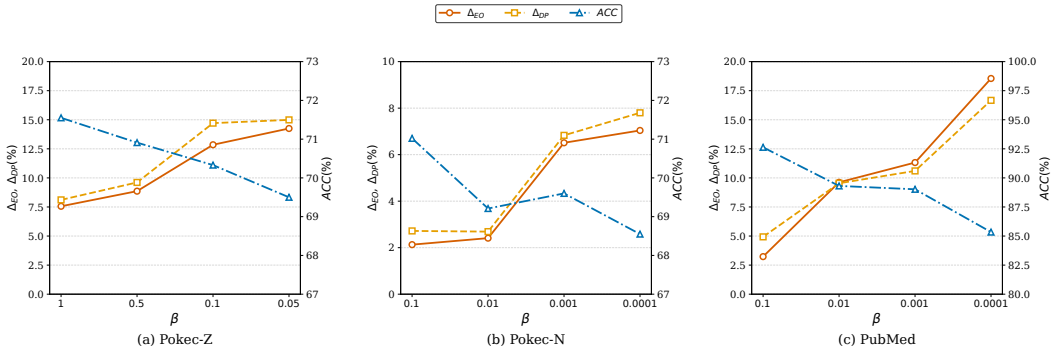


Figure 5: The impact of different values of β on the attack performance (%) of UFPAT. GCN is used as the victim model, and fairness poisoning attacks are conducted on three datasets. The x-axis denotes the varying values of β, while the left y-axis represents Δ_{EO} and Δ_{DP}, and the right y-axis corresponds to ACC.

Table 8: Attack performance of our method and SOTA baselines on fairness poisoning attacks against classical 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	Books-History			Ele-Computers			Ele-Photo		
		ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
GCN	Clean	95.65	1.02	2.24	96.45	0.59	0.55	99.58	0.21	2.00
	FA-GNN	94.58	2.86	1.22	95.35	1.10	1.47	97.79	5.67	1.07
	G-FairAttack	-	-	-	-	-	-	97.50	3.43	1.41
	FATE	-	-	-	-	-	-	97.52	2.23	1.70
	NIFA	93.82	4.10	0.14	94.09	0.75	1.38	97.77	2.49	0.81
	WTGIA	94.52	1.62	1.57	95.82	0.49	1.05	98.71	0.18	1.72
	ATAG-LLM	94.65	1.11	2.13	95.41	0.48	1.60	98.59	0.25	2.00
	Ours	94.02	**8.86**	**3.61**	94.21	**10.41**	**11.37**	98.31	**6.06**	**4.81**

K MORE DATASET

In this section, we present results on additional datasets and employ randomly assigned sensitive attributes to mitigate the influence of graph data distributions on the attack performance. We adopt the TAG datasets introduced in Yan et al. (2023), the first open benchmark specifically designed for text-attributed graphs. These datasets integrate text-annotated data from multiple domains and are cleaned and consolidated into structured graph datasets. We use the **Books-History**, **Ele-Computers**, and **Ele-Photo** datasets. Since these datasets do not provide sensitive attribute labels, following prior work Hussain et al. (2022); Zhang et al. (2024a); Luo et al. (2024b), we assign each node a binary sensitive attribute (0 or 1), sampled uniformly and independently of its classification label. For consistency with prior studies, we focus on binary classification–based fairness analysis

by selecting the first two classes of each dataset as labeled nodes and assigning a label of -1 to all remaining nodes. The labeled nodes are split into 50%/25%/25% for training, validation, and testing, respectively. As shown in Table 8, even when randomly assigned sensitive attributes are used to mitigate the influence of graph data distributions, UFPAT still achieves strong attack performance.

L Cost of UFPAT and Impact of Different LLM

We further evaluate the impact of different LLM APIs on the attack performance of the UFPAT. In UFPAT, the LLM is primarily used to generate readable textual features for malicious nodes. As detailed in Section 4.2, UFPAT treats the LLM to generate node textual features that are semantically aligned with the positive sample set while remaining distinct from the negative set through contrastive prompting. Meanwhile, the length of the generated text is constrained to not exceed the average length of node texts in the dataset. As shown in Table 9, GPT-5 mini is adopted as the default LLM API in our experiments, and DeepSeek-V3.2 is additionally introduced to assess the impact of different LLM choices on attack performance. The results indicate that text features generated by different LLMs exhibit a high degree of consistency in terms of attack effectiveness, with only negligible differences in fairness poisoning performance. This suggests that UFPAT is insensitive to the specific choice of LLM, and that semantically natural and readable textual features generated by different LLMs can all effectively support the attack process. These findings further validate the robustness and generality of UFPAT with respect to its text generation module.

Table 9: The impact of different LLM on the attack performance (%) of UFPAT. GCN is used as the victim model, and fairness poisoning attacks are conducted on three datasets.

	Pokec-Z			Pokec-N			PubMed		
	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
DeepSeek-v3.2	70.52	11.71	14.84	70.26	6.35	7.02	88.77	11.83	11.29
GPT-5 mini	70.33	12.85	14.71	69.60	6.51	6.83	89.02	11.32	10.60

Following standard practice, we evaluate the computational cost of our method by quantifying the frequency of LLM API invocations and their corresponding token consumption. Specifically, the token usage for a single LLM call can be formulated as:

$$T_{call} = T_{input} + T_{output}$$

where T_{input} denotes the tokens in the input prompt and the associated positive/negative sample sets, and T_{output} represents the tokens generated by the LLM. Given the inherent stochasticity in text generation, we report the average number of completion tokens and explicitly impose an upper bound constraint during the generation process. Throughout all experiments, UFPAT invokes the LLM $b \times c$ times. Under our experimental settings, each call consumes approximately $T_{input} = 1668$ input tokens and $T_{output} = 208$ output tokens on average. The number of LLM calls shows a linear relationship with the number of injected nodes. Under our experimental configuration, the total token volume remains within a tractable scale. Furthermore, as demonstrated in Table 9, the proposed method is model-agnostic and can be seamlessly integrated with open-source LLMs, thereby further reducing practical deployment overhead. The aforementioned analysis highlights that while UFPAT introduces an LLM as a text generation module, maintains favorable cost-efficiency.

M Readability Assessments and Defense to UFPAT

As illustrated in Table 2, given the diminished efficacy of existing defensive strategies, a natural countermeasure against the proposed attack involves an in-depth distributional analysis of node features in text-attributed graphs. Specifically, we perform a multi-faceted evaluation focusing on linguistic readability and feature-level specificity. This process involves quantifying metrics such as GPT-2 perplexity, Sentence-BERT cosine similarity, and the Gini coefficient, which collectively serve as a basis for heuristic filtering to identify and sequester potential malicious nodes:

- To evaluate the linguistic plausibility of the synthesized node text, we compute the average GPT-2 Perplexity (PPL). PPL serves as a metric to quantify the linguistic fluency and

naturalness of the generated sequences. Specifically, a lower PPL score indicates that the text exhibits higher predictability and coherence under the distribution of a pre-trained language model. PPL is defined as the exponentiated average negative log-likelihood of a token sequence.

- To assess whether the synthesized text aligns semantically with authentic nodes, we compute the average Sentence-BERT (SBERT) cosine similarity using a fixed pre-trained language model. SBERT cosine similarity measures the semantic proximity between the generated and reference texts in a high-dimensional embedding space. A higher similarity score indicates superior semantic fidelity, suggesting that the generated features effectively preserve the core informational content of the target distribution.
- To evaluate whether the synthesized text remains aligned with authentic nodes after embedding, we compute the average Gini coefficient using a fixed pre-trained language model. By quantifying the distributional disparity of node feature values, the Gini coefficient serves as a metric to assess the statistical dispersion of the malicious node representations.

Table 10: Analysis of the node feature distributions within the TAG, where UFPAT denotes the set of all malicious nodes generated by our method, while Clean denotes a baseline set of nodes randomly sampled from the original graph, equal in quantity to the malicious nodes.

	Pokec-Z			Pokec-N			PubMed		
	PPL	SBERT Sim	Gini	PPL	SBERT Sim	Gini	PPL	SBERT Sim	Gini
Clean	189.91	0.8590 (Intra)	0.1008	151.54	0.8668 (Intra)	0.1012	17.85	0.8973 (Intra)	0.1301
UFPAT	122.63	0.8770 (Inter)	0.1009	182.56	0.8870 (Inter)	0.1013	15.50	0.8959 (Inter)	0.1303

As illustrated in Table 10, PPL denotes the average perplexity of the node sets. SBERT Sim represents the average Sentence-BERT cosine similarity, where Clean indicates the intra-set similarity among original nodes, and UFPAT signifies the inter-set similarity between the synthesized malicious nodes and the original node set. Gini refers to the average Gini coefficient of the node feature distributions. The empirical results demonstrate that the discrepancies in semantic and embedding features between the poisoned nodes generated by UFPAT and the authentic nodes are negligible, as evidenced by the Gini coefficient and SBERT similarity metrics. Notably, the PPL of the poisoned nodes remains remarkably low, even achieving empirical parity with—or outperforming—the original nodes in terms of linguistic fluency. Consequently, these findings suggest that defenders are unlikely to mitigate the threat through conventional anomaly detection or outlier-based filtering strategies. This underscores UFPAT's capability to execute fairness poisoning attacks within complex, real-world semantic environments while maintaining exceptional stealthiness.

Table 11: Degree-based defense strategy.

	Pokec-Z			Pokec-N			PubMed		
	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
Clean	71.01	5.27	7.46	70.42	1.53	0.93	92.68	3.36	4.12
$K=0$	70.33	12.85	14.71	69.60	6.51	6.83	89.02	11.32	10.60
$K=5$	70.87	12.79	14.66	70.29	4.80	5.84	89.31	9.82	9.97
$K=10$	70.64	4.36	6.31	70.05	4.48	4.83	89.55	9.94	9.46
$K=15$	71.07	5.34	7.54	70.34	2.66	3.28	89.47	8.73	8.55

In addition, we report a degree-based defense strategy: when fairness bias is detected, we remove the top-k nodes. Specifically, we select the nodes with the highest degrees, where k is set to 5, 10, or 15, and delete their associated edges. As shown in Table 11, we evaluate this strategy on the GCN model across three datasets. The results indicate that this strategy can partially mitigate the UFPAT attack. We also evaluated the defense performance of Node-aware Bi-smoothing Lai et al. (2024), a defense method specifically designed for graph injection attacks, with the drop rate set to 0.25. As shown in Table 12, UFPAT remains effective even under this targeted defense setting.

Consequently, a more direct and effective defensive strategy necessitates a paradigm shift from focusing solely on traditional utility performance to the real-time auditing of fairness metrics in TAG-based GNN models. Such a proactive approach enables the early identification of data vulnerabilities, thereby facilitating timely intervention to preclude more severe downstream degradation.

Table 12: Node-aware Bi-smoothing defense strategy.

	Pokec-Z			Pokec-N			PubMed		
	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}	ACC	Δ_{EO}	Δ_{DP}
Clean	70.18	4.40	6.47	70.27	1.02	0.97	92.46	3.42	4.36
UFPAT	69.79	7.57	9.28	69.68	4.95	5.30	90.19	9.46	8.37

Ultimately, these findings underscore the urgent need for future research to develop more resilient and adaptive defense mechanisms capable of countering sophisticated semantic-level attacks.

N Societal Impact

This work studies fairness attacks on text-attributed graphs. The positive societal impact is that it exposes potential fairness vulnerabilities in graph learning systems, especially when textual attributes and graph structures jointly influence model predictions. By identifying such risks, this work can encourage the design of more robust, secure, and fairness-aware graph learning methods. However, the work also has potential negative societal impacts. The proposed attack framework may be misused by malicious actors to intentionally worsen group-level fairness or manipulate model behavior in sensitive applications. We therefore present this study as a controlled robustness analysis, with the goal of raising awareness and motivating future defenses rather than enabling harmful deployment. Similarly, we provide possible defense strategies in the Appendix M.

O Limitations

Although UFPAT demonstrates strong fairness poisoning effectiveness and cross-encoder transferability, it still has several limitations. First, while UFPAT is designed to preserve utility, it may cause slightly more noticeable accuracy degradation for certain dataset–victim model combinations, which could reduce attack stealthiness. Second, our threat model assumes access to graph structure, node texts, labels, sensitive attributes, and the training node set, while the victim model architecture, parameters, and text encoder remain hidden; therefore, extending UFPAT to stricter black-box settings with less side information remains an important direction. Finally, future work should evaluate broader fairness notions, multi-valued sensitive attributes, and more diverse real-world TAG scenarios.