Endowing Pre-trained Graph Models with Provable Fairness

Zhongjian Zhang*
zhangzj@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Mengmei Zhang*
zhangmm@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Yue Yu
yuyue1218@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Cheng Yang†
yangcheng@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Jiawei Liu
liu_jiawei@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Chuan Shi†
shichuan@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

ABSTRACT

Pre-trained graph models (PGMs) aim to capture transferable inherent structural properties and apply them to different downstream tasks. Similar to pre-trained language models, PGMs also inherit biases from human society, resulting in discriminatory behavior in downstream applications. The debiasing process of existing fair methods is generally coupled with parameter optimization of GNNs. However, different downstream tasks may be associated with different sensitive attributes in reality, directly employing existing methods to improve the fairness of PGMs is inflexible and inefficient. Moreover, most of them lack a theoretical guarantee, i.e., provable lower bounds on the fairness of model predictions, which directly provides assurance in a practical scenario. To overcome these limitations, we propose a novel adapter-tuning framework that endows pre-trained **Graph** models with **P**rovable f**A**i**R**ness (called GraphPAR[1]). GraphPAR freezes the parameters of PGMs and trains a parameter-efficient adapter to flexibly improve the fairness of PGMs in downstream tasks. Specifically, we design a sensitive semantic augmenter on node representations, to extend the node representations with different sensitive attribute semantics for each node. The extended representations will be used to further train an adapter, to prevent the propagation of sensitive attribute semantics from PGMs to task predictions. Furthermore, with GraphPAR, we quantify whether the fairness of each node is provable, i.e., predictions are always fair within a certain range of sensitive attribute semantics. Experimental evaluations on real-world datasets demonstrate that GraphPAR achieves state-of-the-art prediction performance and fairness on node classification task. Furthermore, based on our GraphPAR, around 90% nodes have provable fairness.

CCS CONCEPTS

• **Information systems** → **Data mining**; • **Applied computing** → **Law, social and behavioral sciences**.

KEYWORDS

Graph Neural Networks, Fairness, Pre-trained Graph Models

ACM Reference Format:
Zhongjian Zhang, Mengmei Zhang, Yue Yu, Cheng Yang, Jiawei Liu, and Chuan Shi. 2024. Endowing Pre-trained Graph Models with Provable Fairness. In *Proceedings of the ACM Web Conference 2024 (WWW '24), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3589334.3645703

1 INTRODUCTION

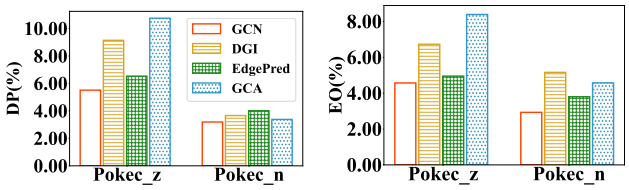
Graph Neural Networks (GNNs) [40, 45] have achieved significant success in analyzing graph-structured data, such as social networks [13] and webpage network [43]. Recently, inspired by pre-trained language models, various pre-trained graph models (PGMs) [18, 39, 46] have been proposed. Generally, PGMs capture transferable inherent graph structure properties through unsupervised learning paradigms in the pre-training phase, and then adapt to different downstream tasks by fine-tuning. As a powerful learning paradigm, PGMs have received considerable attention in the field of graph machine learning and have been broadly applied in various domains, such as recommendation systems [15] and drug discovery [41].

However, recent works [11, 31] have demonstrated that pre-trained language models tend to inherit bias from pre-training corpora, which may result in biased or unfair predictions towards sensitive attributions, such as gender, race and religion. With the same paradigm, PGMs raise the following question: *Do pre-trained graph models also inherit bias from graphs?* To answer this question, we evaluate the node classification fairness of three different PGMs on datasets Pokec_z and Pokec_n [38], the results as depicted in Figure 1. We observe that PGMs are more unfair than vanilla GCN. This is because PGMs can well capture semantic information on graphs during the pre-training phase, which inevitably contains sensitive attribute semantics. A further question naturally arises: *How to improve the fairness of PGMs?* Addressing this problem is highly critical, especially in graph-based high-stake decision-making scenarios, such as social networks [22] and candidate-job

*Both authors contributed equally to this research.
†Corresponding authors.
[1]The source code can be found at https://github.com/BUPT-GAMMA/GraphPAR.

WWW '24, May 13–17, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0171-9/24/05…$15.00
https://doi.org/10.1145/3589334.3645703



(a) Demographic Parity (DP). **(b) Equality Opportunity (EO).**

Figure 1: An example of evaluating the node classification fairness of PGMs on Pokec_z and Pokec_n datasets. DP (↓) and EO (↓) report the fairness of three PGMs (i.e., DGI, EdgePred, GCA) and vanilla GCN.

matching [21], because biased predictions will raise severe ethical and societal issues.

Although substantial methods have been proposed for developing fair GNNs in recent years, directly employing them to improve the fairness of PGMs is inflexible and inefficient. Namely, most existing works generally train a fair GNN for a specific task. For example, methods based on counterfactual fairness [1, 12, 29] train a fair GNN encoder by constructing different counterfactual training samples. Approaches with sensitive attribute classifiers [7, 26, 42] constrain GNNs to capture sensitive semantic information by introducing an adversarial loss in the training phase. Obviously, in the above methods, the debiasing process is coupled with the parameter optimization of GNNs. However, in reality, the same PGM can be used in different downstream tasks and different downstream tasks may be associated with different sensitive attributes [6, 32, 44]. Debiasing for a specific task in the pre-training phase is inflexible, and maintaining a specific PGM for each task is inefficient. Besides, most existing fairness methods lack theoretical analysis and guarantees [3, 20], meaning that they do not provide a practical guarantee, i.e., provable lower bounds on the fairness of model prediction. This is significant for determining whether to deploy models in practical scenarios [5, 19, 34, 35].

In this paper, we attempt to address the above questions by proposing GraphPAR, a novel adapter-tuning framework for efficiently and flexibly endowing PGMs with provable fairness. Specifically, in downstream tasks, we first freeze the parameters of PGMs, then design a sensitive semantic augmenter on node representations, to extend the node representations with different sensitive attribute semantics for each node. The extended representations will be directly used to tune an adapter so that the adapter-processed node representations are independent of sensitive attribute semantics, preventing the propagation of sensitive attribute semantics from PGMs to task predictions. Furthermore, with GraphPAR, we quantify whether the fairness of each node is provable, i.e., predictions are always fair within a certain range of sensitive attribute semantics. For example, when a person's gender semantics gradually transit from male to female, our provable fairness guarantees that the prediction results will not change. In summary, GraphPAR can apply to any PGMs while providing fairness with theoretical guarantees.

Our main contributions can be summarized as follows:

- We first explore the fairness of PGMs and find that PGMs can capture sensitive attribute semantics during the pre-training phase, leading to unfairness in downstream task predictions.
- We propose GraphPAR for efficiently and flexibly endowing PGMs with provable fairness. Specifically, during the adaption of downstream tasks, GraphPAR utilizes an adapter for parameter-efficient adaption and a sensitive semantic augmenter for fairness with practical guarantees.
- Extensive experiments on different real-world datasets demonstrate that GraphPAR achieves state-of-the-art prediction performance and fairness. Moreover, with the help of GraphPAR, around 90% of nodes have provable fairness.

2 RELATED WORK

2.1 Pre-trained Graph Models

Inspired by pre-trained language models, pre-trained graph models (PGMs) capture transferable inherent graph structure properties through unsupervised learning paradigms during the pre-training phase, and then adapt to different downstream tasks by fine-tuning. Based on different pre-training methods, the existing PGMs can mainly be categorized into contrastive and predictive pre-training. Contrastive pre-training maximizes mutual information between different views, encouraging models to capture invariant semantic information across various perspectives. For example, DGI [39] and InfoGraph [36] generate expressive representations for nodes or graphs by maximizing the mutual information between graph-level and substructure-level. GraphCL [48] and its variants [37, 47] further introduce a range of sophisticated augmentation strategies for constructing different views. Unlike contrastive pre-training, predictive pre-training enables models to understand the universal structural and attribute semantics of graphs. For instance, attribute masking is proposed by [18] where the input node attributes or edge are randomly masked, and the GNN is asked to predict them. EdgePred [14] samples negative edges and trains a general GNN to predict edge existence. GraphMAE [17] incorporates feature reconstruction and a re-mask decoding strategy to pre-train a GNN.

Despite the ability of PGMs to capture abundant knowledge that proves valuable for downstream tasks, the conventional fine-tuning process still has some drawbacks, such as overfitting, catastrophic forgetting, and parameter inefficiency[25]. To alleviate these issues, recent research has focused on developing parameter-efficient tuning (delta tuning) techniques that can effectively adapt pre-trained models to downstream tasks [8]. Delta tuning [8] seeks to tune a small portion of parameters and keep the left parameters frozen. For example, prompt tuning [28] aims to modify model inputs rather than PGMs parameters. Adapter tuning [25] trains only a small fraction of the adapter parameters to adapt pre-trained models to downstream tasks.

Though a large number of researches have been proposed on how to design pre-training methods and fine-tune PGMs in downstream tasks, most of them focus on improving performance while ignoring their plausibility in fairness and so on.

2.2 Fairness of Graph

Recent study [7] shows that GNNs tend to inherit bias from training data and the message-passing mechanism of GNNs could magnify the bias. Hence, many efforts have been made to develop fair GNNs. According to the stage at which the debiasing process occurs, the existing methods could be split into the pre-processing, in-processing,

and post-processing methods [3]. Pre-processing methods remove bias before GNN training occurs by targeting the input graph structure, input features, or both. For instance, EDITS [9] propose a novel approach that utilizes the Wasserstein distance to mitigate both attribute and structural bias on graphs. In-processing methods focus on modifying the objective function of GNNs to learn fair or unbiased embeddings during training. For example, NIFTY [1] proposes a novel multiple-objective function incorporating fairness and stability. Graphair [26] introduces an automated augmentation model that generates a graph for fairness and informativeness. A few post-processing methods have been proposed to remove bias from GNNs. FairGNN [7] trains a fair GNN encoder through an adversary task of predicting sensitive attributes. FLIP [30] further addresses the problem of link prediction homophily by adversarial learning.

Although all the methods above have achieved significant success in graph fairness, most of them require optimizing the parameters of GNN. Since different downstream tasks may be associated with different sensitive attributes, these methods cannot flexibly and efficiently improve the fairness of PGMs. Besides, they all lack theoretical analysis and fairness guarantees, which are significant for determining whether to deploy a model in a real-world scenario.

3 PRELIMINARY

3.1 Notations

Given an attributed graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V} = \{v_1, ..., v_n\}$ represents the set of n nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents the set of edges, $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ represents the node features and $\mathbf{x}_i \in \mathbb{R}^d$. The adjacency matrix of the graph $\mathcal{G}$ is denoted as $\mathbf{A} \in \mathbb{R}^{n \times n}$, where $\mathbf{A}_{ij} = 1$ if nodes v_i and v_j are connected, otherwise $\mathbf{A}_{ij} = 0$. Each node i is associated with a binary sensitive attribute $s_i \in \{0, 1\}$ (we assume one single, binary sensitive attribute for simplicity, but our method can easily handle multivariate sensitive attributes as well). Furthermore, we consider a PGM or GNN denoted as f, which takes the graph structure and node features as input and produces node representations. The encoded representations for the n nodes are denoted by $\mathbf{H} = \{\mathbf{h}_i\}_{i=1}^n$, where $\mathbf{H} = f(\mathcal{V}, \mathcal{E}, \mathbf{X})$ and $\mathbf{h}_i \in \mathbb{R}^p$.

In the pre-training phase, the parameters of a PGM f are optimized via self-supervised learning, such as graph contrastive learning [36, 39, 47, 48] or graph context prediction [14, 16–18]. In the downstream adaption phase, adapter tuning freezes the parameter f_θ of the PGM f and tunes parameter g_θ of an adapter g to adapt PGM for different downstream tasks. Generally, $|g_\theta| \ll |f_\theta|$, where $|\cdot|$ denotes the number of parameters. In the adapter, given an input $\mathbf{h}_i \in \mathbb{R}^p$, a down projection projects the input to a q-dimensional space, after which a nonlinear function is applied. Then the up-projection maps the q-dimensional representation back to p-dimensional space.

3.2 Fairness Definition on Graph

The fairness definition on the graph refers to the model prediction results not being influenced by sensitive attributes of nodes, i.e., the prediction results will not change as the sensitive attribute value variations [29].

Definition 1 (Fairness on Graph). *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, the encoder $f(\cdot)$ and the classifier $d(\cdot)$ trained on this graph satisfies fairness if for any node v_i:*

$$P((\hat{y}_i)_{S \leftarrow s} | \mathbf{X}, \mathbf{A}) = P((\hat{y}_i))_{S \leftarrow s'} | \mathbf{X}, \mathbf{A}), \quad s.t.\ \forall s \neq s', \tag{1}$$

where $\hat{y}_i = d \circ f(\mathbf{X}, \mathbf{A})_i$ denotes the predicted label for node v_i, and $s, s' \in \{0, 1\}^n$ are two arbitrary sensitive attribute values.

4 THE PROPOSED FRAMEWORK

In this section, we proposed a novel adapter-tuning framework, GraphPAR, which flexibly and efficiently improves the fairness of PGMs. First, we define the fairness of PGMs on GraphPAR, requiring that the predictions are not affected by the sensitive attribute semantics variations of the node. Next, to achieve the fairness objective, as depicted in Figure 2 (a) Adapter Tuning, GraphPAR consists of two key components: (1) A sensitive semantic augmenter, which extends the node representations with different sensitive attribute semantics for each node, to help further train an adapter. (2) An adapter, which transforms the node representations to be independent of sensitive attribute semantics by adversarial debiasing methods, preventing the propagation of sensitive attribute semantics from PGMs to task predictions.

4.1 Fairness Definition of PGMs

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ and the PGM f pre-trained on this graph. GraphPAR freezes the parameters of f to obtain node representations $\mathbf{H} = f(\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathbf{H}$ inevitably contains sensitive attribute semantics on graph $\mathcal{G}$ and therefore we hope utilizes an adapter g to remove them. Combined with Definition 1, the fairness of PGMs on GraphPAR is defined as follows:

Definition 2 (Fairness of PGMs). *In GraphPAR, a PGM $f(\cdot)$, an adapter $g(\cdot)$ and a classifier $d(\cdot)$ satisfy fairness during adaptation to downstream tasks if for any node v_i:*

$$P((\hat{y}_i)_{\mathbf{S}_\mathbf{h} \leftarrow \mathbf{s}} | \mathbf{H}) = P((\hat{y}_i)_{\mathbf{S}_\mathbf{h} \leftarrow \mathbf{s}'} | \mathbf{H}), \quad \forall \mathbf{s}, \mathbf{s}' \ \ s.t.\ ||\mathbf{s} - \mathbf{s}'||_2 \neq 0, \tag{2}$$

where $\hat{y}_i = d \circ g(\mathbf{h}_i)$ denotes the predicted label for node v_i. $\mathbf{s}$ and $\mathbf{s}'$ are two vectors with the same dimension as the node representation $\mathbf{h}_i$, i.e., $\mathbf{s}, \mathbf{s}' \in \mathbb{R}^p$, representing different sensitive attribute semantics.

The Definition 2 means that the model prediction results remain consistent as the sensitive attribute semantics change. For example, when a person's gender semantics gradually transits from male to female, fairness is satisfied if the model predictions are always consistent, otherwise not satisfied. Thus, the objective of GraphPAR is to remove the impact of sensitive attribute semantics from $\mathbf{H}$ on model prediction by training an adapter g, improving the fairness of PGMs in downstream tasks.

4.2 Sensitive Semantic Augmenter

The sensitive semantic augmenter extends the node representation with different sensitive attribute semantics for each node, to help further train the adapter. Initially, according to known sensitive attribute information on the graph and representations of the nodes, we calculate a vector $\boldsymbol{\alpha}$ that represents the direction of the sensitive attribute semantics. Subsequently, we extend the node representation $\mathbf{h}_i$ for each node via linearly interpolating in the direction of $\boldsymbol{\alpha}$, obtaining a sensitive attribute semantics augmentation set $\mathcal{S}_i$.

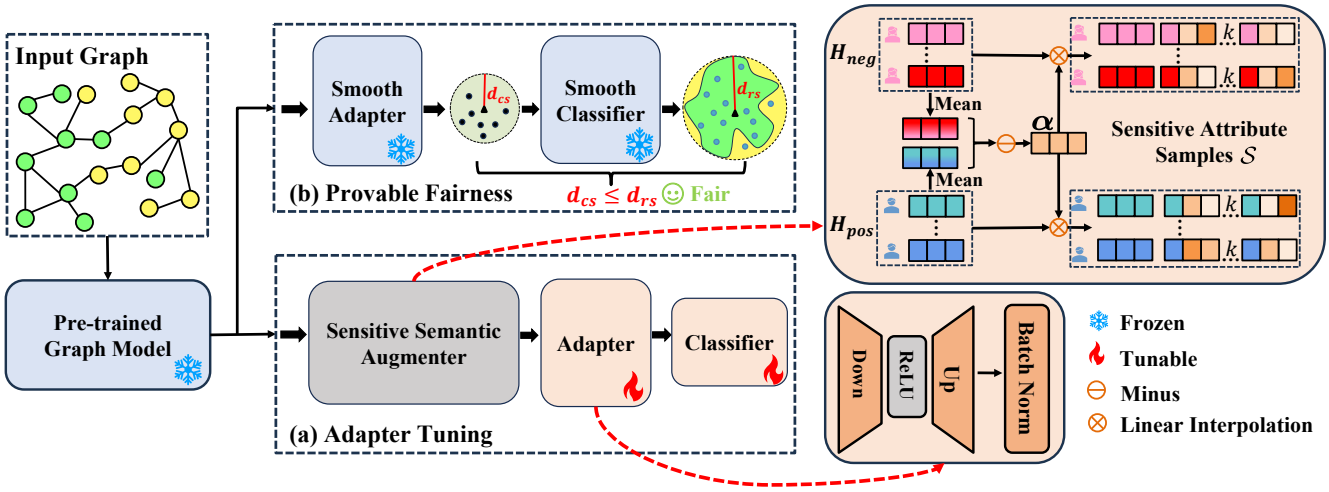


Figure 2: Overview of GraphPAR. In the adapter tuning phase, we first utilize the PGMs to obtain node representations H. Then, we design a sensitive semantic augmenter to extend the node representations with different sensitive attribute semantics, i.e., sensitive attribute samples $\mathcal{S}$. Finally, the extended node representations are used to train an adapter, transforming the node representations to be independent of sensitive attribute semantics. In the provable fairness phase, based on the smoothed versions of the well-trained adapter and classifier, we use the smooth adapter to get its output bound guarantee d_{cs} and use the smooth classifier to get its local robustness guarantee d_{rs}. Sequentially, we quantify whether the fairness of each node is provable by comparing d_{cs} with d_{rs}.

Computing the sensitive attribute semantics vector $\boldsymbol{\alpha}$. Leveraging the capabilities of PGMs in capturing both graph structure and node attributes, we expect to derive a vector $\boldsymbol{\alpha}$ that effectively represents the sensitive attribute semantics. First, we utilize the given PGM f to obtain node representations $\mathbf{H}$. Then, based on known node sensitive attribute s on the graph, we partition the node representations into positive and negative sets, i.e., $\mathbf{H}_{pos}$ and $\mathbf{H}_{neg}$. Subsequently, we calculate the average representation $\mathbf{h}_{pos}$ for nodes with the sensitive attribute and $\mathbf{h}_{neg}$ for nodes without it, both obtained from $\mathbf{H}_{pos}$ and $\mathbf{H}_{neg}$, respectively. Lastly, the difference between $\mathbf{h}_{pos}$ and $\mathbf{h}_{neg}$ represents the sensitive attribute semantics vector:

$$\boldsymbol{\alpha} = \mathbf{h}_{pos} - \mathbf{h}_{neg}, \tag{3}$$

$$\mathbf{h}_{pos} = \frac{1}{n_{pos}} \sum_{i=1}^{n_{pos}} \mathbf{H}_{pos,i}\,, \mathbf{h}_{neg} = \frac{1}{n_{neg}} \sum_{i=1}^{n_{neg}} \mathbf{H}_{neg,i}\,, \tag{4}$$

where n_{pos} and n_{neg} denote the number of positive and negative samples. Intuitively, the key thought in calculating $\boldsymbol{\alpha}$ is to represent the semantic relationships between groups by using the average difference in embeddings across different groups. Averaging is done to obtain a semantic embedding that represents the characteristics of a group and eliminates individual characteristic differences. For example, Mean (salesman, waiter, king) yields an embedding representing the male group, and Mean (saleswoman, waitress, queen) can produce an embedding for the female group. Subtraction is used to capture the semantic relationship between groups. For instance, man − woman represents the translation vector from female to male. If a user u is female, u + (man − woman) can yield a male representation of that user. In summary, with the vector $\boldsymbol{\alpha}$, we expect to move in the direction of $\boldsymbol{\alpha}$ to increase the presence of the sensitive attribute, while moving in the opposite direction diminishes its presence. In the experiment section, we conduct detailed experiments to illustrate the effectiveness of $\boldsymbol{\alpha}$.

Augmenting sensitive attribute semantics. After calculating the sensitive attribute semantics vector $\boldsymbol{\alpha}$, we employ it to augment a set of sensitive attribute $\mathcal{S}_i$ for each node representation $\mathbf{h}_i$. This augmentation is achieved through a linear interpolation method and can be expressed as:

$$\mathcal{S}_i := \{\mathbf{h}_i + t \cdot \boldsymbol{\alpha} \mid |t| \leq \epsilon\} \subseteq \mathbb{R}^p, \tag{5}$$

where ϵ represents the augmentation range applied to the direction of the sensitive attribute semantics. The above augmentation method offers two key advantages: (1) It efficiently extends node representations with different semantics of sensitive attributes as line segments. These line segments correspond to multiple points in the original input space, bypassing complex augmentation designs in the original graph [1, 49]. (2) Although this work primarily focuses on single sensitive attribute scenarios, this method can be simply extended to situations involving multiple sensitive attributes. In such cases, interpolation can be performed along multiple sensitive attribute semantics vectors.

4.3 Training Adapter for PGMs Fairness

Given any PGM f and the sensitive attribute augmentation set $\mathcal{S}$, we now outline how to improve the fairness of PGMs by training a parameter-efficient adapter g while ensuring the prediction performance of downstream tasks. In GraphPAR, we employ two adversarial debiasing methods for training the adapter: random augmentation and min-max adversarial training.

Random augmentation adversarial training (RandAT). During the adapter g training process, we choose k samples from the augmented sensitive attribute set $\mathcal{S}_i$ to obtain adversarial training

set $\hat{\mathcal{S}}_i$, i.e.,

$$\hat{\mathcal{S}}_i = \{\mathbf{h}_i + t_j \cdot \boldsymbol{\alpha}\}_{j=1}^{k},\ t_j \sim \text{Uniform}(-\epsilon, \epsilon), \tag{6}$$

where ϵ represents the augmentation range. These selected samples are then incorporated into the training of the adapter. The optimization loss can be formulated as follows:

$$\mathcal{L}_{\text{RandAT}} = \mathbb{E}_{i \in \mathcal{V}_L}\left[\mathbb{E}_{\mathbf{h}_i' \in \hat{\mathcal{S}}_i}\left[\ell(d \circ g(\mathbf{h}_i'), y_i)\right]\right], \tag{7}$$

where $\mathcal{V}_L$ is the set of labeled nodes, d is a downstream classifier, and $\ell(\cdot)$ is cross-entropy loss which measures the prediction error.

In RandAT, by introducing diverse sensitive attribute semantic samples in the training process, the adapter g and the classifier d become more robust to variations in sensitive information, mitigating potential discriminatory predictions. At the same time, these augmented samples share the same task-related semantics as the original sample, which further helps the adapter and classifier capture task-related semantics.

Min-max adversarial training (MinMax). Unlike RandAT, the key thought behind MinMax is to find and optimize the worst-case in each round. Our objective is to minimize the discrepancy between the representation $\mathbf{h}_i$ and its corresponding augmented sensitive attribute semantics set $\mathcal{S}_i$. This is achieved by ensuring that the representation $\mathbf{h}_i$ closely aligns with the representations within $\mathcal{S}_i$. To quantify this alignment, we seek to minimize the distance between $\mathbf{h}_i$ and $\mathcal{S}_i$. The optimization objective of MinMax is minimizing the following loss function:

$$\mathcal{L}_{MinMax}\left(\mathbf{h}_i\right) = \max_{\mathbf{h}_i' \in \mathcal{S}_i} \left\| g\left(\mathbf{h}_i\right) - g\left(\mathbf{h}_i'\right)\right\|_2, \tag{8}$$

where $\mathbf{h}_i'$ have different sensitive attribute semantics with $\mathbf{h}_i$. Minimizing $\mathcal{L}_{adv}(\mathbf{h}_i)$ is a min-max optimization problem, and adversarial training is effective in this scenario. Since the input domain of the inner maximization problem is a simple line segment about $\boldsymbol{\alpha}$, we can perform adversarial training [10] by uniformly sampling k points from $\mathcal{S}_i$ to construct $\hat{\mathcal{S}}_i$ and approximate it as follow:

$$\mathcal{L}_{MinMax}\left(\mathbf{h}_i\right) \approx \max_{\mathbf{h}_i' \in \hat{\mathcal{S}}_i} \left\| g\left(\mathbf{h}_i\right) - g\left(\mathbf{h}_i'\right)\right\|_2. \tag{9}$$

To further ensure that the adapter g does not filter out useful task information, we introduce cross-entropy classification loss to ensure the performance of the downstream task:

$$\mathcal{L}_{cls}\left(\mathbf{h}_i, y_i\right) = \ell(d \circ f(\mathbf{h}_i), y_i). \tag{10}$$

The final optimization objective of MinMax is as follows:

$$\mathcal{L} = \lambda \mathcal{L}_{MinMax} + \mathcal{L}_{cls}, \tag{11}$$

where λ is a scale factor for balancing accuracy and fairness.

5 PROVABLE FAIR ADAPTATION OF PGMS

In this section, based on GraphPAR, we primarily discuss how to provide provable fairness for each node, i.e., the prediction results are consistent within a certain range of sensitive attribute semantics. We divide this process into two key components as depicted in Figure 2 (b) Provable Fairness: (1) Smooth adapter. We construct a smoothed version for the adapter using center smooth, which provides a bound for the output variation of node representation $\mathbf{h}$ within the range of sensitive attribute semantics change. This guarantees that the range of output results is contained within a minimal enclosing ball centered at $\mathbf{z}$ with a radius of d_{cs}. (2) Smooth classifier. We construct a smoothed version for the classifier using random smooth, which provides local robustness against the center $\mathbf{z}$. By determining whether all points within the minimum enclosing ball are classified into the same class, i.e., $d_{cs} < d_{rs}$, we quantify if the fairness of each node is provable. Note that based on the well-trained adapter and classifier, the smoothed models are constructed by different definitions of the smoothing function. Thus, the process of construction does not require training in any parameters. We leave out the subscript $(\cdot)_i$ for notation simplicity.

5.1 Provable Adaptation

To guarantee the range of change in the representation after applying the adapter g, we employ center smoothing [24] to obtain a smoothed version of the adapter, denoted as $\widehat{g}$. It provides a guarantee for the output bound of the adapter with a representation $\mathbf{h}$ as the input, described in Theorem 1:

Theorem 1 (Center Smoothing [24]). *Let $\widehat{g}$ denote an approximation of the smoothed version of the adapter g, which maps a representation $\mathbf{h}$ to the center point $\widehat{g}(\mathbf{h})$ of a minimum enclosing ball containing at least half of the points $\mathbf{z} \sim g(\mathbf{h} + \mathcal{N}(0, \sigma_{cs}^2 I))$. The formal definition of $\widehat{g}$ as follows:*

$$\widehat{g}(\mathbf{h}) = \underset{\mathbf{z}}{\text{argmin}}\ r\ s.t.\ \mathbb{P}[g(\mathbf{h} + \mathcal{N}(0, \sigma_{cs}^2 I)) \in \mathcal{B}(\mathbf{z}, r)] \geq \frac{1}{2}, \tag{12}$$

where $\mathbf{z}$ and r are the center and radius of the minimum enclosing ball, respectively. Then, for an l_2-perturbation size $\epsilon_1 > 0$ on $\mathbf{h}$, we can produce a guarantee d_{cs} of the output change with confidence $1 - \alpha_{cs}$:

$$\forall\, \mathbf{h}'\ s.t.\ \|\mathbf{h} - \mathbf{h}'\|_2 \leq \epsilon_1\,, \|\widehat{g}(\mathbf{h}) - \widehat{g}(\mathbf{h}')\|_2 \leq d_{cs}, \tag{13}$$

since $\mathbf{h}' - \mathbf{h} = t \cdot \boldsymbol{\alpha}$, we have:

$$\epsilon_1 = \max_t \|t\boldsymbol{\alpha}\| = \epsilon\|\boldsymbol{\alpha}\|_2, \tag{14}$$

where ϵ represents the augmentation range applied to the direction of the sensitive attribute semantics.

Theorem 1 implies that given a node representation $\mathbf{h}$ and its set of sensitive attribute samples $\mathcal{S}$ with range ϵ, a guarantee d_{cs} can be computed with high probability. This d_{cs} represents the range of the adapter output changes, serving as a meaningful certificate. It guarantees that when the sensitive attribute semantics of input $\mathbf{h}$ is perturbed within a range defined by ϵ, the range of output remains within a minimal enclosing ball.

5.2 Provable Classification

Next, it is necessary to demonstrate that predictions for all points within this minimum enclosing ball are classified consistently. This consistency guarantees the effectiveness of debiasing results.

Theorem 2. *(Random Smoothing [4]) Let d be a classifier and let $\varepsilon \sim \mathcal{N}(0, \sigma_{rs}^2 I)$. The smoothing version of the classifier $\widehat{d}$ is defined as follows:*

$$\widehat{d}\,(\mathbf{z}) = \arg\max_{y \in \mathcal{Y}} \mathbb{P}_\varepsilon(d(\mathbf{z} + \varepsilon) = y). \tag{15}$$

Suppose $y_A \in \mathcal{Y}$ and $\underline{p_A}, \overline{p_B} \in [0,1]$ satisfy:

$$\mathbb{P}_\varepsilon(d(\mathbf{z}+\varepsilon) = y_A) \geq \underline{p_A} \geq \overline{p_B} \geq \max_{y_B \neq y_A} \mathbb{P}_\varepsilon(d(\mathbf{z}+\varepsilon) = y_B). \quad (16)$$

Then, we have $\widehat{d}(\mathbf{z}+\delta) = y_A$ for all δ satisfying $\|\delta\|_2 < d_{rs}$, where d_{rs} can be obtain as follow:

$$d_{rs} := \frac{\sigma_{rs}}{2}(\Phi^{-1}(\underline{p_A}) - (\Phi^{-1}(\overline{p_B})), \quad (17)$$

where $\mathcal{Y}$ denotes the set of class labels, Φ is the cumulative distribution function (CDF) of the standard normal distribution $\mathcal{N}(0,1)$, and Φ^{-1} is its inverse.

Theorem 2 derives a local robustness radius d_{rs} for the input $\mathbf{z}$ by employing the smoothed version $\widehat{d}$ of the classifier d. This robustness guarantees that within the verified region of input, which is bounded by d_{rs}, the classification output of $\widehat{d}$ remains consistent, providing a guarantee of stability and consistency in the prediction results. Theorem 2 is especially important for providing provable fairness, because if $d_{cs} < d_{rs}$, then it guarantees consistency in the predictions to different sensitive attribute semantic samples.

5.3 GraphPAR Provides Provable Fairness

To establish a theoretical guarantee for the debiasing effect of adapter g, combining Theorem 1 and Theorem 2, we define the provable fairness of PGMs as follows:

Definition 3 (Provable Fairness of PGMs). *Given a node representation $\mathbf{h}$, the debiasing process M satisfies:*

$$M(\mathbf{h}) = M(\mathbf{h}'), \forall\, \mathbf{h}' \in \mathcal{S}\,, \quad (18)$$

where $\mathcal{S}$ is the set of sensitive attribute augmentations for $\mathbf{h}$.

With the aforementioned two smoothing techniques, the provable fairness of PGMs is naturally achieved with the following theorem:

Theorem 3. *Assuming we have a PGM f, a center smoothing adapter $\widehat{g}$, and a random smoothing classifier $\widehat{d}$. For the i-th node, if $\widehat{g}$ obtains a output guarantee d_{cs} with confidence $1-\alpha_{cs}$ and $\widehat{d}$ obtains a local robustness guarantee d_{rs} with confidence $1-\alpha_{rs}$, and satisfy $d_{cs} < d_{rs}$, then the fairness of the debiasing $M = \widehat{d} \circ \widehat{g} \circ f(\mathcal{V}, \mathcal{E}, \mathbf{X})_i$ is provable with a confidence $1-\alpha_{cs}-\alpha_{rs}$, the definition of the formalization is as follows:*

$$\forall\, \mathbf{h}' \in \mathcal{S} : \widehat{d} \circ \widehat{g}(\mathbf{h}) = \widehat{d} \circ \widehat{g}(\mathbf{h}')\,, \quad (19)$$

where $\mathbf{h} \in \mathbf{H}, \mathbf{H} = f(\mathcal{V}, \mathcal{E}, \mathbf{X})$.

The detailed algorithms process of Theorem 3 is referred to Appendix A, and the proof of Theorem 3 is referred to Appendix B.

6 EXPERIMENTS

In this section, we extensively evaluate GraphPAR to answer the following research questions (RQs). **RQ1**: How effective is GraphPAR compared to existing graph fairness methods? **RQ2**: Compared to methods without debiasing adaptation, does GraphPAR show improvement in the number of nodes with provable fairness? **RQ3**: How effective is the vector $\boldsymbol{\alpha}$ in representing the sensitive attribute semantics direction? **RQ4**: How do different hyperparameters of GraphPAR impact the classification performance and fairness? **RQ5**: How parameter-efficient is GraphPAR?

Table 1: Datasets Statistics.

Dataset	Credit	Pokec_n	Pokec_z	Income
#Nodes	30,000	66,569	67,797	14,821
#Features	13	266	277	14
#Edges	1,436,858	729,129	882,765	100,483
Node label	Future default	Working field	Working field	Income level
Sensitive attribute	Age	Region	Region	Race
Avg. degree	95.79	16.53	19.23	13.6

Experimental setup. We test GraphPAR on the node classification task. These are common graph datasets with sensitive attributes collected from various domains. We choose four public datasets *Income* [2], *Credit* [1], *Pokec_z* and *Pokec_n* [7]. Datasets statistics refer to Table 1 and more details refer to Appendix C.1. Implementation details of GrahPAR refer to Appendix C.2. We report the experiment results over five runs with different random seeds.

Baselines. We compare GraphPAR to four baselines: vanilla GCN [23], graph fairness methods FairGNN [7], NIFTY [1], and EDITS [9]. We choose contrastive pre-training DGI [39] and GCA [50], as well as predictive pre-training EdgePred [14] as the backbone of GraphPAR. More details of baselines refer to Appendix C.3.

6.1 Prediction Performance and Fairness (RQ1)

We choose accuracy (ACC) and macro-F1 (F1) to measure how well the nodes are classified, demographic parity (DP) and equality of opportunity (EO) to measure how fair the classification is. The results are shown in Table 2; additional results on Income refer to Table 4 in Appendix D.1. We interpret the results as follows:

- GraphPAR outperforms baseline models both in classification and fairness performance. GraphPAR is demonstrated to be superior in both classification and fairness performances, enhancing existing PGM models and outperforming other graph fairness methods. This result supports the effectiveness of GraphPAR in addressing fairness issues in the embedding space: (1) Powerful pre-training strategies enable the embeddings to include intrinsic information for downstream tasks. (2) Since PGMs also capture sensitive attribute information, the sensitive semantics vector can be effectively constructed. (3) Augmenting in the embedding space is independent of task labels; thus, the sensitive semantic augmenter does not corrupt the downstream performance.
- Performance of GraphPAR varies among different PGMs. The prediction performance and fairness vary when choosing different PGMs as the backbone. Usually, we observe that contrastive pre-training methods DGI and GCA perform better than the predictive method EdgePred, implying the performance of GraphPAR is positively related to the semantic capture ability of PGMs.
- RandAT and MinMax perform well but in different ways. It is worth mentioning that RandAT often achieves the best result on classification while MinMax often performs the best on fairness. The following differences in the training schemes directly lead to the result above: (1) In downstream classification loss, RandAT utilizes all augmented samples, while MinMax only utilizes the original sample. As a result, RandAT often outperforms MinMax on classification metrics ACC and F1. We regard that classification benefits from data augmentation [33], as these augmented samples share the same task-related semantics as the original samples,

Table 2: Performance and fairness ($\%\pm\sigma$) on node classification. The best results are in bold and runner-up results are underlined.

Method		Credit				Pokec_z				Pokec_n			
		ACC (↑)	F1 (↑)	DP (↓)	EO (↓)	ACC (↑)	F1 (↑)	DP (↓)	EO (↓)	ACC (↑)	F1 (↑)	DP (↓)	EO (↓)
	GCN	69.73±0.04	79.14±0.02	13.28±0.15	12.66±0.24	67.54±0.48	68.93±0.39	5.51±0.67	4.57±0.29	**70.11±0.34**	67.37±0.38	3.19±0.86	2.93±0.95
	FairGNN	72.50±4.09	81.80±3.86	9.20±3.35	7.64±3.58	67.47±1.12	69.35±3.14	1.91±1.01	1.04±1.11	68.42±2.04	64.34±2.32	1.41±1.30	1.50±1.23
	NIFTY	70.89±0.59	80.23±0.54	9.93±0.59	8.79±0.71	65.83±3.90	66.99±4.26	5.47±2.13	2.64±1.02	68.97±1.21	66.77±1.27	1.68±0.90	1.38±0.91
	EDITS	66.80±1.03	76.64±1.13	10.21±1.14	8.78±1.15	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
DGI	Naive	75.72±2.18	84.73±2.00	7.87±2.22	6.51±2.79	67.87±0.51	70.23±0.80	4.69±1.95	3.03±1.34	68.58±1.22	65.66±1.37	3.58±3.09	4.99±3.68
	GraphPAR$_{RandAT}$	**76.88±1.33**	**85.85±1.36**	5.93±2.91	4.44±3.34	67.05±1.33	**70.50±0.69**	1.90±1.22	0.84±0.28	68.92±1.55	65.61±1.33	**1.19±0.65**	2.11±1.60
	GraphPAR$_{MinMax}$	74.37±2.91	83.46±2.64	**3.81±2.37**	**2.60±2.48**	**68.32±0.55**	68.35±2.38	1.64±0.78	**0.53±0.39**	68.43±0.55	**68.20±2.22**	1.73±0.76	1.11±0.88
EdgePred	Naive	69.66±1.74	79.30±1.63	7.89±2.28	6.67±2.42	67.33±0.44	69.17±0.52	6.00±3.04	3.95±2.52	68.60±0.53	65.56±0.79	2.48±0.86	5.29±2.71
	GraphPAR$_{RandAT}$	69.97±2.35	79.55±2.24	6.36±2.19	4.83±2.70	66.87±1.12	68.86±0.46	1.99±1.12	2.27±1.23	68.49±1.41	65.45±1.02	1.79±0.85	3.69±0.68
	GraphPAR$_{MinMax}$	68.53±1.23	78.19±1.14	5.10±2.31	4.52±2.17	67.51±0.55	69.03±0.82	**1.45±1.40**	1.15±0.85	69.10±0.91	65.00±1.10	1.28±0.97	3.31±2.06
GCA	Naive	75.28±0.51	84.35±0.47	8.56±0.97	6.21±0.90	67.63±0.44	70.24±0.98	7.68±2.19	4.82±1.43	67.85±1.23	65.81±1.35	2.90±2.61	3.23±1.05
	GraphPAR$_{RandAT}$	75.50±1.29	84.66±1.27	5.51±2.44	3.98±1.96	66.73±2.22	70.32±0.73	4.23±2.50	2.94±1.84	68.11±0.44	64.43±1.05	2.35±1.12	2.42±1.62
	GraphPAR$_{MinMax}$	73.74±2.01	82.96±1.74	4.90±1.90	2.96±1.66	66.59±1.28	68.74±1.17	2.33±2.28	2.42±1.72	68.11±0.70	65.49±1.57	1.41±0.86	**0.94±0.59**

which helps the adapter and classifier further capture task-related semantics. (2) To debias sensitive information, MinMax minimizes the largest distance between an individual $\mathbf{h}_i$ and other samples $\mathbf{h}'_i$ in the sensitive augmentation set $\mathcal{S}_i$, which can achieve a better debiasing result against the sampling strategy in RandAT that performs adversarial training on all augmented samples.

These empirical findings straightforwardly demonstrate the characteristics of RandAT in Equation 7 and MinMax in Equation 8.

6.2 Debiasing Guarantee (RQ2)

To additionally guarantee how fair the classification is, we evaluate the provable fairness of GraphPAR compared with methods without debiasing adaptation, i.e., naive PGMs. Here, with the smoothed adapter and classifier, the metrics are accuracy (ACC) and provable fairness (Prov_Fair) in Definition 3. The result is presented in Table 3; additional results on Income refer to Table 5 in Appendix D.1. We have the following observations:

• Different from naive PGMs that show little or nearly zero provable fairness, RandAT achieves much better provable fairness, and MinMax has its fairness guaranteed very well. According to Theorem 3 where the provable fairness of PGMs satisfies $d_{cs} < d_{rs}$, since d_{rs} is the same, but d_{cs} is different among training schemes: naive PGMs do not optimize d_{cs}, thus the fairness is nearly not guaranteed; RandAT conduct adversarial training by using many samples with different sensitive attribute semantics, which has a positive effect on minimizing d_{cs} but not in an explicit way; MinMax achieves the best provable fairness by directly finding and optimizing d_{cs} with min-max training.

• The classification performances of RandAT and MinMax are competitive to naive PGMs. On the one hand, RandAT does not lose its classification performance because its augmentation is performed in sensitive semantics and does not introduce noise to task-related information; on the other hand, MinMax trains the downstream classifier with original data, implying that the adapter almost has no adverse effect on the classification while guaranteeing fairness.

In conclusion, the empirical results above support that when trained with RandAT and MinMax, GraphPAR guarantees fairness without compromising its classification performance.

6.3 The Effectiveness of $\boldsymbol{\alpha}$ (RQ3)

To verify whether $\boldsymbol{\alpha}$ satisfies our expectations, i.e., moving in the direction of $\boldsymbol{\alpha}$ increases the presence of the sensitive attribute while moving in the opposite direction diminishes its presence. First, we divide $\mathbf{H}$ into training and test sets. We take the training set to train a sensitive attribute classifier d_{sens} and compute $\boldsymbol{\alpha}$ by Equation 3. Next, we randomly construct 100 vectors at angles of 30, 60, and 90 to $\boldsymbol{\alpha}$, respectively. We use $\boldsymbol{\alpha}'$ to denote the vector with different angles and the same size as $\boldsymbol{\alpha}$. Then, we move the node representations in the test set along the direction of $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$ with varying augmentation degree t. Lastly, we utilize the classifier d_{sens} to predict the accuracy of the sensitive attribute on the test set. We report the average accuracy at different angles, and the results are presented in Figure 3, revealing the following findings:

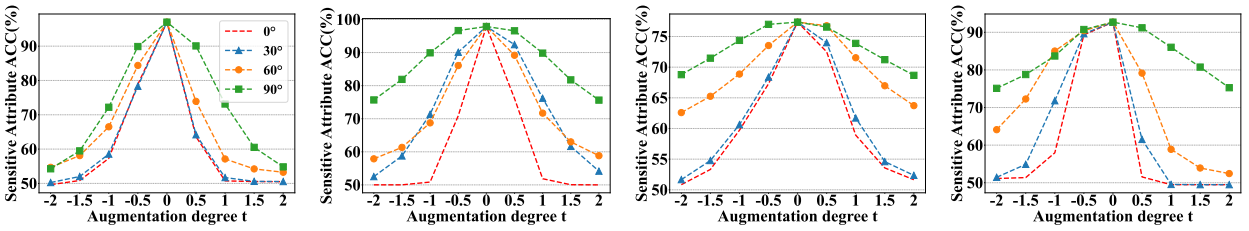
Table 3: Provable fairness under different training schemes.

Dataset	PGM	Naive		GraphPAR$_{RandAT}$		GraphPAR$_{MinMax}$	
		ACC (↑)	Prov_Fair (↑)	ACC (↑)	Prov_Fair (↑)	ACC (↑)	Prov_Fair (↑)
Credit	DGI	72.80	27.63	**75.39**	37.05	72.71	**89.59**
	EdgePred	66.87	5.41	**67.02**	44.20	66.41	**96.28**
	GCA	72.86	0.28	**73.25**	20.26	70.10	**92.92**
Pokec_z	DGI	**67.30**	1.47	67.21	10.99	67.28	**94.51**
	EdgePred	66.02	0	66.27	37.51	**66.80**	**90.97**
	GCA	**66.92**	13.9	66.67	16.14	65.22	**95.75**
Pokec_n	DGI	**68.45**	0.70	67.52	0.52	68.38	**77.97**
	EdgePred	67.58	0	**68.15**	21.17	**68.15**	**88.76**
	GCA	67.49	17.80	**67.52**	10.03	67.30	**91.16**

• When no movement is performed, i.e., $t = 0$, the accuracy is the highest. This again demonstrates that the pre-training inevitably captures the sensitive attribute information present in the dataset.

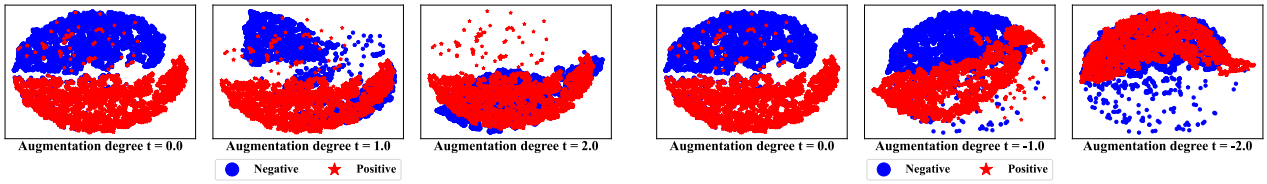
• For the original $\boldsymbol{\alpha}$, i.e., the angle is 0 (red dotted line), as the augmentation degree $|t|$ increases, the prediction accuracy of d_{sens} gradually decreases until it reaches 50%. This is because modifying the node representations along the same sensitive semantics direction makes all nodes increasingly similar in sensitive attribute semantics. For instance, when moving toward $t > 0$, nodes initially classified as negative samples move to positive samples, while nodes previously classified as positive samples remain positive. Consequently, the classifier d_{sens} can only accurately classify half of the nodes. To gain a more concrete understanding, we also visualized the augmentation process using t-SNE. The visualization results are depicted in Figure 4; more results refer to the Appendix D.2.

• For the $\boldsymbol{\alpha}'$ that has different angles and the same size as $\boldsymbol{\alpha}$, we observe that as the angle increased, the impact of augmentation on sensitive attribute semantics decreased, meaning the change in accuracy of the sensitive attribute classifier d_{sens} was smaller. This also validates the effectiveness of the direction of $\boldsymbol{\alpha}$.



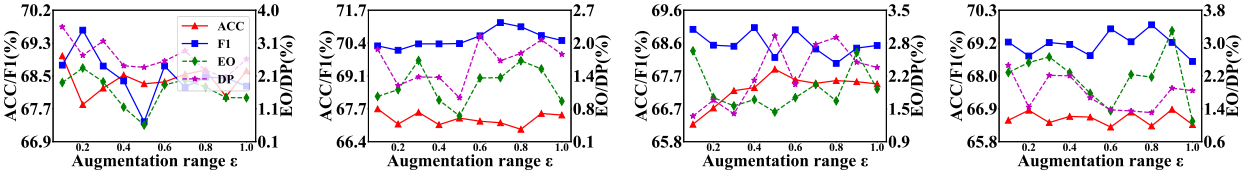
(a) Pokec_n. (b) Pokec_z. (c) Credit. (d) Income.

Figure 3: Sensitive attribute prediction accuracy under different augmentation degree t.



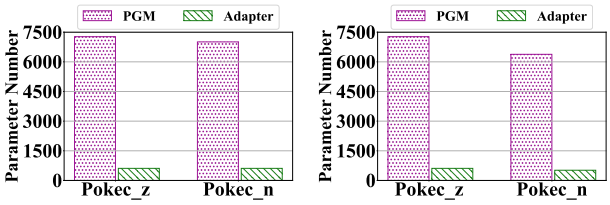
(a) Representations gradually move from negative to positive. (b) Representations gradually move from positive to negative.

Figure 4: The effect of augmentation degree t to node representations on pokec_z.



(a) DGI-MinMax. (b) DGI-RandAT. (c) EdgePred-MinMax. (d) EdgePred-RandAT.

Figure 5: The effect of augmentation range ϵ to GraphPAR$_{minmax}$ and GraphPAR$_{RandAT}$ on Pokec_z.



(a) Infomax. (b) EdgePred.

Figure 6: Comparison of PGMs and Adapter in the number of parameters tuned.

6.4 Hyperparameter Sensitivity Analysis (RQ4)

To further validate how the hyperparameters impact the performance of GraphPAR, we conduct hyperparameter sensitivity analysis experiments on the augmentation range ϵ, augmentation sample number k, and fairness loss scale λ. The best hyperparameter ϵ, k, λ for fairness metrics varies among PGMs, datasets, and training methods (RandAT and MinMax). Still, they consistently outperform naive PGMs, illustrating the effectiveness of GraphPAR in improving the fairness of PGMs. For example, as shown in Figure 5, on Pokec_z trained with MinMax, GraphPAR on DGI achieves the best fairness when $\epsilon = 0.5$, while $\epsilon = 0.3$ for EdgePred. A key observation is that when ϵ is tuned between 0 and 1, ACC and F1 are stable, while EO and DP fluctuate. This suggests the sensitive semantic augmenter does not corrupt task-related information while successfully capturing sensitive attribute information. More experiment results and analysis refer to the Appendix D.3.

6.5 Efficiency Analysis (RQ5)

We demonstrate the parameter efficiency of GraphPAR by comparing the parameters in PGMs with the adapter. As shown in Figure 6, the number of tuned parameters in GrahpPAR is 91% smaller than in the PGM. By contrast, since the parameter of the GNN encoder has to be tuned in existing fair methods, the number of tuned parameters would be equal to or even larger than the size of PGMs, far exceeding that in GraphPAR. In conclusion, GraphPAR is super parameter-efficient, which is well-suited for PGMs.

7 CONCLUSION

In this work, we explore fairness in PGMs for the first time. We discover that PGMs inevitably capture sensitive attribute semantics during pre-training, resulting in unfairness in downstream tasks. To address this problem, we propose GraphPAR to efficiently and flexibly endow PGMs with fairness during the adaptation for downstream tasks. Furthermore, with GraphPAR, we provide theoretical guarantees for fairness. Extensive experiments on real-world datasets demonstrate the effectiveness of GraphPAR in achieving fair predictions and providing provable fairness. In the future, we will further explore other trustworthy directions of PGMs.

ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China (No. U20B2045, 61772082, 62002029, 62192784, 62172052, U1936104) and Young Elite Scientists Sponsorship Program (No. 2023QNRC001) by CAST.

REFERENCES

[1] Chirag Agarwal, Himabindu Lakkaraju, and Marinka Zitnik. 2021. Towards a unified framework for fair and stable graph representation learning. In *Uncertainty in Artificial Intelligence*. PMLR, 2114–2124.

[2] Arthur Asuncion and David Newman. 2007. UCI machine learning repository.

[3] April Chen, Ryan A Rossi, Namyong Park, Puja Trivedi, Yu Wang, Tong Yu, Sungchul Kim, Franck Dernoncourt, and Nesreen K Ahmed. 2023. Fairness-aware graph neural networks: A survey. *arXiv preprint* (2023).

[4] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*. PMLR, 1310–1320.

[5] EU COM. 2021. Laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts. Proposal for a regulation of the European parliament and of the council.

[6] Elliot Creager, David Madras, Jörn-Henrik Jacobsen, Marissa Weis, Kevin Swersky, Toniann Pitassi, and Richard Zemel. [n. d.]. Flexibly fair representation learning by disentanglement. In *International conference on machine learning*.

[7] Enyan Dai and Suhang Wang. 2021. 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*.

[8] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint* (2022).

[9] Yushun Dong, Ninghao Liu, Brian Jalaian, and Jundong Li. 2022. Edits: Modeling and mitigating data bias for graph neural networks. In *Proceedings of the ACM Web Conference 2022*.

[10] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. 2019. Exploring the landscape of spatial robustness. In *International conference on machine learning*.

[11] Michael Gira, Ruisu Zhang, and Kangwook Lee. 2022. Debiasing pre-trained language models via efficient fine-tuning. In *Proceedings of the Second Workshop on Language Technology for Equality, Diversity and Inclusion*.

[12] Zhimeng Guo, Jialiang Li, Teng Xiao, Yao Ma, and Suhang Wang. 2023. Improving Fairness of Graph Neural Networks: A Graph Counterfactual Perspective. *arXiv preprint* (2023).

[13] Zhiwei Guo and Heng Wang. 2020. A deep graph neural network-based mechanism for social recommendations. *IEEE Transactions on Industrial Informatics* (2020).

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

[15] Bowen Hao, Jing Zhang, Hongzhi Yin, and Cuiping Li. 2021. Pre-training graph neural networks for cold-start users and items representation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*.

[16] Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, Evgeny Kharlamov, and Jie Tang. 2023. GraphMAE2: A Decoding-Enhanced Masked Self-Supervised Graph Learner. In *Proceedings of the ACM Web Conference 2023*.

[17] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[18] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *International Conference on Learning Representations*.

[19] Elisa Jillson. 2021. Aiming for truth, fairness, and equity in your company's use of AI. *Federal Trade Commission* (2021).

[20] Nikola Jovanović, Mislav Balunovic, and Dimitar Iliev Dimitrov. 2023. FARE: Provably Fair Representation Learning with Practical Certificates. (2023).

[21] Krishnaram Kenthapadi, Benjamin Le, and Ganesh Venkataraman. 2017. Personalized job recommendation system at linkedin: Practical challenges and lessons learned. In *Proceedings of the eleventh ACM conference on recommender systems*.

[22] Moein Khajehnejad, Ahmad Asgharian Rezaei, Mahmoudreza Babaei, Jessica Hoffmann, Mahdi Jalili, and Adrian Weller. 2020. Adversarial graph embeddings for fair influence maximization over social networks. *arXiv preprint* (2020).

[23] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint* (2016).

[24] Aounon Kumar and Tom Goldstein. 2021. Center smoothing: Certified robustness for networks with structured outputs. *Advances in Neural Information Processing Systems* (2021).

[25] Shengrui Li, Xueting Han, and Jing Bai. 2023. AdapterGNN: Efficient Delta Tuning Improves Generalization Ability in Graph Neural Networks. *arXiv preprint* (2023).

[26] Hongyi Ling, Zhimeng Jiang, Youzhi Luo, Shuiwang Ji, and Na Zou. 2022. Learning fair graph representations via automated data augmentations. In *The Eleventh International Conference on Learning Representations*.

[27] Yaoqi Liu, Cheng Yang, Tianyu Zhao, Hui Han, Siyuan Zhang, Jing Wu, Guangyu Zhou, Hai Huang, Hui Wang, and Chuan Shi. 2023. GammaGL: A Multi-Backend Library for Graph Neural Networks. (2023).

[28] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. 2023. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference 2023*.

[29] Jing Ma, Ruocheng Guo, Mengting Wan, Longqi Yang, Aidong Zhang, and Jundong Li. 2022. Learning fair node representations with graph counterfactual fairness. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*.

[30] Farzan Masrour, Tyler Wilson, Heng Yan, Pang-Ning Tan, and Abdol Esfahanian. 2020. Bursting the filter bubble: Fairness-aware network link prediction. In *Proceedings of the AAAI conference on artificial intelligence*.

[31] Nicholas Meade, Elinor Poole-Dayan, and Siva Reddy. 2022. An Empirical Survey of the Effectiveness of Debiasing Techniques for Pre-trained Language Models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

[32] L. Oneto and Silvia Chiappa. 2020. Fairness in Machine Learning. *ArXiv* abs/2012.15816 (2020).

[33] Luis Perez and Jason Wang. 2017. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint* (2017).

[34] Momchil Peychev, Anian Ruoss, Mislav Balunović, Maximilian Baader, and Martin Vechev. [n. d.]. Latent space smoothing for individually fair representations. In *European Conference on Computer Vision*.

[35] Anian Ruoss, Mislav Balunovic, Marc Fischer, and Martin Vechev. 2020. Learning certified individually fair representations. *Advances in neural information processing systems* (2020).

[36] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. 2019. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *International Conference on Learning Representations*.

[37] Mengying Sun, Jing Xing, Huijun Wang, Bin Chen, and Jiayu Zhou. 2021. Mocl: Contrastive learning on molecular graphs with multi-level domain knowledge. *arXiv preprint* (2021).

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

[39] Petar Veličković, William Fedus, WilliamL. Hamilton, Pietro Liò, Yoshua Bengio, and RDevon Hjelm. 2018. Deep Graph Infomax. *International Conference on Learning Representations,International Conference on Learning Representations* (2018). https://doi.org/10.17863/cam.40744

[40] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community preserving network embedding. In *Proceedings of the AAAI conference on artificial intelligence*.

[41] Y Wang, J Wang, Z Cao, and AB Farimani. [n. d.]. MolCLR: Molecular contrastive learning of representations via graph neural networks. *arXiv preprint* ([n. d.]).

[42] Yu Wang, Yuying Zhao, Yushun Dong, Huiyuan Chen, Jundong Li, and Tyler Derr. 2022. 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*.

[43] Fei Wu, Xiao-Yuan Jing, Pengfei Wei, Chao Lan, Yimu Ji, Guo-Ping Jiang, and Qinghua Huang. 2022. Semi-supervised multi-view graph convolutional networks with application to webpage classification. *Information Sciences* (2022).

[44] Yiqing Wu, Ruobing Xie, Yongchun Zhu, Fuzhen Zhuang, Ao Xiang, Xu Zhang, Leyu Lin, and Qing He. 2022. Selective fairness in recommendation via prompts. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[45] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 1 (2020).

[46] Jun Xia, Lirong Wu, Jintao Chen, Bozhen Hu, and Stan Z Li. 2022. Simgrace: A simple framework for graph contrastive learning without data augmentation. In *Proceedings of the ACM Web Conference 2022*.

[47] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. 2021. Graph contrastive learning automated. In *International Conference on Machine Learning*.

[48] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in neural information processing systems* (2020).

[49] Xu Zhang, Liang Zhang, Bo Jin, and Xinjiang Lu. 2021. A multi-view confidence-calibrated framework for fair and stable graph representation learning. In *2021 IEEE International Conference on Data Mining (ICDM)*.

[50] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*.

A ALGORITHM

We present the whole algorithm process of GraphPAR as follows:

Algorithm 1: GraphPAR

Data: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, pre-trained graph model f
Result: Adapter g and classifier d, and the provable fairness of each node

1. GraphPAR Training:
Compute the sensitive semantic vector $\boldsymbol{\alpha}$ as Eq 3;
for *each epoch* **do**
  Sample the augmentation set $\hat{\mathcal{S}}_i$ for each node i as Eq 6;
  if *Train with RandAT* **then**
    Compute $\mathcal{L}$ by Eq 7;
  else
    Compute $\mathcal{L}$ by Eq 11;
  end
  Backward pass with $\mathcal{L}$;
end
2. Provide Provable Fairness with Smoothing:
Do adversarial training on the classifier d;
Construct the smoothed adapter $\widehat{g}$ by Eq 12 and the smoothed classifier $\widehat{d}$ by Eq 15;
for *each node i in $\mathcal{V}$* **do**
  Compute the guarantee $d_{cs,i}$ of the adapter as Theorem 1;
  Compute the guarantee $d_{rs,i}$ of the classifier as Theorem 2;
  If $d_{cs,i} < d_{rs,i}$, then node i has a provable fairness;
end

B PROOF OF THEOREM 3

Proof. Recall the definition of $g_{\mathbf{h}}(t) := g(\mathbf{h} + t \cdot \boldsymbol{\alpha})$ and note that for $\mathbf{h}' = \mathbf{h} + t' \cdot \boldsymbol{\alpha}$, the center smoothing of

$$\widehat{g_{\mathbf{h}'}}(t) \sim g_{\mathbf{h}'}(t + \mathcal{N}(0, \sigma_{cs}^2)) = g(\mathbf{h}' + (t + \mathcal{N}(0, \sigma_{cs}^2)) \cdot \boldsymbol{\alpha}),$$

$$\widehat{g_{\mathbf{h}}}(t + t') \sim g_{\mathbf{h}}(t + t' + \mathcal{N}(0, \sigma_{cs}^2)) = g(\mathbf{h} + (t + t' + \mathcal{N}(0, \sigma_{cs}^2)) \cdot \boldsymbol{\alpha}).$$

Since $\mathbf{h}' = \mathbf{h} + t' \cdot \boldsymbol{\alpha}$, the sampling distributions are the same, hence $\widehat{g_{\mathbf{h}'}}(t) = \widehat{g_{\mathbf{h}}}(t + t')$, and in particular $\widehat{g}(\mathbf{h}') = \widehat{g_{\mathbf{h}'}}(0) = \widehat{g_{\mathbf{h}}}(t')$.

Now, let us get back to Equation 19. By definition of $\mathcal{S}$, for all $\mathbf{h}' \in \mathcal{S}$, $\mathbf{h}' = \mathbf{h} + t' \cdot \boldsymbol{\alpha}$ for some $t' \in [-\epsilon, \epsilon]$. Moreover, $\mathbf{z}_{cs} = \widehat{g}(\mathbf{h}) = \widehat{g_{\mathbf{h}}}(0)$ and $\widehat{g}(\mathbf{h}') = \widehat{g_{\mathbf{h}}}(t')$. Theorem 1 tells us that with confidence $1 - \alpha_{cs}$:

$$\begin{aligned} &\left\|\widehat{g_{\mathbf{h}}}(0) - \widehat{g_{\mathbf{h}}}(t')\right\|_2 \leq d_{cs},\ \forall\, t' \in [-\epsilon, \epsilon] \\ \Longleftrightarrow\ &\left\|\mathbf{z}_{cs} - \widehat{g}(\mathbf{h}')\right\|_2 \leq d_{cs},\ \forall\, \mathbf{h}' \in \mathcal{S}, \end{aligned} \tag{20}$$

provided that the center smoothing computation of d_{cs} does not abstain.

Finally, we consider the last component of the pipeline, i.e., the smoothed classifier $\widehat{d}$. Provided that $\widehat{d}$ does not abstain at the input d_{cs}, Theorem 2 provides us with a radius d_{rs} around $\mathbf{z}_{cs}$ such that with confidence $1 - \alpha_{rs}$:

$$\begin{aligned} &\widehat{d}(\mathbf{z}_{cs}) = \widehat{d}(\mathbf{z}_{cs} + \boldsymbol{\delta}),\ \forall \boldsymbol{\delta} \text{ s.t. } \|\boldsymbol{\delta}\|_2 < d_{rs} \\ \Longleftrightarrow &\widehat{d}(\mathbf{z}_{cs}) = \widehat{d}(\mathbf{z}'),\ \forall \mathbf{z}' \text{ s.t. } \left\|\mathbf{z}_{cs} - \mathbf{z}'\right\|_2 < d_{rs}. \end{aligned} \tag{21}$$

If $d_{cs} < d_{rs}$, combining the conclusions in Equation 20 and Equation 21 and applying the union bound, we obtain that with confidence $1 - \alpha_{cs} - \alpha_{rs}$ we have $\widehat{d}(\mathbf{z}_{cs}) = \widehat{d}(\widehat{g}(\mathbf{h}'))$ for all $\mathbf{h}' \in \mathcal{S}$, that is,

$$\forall\, \mathbf{h}' \in \mathcal{S}(\mathbf{h}) : \widehat{d} \circ \widehat{g}(\mathbf{h}) = \widehat{d} \circ \widehat{g}(\mathbf{h}') \tag{22}$$

as required by Definition 3. The same proof technique can extend to the multiple sensitive attribute vectors case. □

C MORE EXPERIMENTAL DETAILS

C.1 Datasets

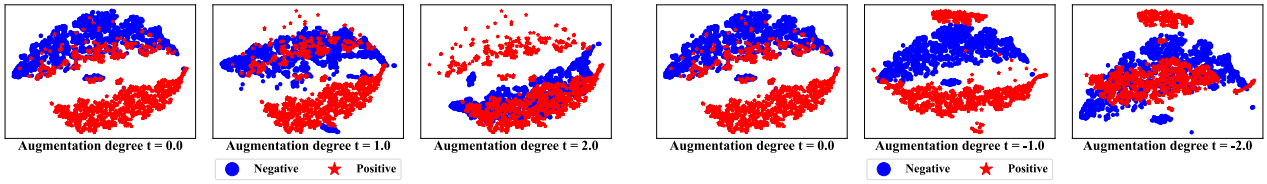
Detailedly, *Income* [2] is collected from the Adult Data Set. The sensitive attribute is race, and the task is to classify whether an individual salary exceeds 50,000$. *Credit* [1] encompasses a network of individuals connected according to the likeness of their spending and payment habits. The sensitive attribute is the age of these individuals, and the objective is to predict whether their default payment method is credit card or not. *Pokec_z* and *Pokec_n* [7] are created by sampling from Pokec based on geographic regions. Pokec encompasses anonymized data from the complete social network in 2012. The sensitive attribute is the region, and the predicted label is the working field.

C.2 Implementations

Unless otherwise specified, we set the hyperparameters as follows: For the sensitive semantics augmented, sensitive attribute semantics augmentation range $\epsilon = 0.5$, number of randomly selected augmentation samples $k = 20$, fairness loss scale factor $\lambda = 0.1$. For the adapter, the dimension size of the down projection is half of the input, the learning rate is 0.01, and the training epoch is 1000. We use GCN as the backbone for all PGMs and take the Adam optimizer. Referring to random smooth [4], after adapting PGMs to downstream tasks, we uniformly utilize Gaussian data augmentation with a variance of 1 to additionally adversarial train the classifier for 100 rounds, which maximizes the number of nodes with provable fairness without compromising accuracy. Following the parameter settings in center smooth [24] and random smooth [4], we utilize the well-trained adapter and the classifier to construct their smoothed versions, respectively. In the future, we will also provide an implementation based on GammaGL [27] at https://github.com/BUPT-GAMMA/GammaGL.

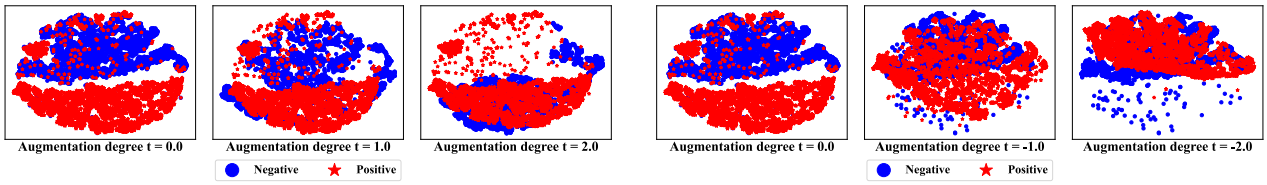
C.3 Baselines

We compare GraphPAR to four baseline models: GCN [23] is the most common GNN; FairGNN [7] is a framework for fair node classification using GNNs given limited sensitive attribute information; NIFTY [1] achieves fairness by maximizing the similarity of representations learned from the original graph and their augmented counterfactual graphs. EDITS [9] debiases the input network to remove the sensitive information in the graph data. Since GraphPAR is based on PGMs, we include three types of PGM as baseline models: contrastive pre-training models DGI [39] and GCA [50] that maximize the mutual information between different views, as well as predictive pre-training model EdgePred [14] that reconstructs masked edges as its task.



(a) Representations gradually move from negative to positive. **(b) Representations gradually move from positive to negative.**

Figure 7: The effect of augmentation degree t to node representations on Income.



(a) Representations gradually move from negative to positive. **(b) Representations gradually move from positive to negative.**

Figure 8: The effect of augmentation degree t to node representations on pokec_n.

Table 4: Performance and fairness (% ± σ) on node classification. The best results are in bold and runner-up results are underlined.

Method		Income			
		ACC (↑)	F1 (↑)	DP (↓)	EO (↓)
	GCN	69.08±0.35	**49.39±0.13**	29.73±1.43	33.54±3.43
	FairGNN	68.90±1.49	47.26±0.70	15.39±2.45	21.51±3.59
	NIFTY	70.37±1.86	47.87±0.33	26.84±1.27	29.09±1.53
	EDITS	69.02±0.59	49.21±0.37	27.11±2.76	31.11±4.23
DGI	Naive	76.62±0.60	48.15±2.35	23.07±5.81	30.26±7.59
	GraphPAR$_{RandAT}$	**76.63±1.10**	46.94±1.19	15.43±4.48	19.80±7.93
	GraphPAR$_{MinMax}$	75.29±1.60	47.27±1.08	**8.75±1.33**	**7.90±3.84**
EdgePred	Naive	69.15±2.02	46.34±2.63	29.73±3.19	35.79±7.83
	GraphPAR$_{RandAT}$	70.41±2.03	45.51±3.27	23.51±7.42	28.40±13.50
	GraphPAR$_{MinMax}$	69.06±3.69	46.58±1.28	11.68±7.06	14.18±7.65
GCA	Naive	75.00±2.10	46.91±3.76	21.52±6.25	27.73±9.08
	GraphPAR$_{RandAT}$	74.95±1.41	46.55±2.72	16.72±3.80	22.30±6.05
	GraphPAR$_{MinMax}$	75.44±1.79	46.70±1.74	9.92±3.75	16.99±4.56

D ADDITIONAL EXPERIMENT ANALYSIS

D.1 Effectiveness of GraphPAR on Income

As shown in Table 4 and Table 5, similar to performance on the Credit, Pokec_n, and Pokec_z datasets, GraphPAR outperforms baseline models in terms of classification performance and fairness. By employing the two proposed adapter tuning methods, GraphPAR significantly enhances the fairness of PGMs in downstream tasks without nearly compromising prediction performance. Moreover, based on GraphPAR$_{MinMax}$, around 83% of nodes exhibit provable fairness.

D.2 More Visualization Results on α

To gain a more concrete understanding of the augmentation process in the direction of α, we also visualized the augmentation process using t-SNE on Income and Pokec_n datasets, and the visualization results depicted in Figure 7 and Figure 8, respectively.

Table 5: Provable fairness under different training schemes.

Dataset	PGM	Naive		GraphPAR$_{RandAT}$		GraphPAR$_{MinMax}$	
		ACC (↑)	Prov_Fair (↑)	ACC (↑)	Prov_Fair (↑)	ACC (↑)	Prov_Fair (↑)
Income	DGI	72.85	0.02	**73.42**	0.94	73.19	**81.01**
	EdgePred	67.17	0.01	**68.24**	7.64	66.09	**80.62**
	GCA	70.91	0.45	**71.59**	5.84	72.47	**90.68**

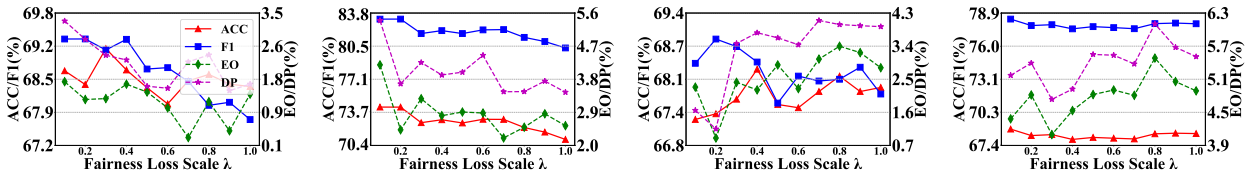
D.3 More Hyperparameter Analysis

We conduct a more detailed hyperparameter sensitivity analysis for GraphPAR, focusing on three key hyperparameters: the augmentation range ϵ, the augmentation sample number k, and the fairness loss scale λ. They play a crucial role in shaping the prediction performance and fairness of GraphPAR, and understanding their sensitivity is vital for finding the best model for prediction performance and fairness.

Augmentation range sensitivity (ϵ). The augmentation range ϵ dictates the range of linear interpolation on sensitive attribute semantics. Empirically, we find that the range of [0.2,0.4,0.8,1.0] works well for all datasets. An ϵ larger than 1 would probably harm the prediction accuracy. Within a certain range, the larger the augmentation range ϵ, i.e., the larger the range of sensitive attributes considered, the model fairer. For example, as depicted in Figure 10 (a), when the PGM is DGI and the debiasing method is MinMax, the metrics of DP and EO tend to decrease with increasing ϵ on the Credit dataset.

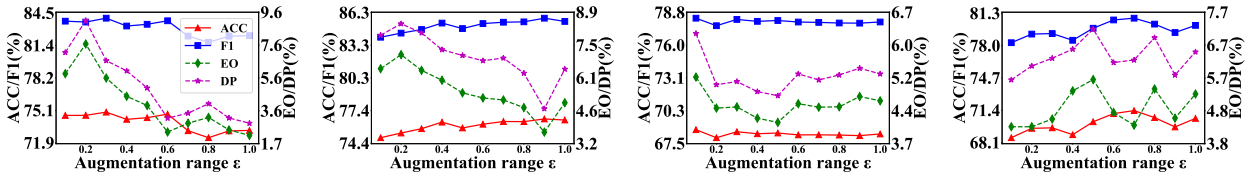
Fairness loss scale factor sensitivity (λ). λ is a scale factor for balancing accuracy and fairness. We find that different pre-training methods require different values of λ. As depicted in Figure 9, when the PGM is DGI, the optimal λ is 0.7 in the Pokec_z and Credit datasets. However, the optimal λ is 0.2 when the PGM is EdgePred.

Augmentation sample number sensitivity (k). k is the augmentation sample number for each node. According to Figure 11 and Figure 12, the optimal k is associated with the dataset, the pre-training method, and the adapter training strategy, but the general RandAT requires a larger k value than MinMax.



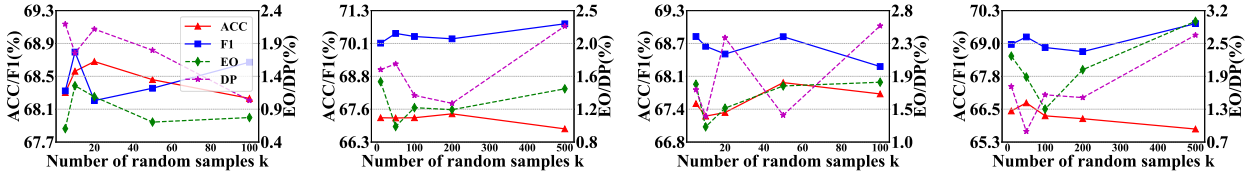
(a) DGI-Pokec_z. (b) DGI-Credit. (c) EdgePred-Pokec_z. (d) EdgePred-Credit.

Figure 9: The effect of fairness loss scale factor λ to GraphPAR$_{minmax}$.



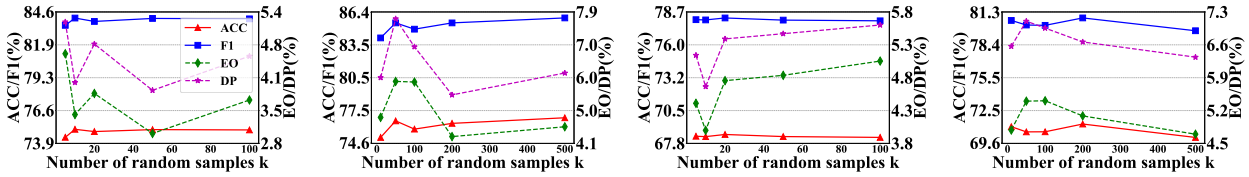
(a) DGI-MinMax. (b) DGI-RandAT. (c) EdgePred-MinMax. (d) EdgePred-RandAT.

Figure 10: The effect of augmentation range ϵ to GraphPAR$_{minmax}$ and GraphPAR$_{RandAT}$ in the Credit dataset.



(a) DGI-MinMax. (b) DGI-RandAT. (c) EdgePred-MinMax. (d) EdgePred-RandAT.

Figure 11: The effect of augmentation sample number k to GraphPAR$_{minmax}$ and GraphPAR$_{RandAT}$ in the Pokec_z dataset.



(a) DGI-MinMax. (b) DGI-RandAT. (c) EdgePred-MinMax. (d) EdgePred-RandAT.

Figure 12: The effect of augmentation sample number k to GraphPAR$_{minmax}$ and GraphPAR$_{RandAT}$ in the Credit dataset.