Interpreting Unfairness in Graph Neural Networks via Training Node Attribution

Yushun Dong[1], **Song Wang**[1], **Jing Ma**[1], **Ninghao Liu**[2], **Jundong Li**[1]

[1]University of Virginia
[2]University of Georgia
{yd6eb, sw3wv, jm3mr, jundong}@virginia.edu, ninghao.liu@uga.edu

Abstract

Graph Neural Networks (GNNs) have emerged as the leading paradigm for solving graph analytical problems in various real-world applications. Nevertheless, GNNs could potentially render biased predictions towards certain demographic subgroups. Understanding how the bias in predictions arises is critical, as it guides the design of GNN debiasing mechanisms. However, most existing works overwhelmingly focus on GNN debiasing, but fall short on explaining how such bias is induced. In this paper, we study a novel problem of interpreting GNN unfairness through attributing it to the influence of training nodes. Specifically, we propose a novel strategy named Probabilistic Distribution Disparity (PDD) to measure the bias exhibited in GNNs, and develop an algorithm to efficiently estimate the influence of each training node on such bias. We verify the validity of PDD and the effectiveness of influence estimation through experiments on real-world datasets. Finally, we also demonstrate how the proposed framework could be used for debiasing GNNs. Open-source code can be found at https://github.com/yushundong/BIND.

Introduction

Graph data is pervasive among a plethora of realms, e.g., financial fraud detection (Wang et al. 2019; Pourhabibi et al. 2020; Cheng et al. 2020), social recommendation (Fan et al. 2019; Song et al. 2019; Guo and Wang 2020), and chemical reaction prediction (Do, Tran, and Venkatesh 2019; Shi et al. 2020; Kwon et al. 2022). As one of the state-of-the-art approaches to handle graph data, Graph Neural Networks (GNNs) have been attracting increasing attention (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Veličković et al. 2017). Over the years, various graph analytical tasks have benefited from GNNs, where node classification is among the most widely studied ones (Kipf and Welling 2017; Wu et al. 2019, 2020). Nevertheless, in node classification, GNNs often yield results with discrimination towards specific demographic subgroups described by certain sensitive attributes (Dong et al. 2022a; Dai and Wang 2021a; Agarwal, Lakkaraju, and Zitnik 2021; Zhang et al. 2022b; Wang et al. 2022), such as gender, race, and religion. In many high-stake applications, critical decisions are made based on the classification results of GNNs (Shumovskaia et al. 2020), e.g., crime forecasting (Jin et al. 2020), and the exhibited bias (i.e., unfairness) is destructive for the involved individuals (Dong et al. 2022b,c; Song et al. 2022). To tackle this problem, there has been a line of works focusing on debiasing GNNs in node classification (Dong et al. 2022a; Dai and Wang 2021a; Agarwal, Lakkaraju, and Zitnik 2021; Dong et al. 2021; Loveland et al. 2022; Dai and Wang 2022). Their goal is to relieve the bias in GNN predictions on the test set and in this paper we refer to it as model bias.

In addition to debiasing GNNs, it is also critical to interpret how the model bias arises in GNNs. This is because such an understanding not only helps to determine whether a specific node should be involved in the training set, but also has much potential to guide the design of GNN debiasing methods (Dong et al. 2022a; Loveland et al. 2022; Li et al. 2021). Nevertheless, most existing GNN interpretation methods aim to understand how a prediction is made (Yuan et al. 2020b; Liu, Feng, and Hu 2022) instead of other aspects such as fairness. Consequently, although the graph data has been proved to be a significant source of model bias (Dong et al. 2022a; Li et al. 2021), existing works are unequipped to tackle this problem. In this paper, we aim to address this problem at the instance (node) level. Specifically, given a GNN trained for node classification, we aim to answer: "*To what extent the GNN model bias is influenced by the existence of a specific training node in this graph?*"

Nevertheless, answering the above question is technically challenging. Essentially, there are three main challenges: (1) *Influence Quantification.* To depict the influence of each training node on the model bias of GNNs, the first and foremost challenge is to design a principled fairness metric. A straightforward approach is to directly employ traditional fairness metrics (e.g., Δ_{SP} for *Statistical Parity* (Dwork et al. 2012) and Δ_{EO} for *Equal Opportunity* (Hardt, Price, and Srebro 2016a)). However, these metrics are not applicable in our task. The reason is that most of them are computed based on the predicted labels, while a single training node can barely twist these predicted labels on test data (Zhang et al. 2022a; Sun et al. 2020). Consequently, the influence of a single training node on the model bias would be hard to capture. (2) *Computation Efficiency.* To compute the influence of each training node on the model bias, a natural way is to re-train the GNN on a new graph with this specific training node being deleted and observe how the exhibited model bias changes. However, such a re-training process is prohibitively expensive. (3) *Non-I.I.D. Characterization.* Graph data goes against the widely

adopted i.i.d. assumption, as neighboring nodes are often dependent on each other (Ma, Deng, and Mei 2021; Ying et al. 2019). Therefore, when a specific node is deleted from the graph, all its neighbors could exert different influences on the model bias of GNN during training. Such complex dependencies bring obstacles towards the analysis of node influence on the model bias.

To tackle the above challenges, in this paper, we propose a novel framework named BIND (Biased traIning Node iDentification) to quantify and estimate the influence of each training node on the model bias of GNNs. Specifically, to handle the first challenge, we propose *Probabilistic Distribution Disparity* (PDD) as a principled strategy to quantify the model bias. PDD directly quantifies the exhibited bias in the GNN probabilistic output instead of the predicted labels. Therefore, PDD is with finer granularity and is more suitable for capturing the influence of each specific training node compared with traditional fairness metrics. To handle the second challenge, we propose an estimation algorithm for the node influence on model bias, which avoids the re-training process and thus achieves better efficiency. To tackle the third challenge, we also characterize the dependency between nodes based on the analysis of the training loss for GNNs. Finally, experiments on real-world datasets corroborate the effectiveness of BIND. Our contributions are mainly summarized as (1) **Problem Formulation.** We formulate a novel problem of interpreting the bias exhibited in GNNs through attributing to the influence of training nodes; (2) **Metric and Algorithm Design.** We propose a novel framework BIND to quantify and efficiently estimate the influence of each training node on the model bias of GNNs; (3) **Experimental Evaluation.** We perform comprehensive experiments on real-world datasets to evaluate the effectiveness of the proposed framework BIND.

Preliminaries

We first present the notations used in this paper. Then, we define the problem of interpreting GNN unfairness through quantifying the influence of each specific training node.

Notations. In this paper, matrices, vectors, and scalars are represented with bold uppercase letters (e.g., $\boldsymbol{A}$), bold lowercase letters (e.g., $\boldsymbol{x}$), and normal lowercase letters (e.g., n), respectively. We denote an input graph as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{X}\}$, where $\mathcal{V} = \{v_1, ..., v_n\}$ denotes the node set, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents the edge set, $\mathcal{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$ is the node attribute vectors, and $\boldsymbol{x}_i$ $(1 \leq i \leq n)$ represents the attribute vector of node v_i. We denote $\mathcal{G}_{-i}$ as the new graph with node v_i being deleted from $\mathcal{G}$. Additionally, we employ $\mathcal{V}'$ $(\mathcal{V}' \subseteq \mathcal{V})$ to represent the training node set, where $|\mathcal{V}'| = m$. The nodes in graph $\mathcal{G}$ are mapped to the output space with a trained GNN $f_{\boldsymbol{W}}$, where $\boldsymbol{W}$ represents the learnable parameters of the GNN model. We denote the optimized parameters (i.e., the parameters after training) as $\hat{\boldsymbol{W}}$. In node classification, the probabilistic classification output for the n nodes is denoted as $\hat{\mathcal{Y}} = \{\hat{\boldsymbol{y}}_1, ..., \hat{\boldsymbol{y}}_n\}$, where $\hat{\boldsymbol{y}}_i \in \mathbb{R}^c$, and c is the number of classes. We use Y and S to denote the ground truth label and the sensitive attribute for nodes, respectively. For an L-layer GNN $f_{\boldsymbol{W}}$, we define the subgraph up to L hops away centered on v_i as its computation graph (denoted as $\mathcal{G}_i = \{\mathcal{V}_i, \mathcal{E}_i, \mathcal{X}_i\}$). Here $\mathcal{V}_i, \mathcal{E}_i$, and $\mathcal{X}_i$ denote the set of nodes, edges, and node attributes in $\mathcal{G}_i$, respectively. It is worth noting that existing works have proven that $\mathcal{G}_i$ fully determines the information $f_{\boldsymbol{W}}$ utilizes to make the prediction of v_i (Ying et al. 2019). For node v_i, we use $\mathcal{V}_i'$ to indicate the intersection between $\mathcal{V}_i$ and $\mathcal{V}'$, i.e., $\mathcal{V}_i' = \mathcal{V}_i \cap \mathcal{V}'$, which is the set of training nodes in $\mathcal{G}_i$.

Problem Statement. The problem of interpreting GNN unfairness is formally defined as follows.

Problem 1. ***GNN Unfairness Interpretation.*** *Given the graph data $\mathcal{G}$ and a GNN model $f_{\hat{\boldsymbol{W}}}$ trained based on $\mathcal{G}$, we define the problem of interpreting GNN unfairness as to quantify the influence of each training node to the unfairness exhibited in GNN predictions on the test set.*

Methodology

In this section, we first briefly introduce GNNs for the node classification task. Then, to tackle the challenge of *Influence Quantification*, we propose Probabilistic Distribution Disparity (PDD) to measure model bias and define node influence on the bias in a trained GNN. Furthermore, to tackle the challenge of *Computation Efficiency*, we design an algorithm to estimate the node influence on the model bias. Finally, we introduce how to characterize the dependency between nodes in influence estimation, which tackles the challenge of *Non-I.I.D. Characterization*.

GNNs in Node Classification

In the node classification task, GNNs take the input graph $\mathcal{G}$ and output a probabilistic output matrix $\hat{\boldsymbol{Y}}$, where the i-th row in $\hat{\boldsymbol{Y}}$ is $\hat{\boldsymbol{y}}_i$, i.e., the probabilistic prediction of a node's membership over all possible classes. Usually, there are multiple layers in GNNs, where the formulation of the l-th layer can be summarized as:

$$\boldsymbol{z}_i^{(l+1)} = \sigma\left(\mathrm{AGG}\left(\boldsymbol{z}_i^{(l)}, h\left(\left\{\boldsymbol{z}_j^{(l)} : v_j \in \mathcal{N}(v_i)\right\}\right)\right)\right). \quad (1)$$

Here $\boldsymbol{z}_i^{(l)}$ is the embedding of node i at the l-th layer; $\mathcal{N}(v_i)$ is the set of one-hop neighbors around v_i; $h(\cdot)$ is a function with learnable parameters; $\mathrm{AGG}(\cdot)$ and $\sigma(\cdot)$ denote the aggregation function (e.g., mean operator) and activation function (e.g., ReLU), respectively. Later on, a loss function $L_{\mathcal{V}'}$ (e.g., cross-entropy loss) defined on the set of training nodes $\mathcal{V}'$ is employed for GNN training.

Probabilistic Distribution Disparity

Traditional bias metrics such as Δ_{SP} for statistical parity and Δ_{EO} for equal opportunity are computed on the predicted class labels. However, a single training node can hardly twist these predicted labels (Zhang et al. 2022a; Sun et al. 2020). Hence the node-level contribution to model bias can barely be captured by traditional bias metrics. To capture the influence of a single training node on model bias, we propose Probabilistic Distribution Disparity (PDD) as a novel bias quantification strategy. PDD can be instantiated with different fairness notions to depict the model bias from different perspectives. Specifically, we assume the population is divided into different sensitive subgroups, i.e., demographic subgroups described by the sensitive attribute. To achieve finer granularity, we define PDD as the Wasserstein-1 distance (Kantorovich 1960) between the probability distributions of a variable of

interest in different sensitive subgroups. Compared with traditional fairness metrics, continuous changes brought by each specific training node are reflected in the measured distributions, and Wasserstein distance is theoretically more sensitive to the change of the measured distributions over other commonly used distribution distance metrics (Arjovsky, Chintala, and Bottou 2017). In addition, we note that the variable of interest depends on the chosen fairness notion in applications, and a larger value of PDD indicates a higher level of model bias. We introduce two instantiations of PDD based on two traditional fairness notions, including *Statistical Parity* (Dwork et al. 2012) and *Equal Opportunity* (Hardt, Price, and Srebro 2016a). Both notions are based on binary classification tasks and binary sensitive attributes (generalizations to non-binary cases can be found in Appendix A[1]). For example, Statistical Parity requires the probability of positive predictions to be the same across two sensitive subgroups, where the variable of interest is the GNN probabilistic output $\hat{\boldsymbol{y}}$. We use $\hat{\mathcal{Y}}^{(S=j)}$ to denote the set of the probabilistic predictions for test nodes whose sensitive attribute S equals to j ($j \in \{0,1\}$). Let the distribution of the probabilistic predictions in $\hat{\mathcal{Y}}^{(S=0)}$ and $\hat{\mathcal{Y}}^{(S=1)}$ be $P_{\hat{\boldsymbol{y}}}^{(S=0)}$ and $P_{\hat{\boldsymbol{y}}}^{(S=1)}$, respectively. The PDD instantiated with statistical parity Γ_{SP} is

$$\Gamma_{SP} = \text{Wasserstein}_1(P_{\hat{\boldsymbol{y}}}^{(S=0)}, P_{\hat{\boldsymbol{y}}}^{(S=1)}), \tag{2}$$

where $\text{Wasserstein}_1(\cdot,\cdot)$ takes two distributions as input and outputs the Wasserstein-1 distance between them. Denote Y as the ground truth for node classification. Similarly, we can also instantiate PDD based on Equal Opportunity Γ_{EO} as

$$\Gamma_{EO} = \text{Wasserstein}_1(P_{\hat{\boldsymbol{y}}}^{(S=0,Y=1)}, P_{\hat{\boldsymbol{y}}}^{(S=1,Y=1)}). \tag{3}$$

$P_{\hat{\boldsymbol{y}}}^{(S=0,Y=1)}$ and $P_{\hat{\boldsymbol{y}}}^{(S=1,Y=1)}$ are model prediction distributions for nodes with $(S=0,Y=1)$ and $(S=1,Y=1)$, respectively. With such a strategy, we then define node influence on model bias.

Definition 1. ***Node Influence on Model Bias.*** *Let $f_{\hat{\boldsymbol{W}}}$ and $f_{\hat{\boldsymbol{W}}'}$ denote the GNN model trained on graph $\mathcal{G}$ and $\mathcal{G}_{-i}$ (i.e., $\mathcal{G}$ with node $v_i \in \mathcal{V}'$ being deleted), respectively. Let Γ_1 and Γ_2 be the Probabilistic Distribution Disparity value based on the output of $f_{\hat{\boldsymbol{W}}}$ and $f_{\hat{\boldsymbol{W}}'}$ for nodes in test set. We define $\Delta\Gamma = \Gamma_2 - \Gamma_1$ as the influence of node v_i on the model bias.*

The rationale behind this definition is to measure to what extent Γ changes if the GNN model is trained on a graph without v_i. Thus, $\Delta\Gamma$ depicts the influence of node v_i on the model bias. For both instantiations of Γ (i.e., Γ_{SP} and Γ_{EO}), if $\Delta\Gamma > 0$, deleting the training node v_i from $\mathcal{G}$ leads to a more unfair (or biased) GNN model. This indicates that node v_i contributes to improving the fairness level, i.e., v_i is helpful for fairness. Nevertheless, the above computation requires re-training the GNN to obtain the influence of each training node, which is too expensive if we want to compute the influence of all nodes in the training set. In Section , we introduce how to efficiently estimate $\Delta\Gamma$.

Node Influence on Model Bias Estimation

It is noteworthy that PDD is a function of $\hat{\boldsymbol{W}}$ for a trained GNN, as $\hat{\boldsymbol{W}}$ directly determines the probabilistic predictions for test nodes. Hence we first characterize how a training node in $\mathcal{G}$ influences $\hat{\boldsymbol{W}}$, followed by how this node influences PDD via applying the chain rule. Formally, the optimal parameters $\hat{\boldsymbol{W}}$ minimize the objective function $L_{\mathcal{V}'}(\mathcal{G}, \boldsymbol{W})$ of the node classification task, so that:

$$\hat{\boldsymbol{W}} \overset{\text{def}}{=} \arg\min_{\boldsymbol{W}} L_{\mathcal{V}'}(\mathcal{G}, \boldsymbol{W}) = \arg\min_{\boldsymbol{W}} \frac{1}{m}\sum_{i=1}^{m} L_{v_i}(\mathcal{G}_i, \boldsymbol{W}).$$

Here $L_{v_i}(\mathcal{G}_i, \boldsymbol{W})$ denotes the loss term associated with node v_i; $\mathcal{G}_i$ is the computation graph of v_i; m is the total number of training nodes. If a training node v_i is deleted from $\mathcal{G}$, the loss function will change and thus leads to a different $\hat{\boldsymbol{W}}$. We take v_i as an example to analyze the influence on $\hat{\boldsymbol{W}}$ after deleting a training node from $\mathcal{G}$. Traditionally, the existence of node v_i is considered as a binary state, which is either one (if v_i exists in $\mathcal{G}$) or zero (otherwise). But in our analysis, we treat it as a continuous variable to depict the intermediate states of the existence of v_i. Suppose that the existence of v_i is down-weighted in the training of a GNN on $\mathcal{G}$. This operation leads to two changes in the loss function: (1) the loss term associated with node v_i, i.e., $L_{v_i}(\mathcal{G}_i, \boldsymbol{W})$, is down-weighted; (2) the loss terms associated with other training nodes in the computation graph of v_i would also be influenced. The reason is that these nodes could be affected by the information from node v_i during the message passing in GNNs (Kipf and Welling 2017; Ying et al. 2019). Based on the above analysis, we define $\hat{\boldsymbol{W}}_{\epsilon,v_i}$ as the optimal parameter that minimizes the loss function when node v_i is down-weighted as follows:

$$\begin{aligned}\hat{\boldsymbol{W}}_{\epsilon,v_i} \overset{\text{def}}{=} &\arg\min_{\boldsymbol{W}} L_{\mathcal{V}'}(\mathcal{G}, \boldsymbol{W}) \\ &- \epsilon\left(L_{v_i}(\mathcal{G}_i, \boldsymbol{W}) + \tilde{L}_{\mathcal{V}_i'}(\mathcal{G}_i, \boldsymbol{W})\right),\end{aligned} \tag{4}$$

where $\epsilon \in [0, 1/m]$ controls the scale of down-weighting v_i. An illustration in Fig. 1 shows how down-weighting v_i affects the loss values of training nodes in its computation graph. To formally characterize how node v_i influences $\hat{\boldsymbol{W}}$, we have Theorem 1 as follows (see proofs in Appendix C).

Theorem 1. *According to the optimization objective of $\hat{\boldsymbol{W}}_{\epsilon,v_i}$ in Eq. (4), we have*

$$\begin{aligned}\left.\frac{d\hat{\boldsymbol{W}}_{\epsilon,v_i}}{d\epsilon}\right|_{\epsilon=0} = &\left(\frac{\partial^2 L_{\mathcal{V}'}(\mathcal{G}, \hat{\boldsymbol{W}})}{\partial \boldsymbol{W}^2}\right)^{-1} \\ &\cdot\left(\frac{\partial L_{v_i}\left(\mathcal{G}_i, \hat{\boldsymbol{W}}\right)}{\partial \boldsymbol{W}} + \frac{\partial \tilde{L}_{\mathcal{V}_i'}(\mathcal{G}_i, \hat{\boldsymbol{W}})}{\partial \boldsymbol{W}}\right).\end{aligned} \tag{5}$$

Then, we characterize the influence of down-weighting node v_i on the value of PDD. We present Corollary 1 based on the chain rule as follows (see the proofs in Appendix C).

Corollary 1. *Define the derivative of Γ w.r.t. ϵ at $\epsilon = 0$ as $I_\Gamma(v_i)$. According to Theorem 1, we have*

$$I_\Gamma(v_i) \overset{def}{=} \left.\frac{\partial \Gamma}{\partial \epsilon}\right|_{\epsilon=0} = \left(\frac{\partial \Gamma}{\partial \boldsymbol{W}}\right)^{\top} \left.\frac{d\hat{\boldsymbol{W}}_{\epsilon,v_i}}{d\epsilon}\right|_{\epsilon=0}. \tag{6}$$

With Corollary 1, we can now estimate the value change of Γ when node v_i is down-weighted via

$$\Gamma_{\epsilon,v_i} - \Gamma_{0,v_i} = -\epsilon \cdot I_\Gamma(v_i) + o(\epsilon) \approx -\epsilon \cdot I_\Gamma(v_i) \tag{7}$$

[1]Refer to the online version for Appendix.

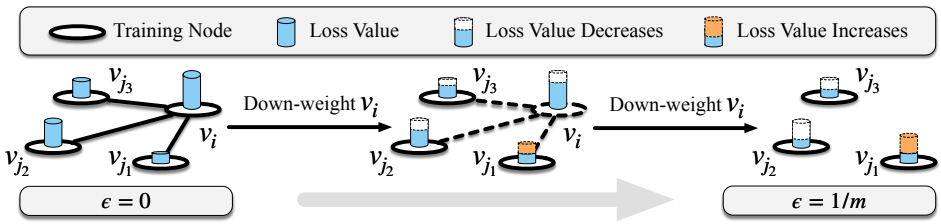


Figure 1: An illustration of how down-weighting node v_i influences the loss values of the training nodes in $\mathcal{G}_i$ (including v_i, v_{j_1}, v_{j_2}, and v_{j_3}). Scenarios from $\epsilon = 0$ to $\epsilon = 1/m$ are presented.

according to the first-order Taylor expansion. Here Γ_{ϵ,v_i} and Γ_{0,v_i} are the PDD values after and before node v_i is down-weighted, respectively. To estimate the value change in Γ for a GNN trained on $\mathcal{G}_{-i}$, we introduce Theorem 2 as follows (see the proofs in Appendix C).

Theorem 2. *Compared with the GNN trained on $\mathcal{G}$, $\Delta\Gamma = \Gamma_{\frac{1}{m},v_i} - \Gamma_{0,v_i}$ is equivalent to the value change in Γ when the GNN mode is trained on graph $\mathcal{G}_{-i}$.*

Theorem 2 enables us to directly compute the $\Delta\Gamma$ for an arbitrary training node v_i, which helps avoid the expensive re-training process. In the next section, we further define $\tilde{L}_{\mathcal{V}'_i}(\mathcal{G}_i, \hat{\boldsymbol{W}})$ and present an algorithm to efficiently estimate the node influence on model bias.

Non-I.I.D. Characterization

Generally, there are two types of dependencies between a training node v_i and other nodes in its computation graph $\mathcal{G}_i$, namely its dependency on other training nodes and its dependency on test nodes. The dependency between training nodes directly influences $\boldsymbol{W}$ during GNN training, and thus influences the probabilistic outcome of all test nodes. Hence it is critical to properly characterize the dependency between v_i and other training nodes. Specifically, we aim to characterize how the loss summation of all training nodes in $\mathcal{G}_i$ changes due to the existence of v_i. We denote the training nodes other than node v_i in $\mathcal{G}_i$ as $\mathcal{V}'_i \backslash \{v_i\}$. For any node $v_j \in \mathcal{V}'_i \backslash \{v_i\}$, we denote $\mathcal{G}_{j,-i}$ as the computation graph of node v_j with node v_i being deleted. $\tilde{L}_{\mathcal{V}_i}(\mathcal{G}_i, \hat{\boldsymbol{W}})$ is then formally given as

$$\tilde{L}_{\mathcal{V}_i}(\mathcal{G}_i, \hat{\boldsymbol{W}}) = \sum_{v_j \in \mathcal{V}'_i \backslash \{v_i\}} \left(L_{v_j}\left(\mathcal{G}_j, \hat{\boldsymbol{W}}\right) - L_{v_j}\left(\mathcal{G}_{j,-i}, \hat{\boldsymbol{W}}\right)\right). \quad (8)$$

The first term represents the summation of loss for nodes in $\mathcal{V}'_i \backslash \{v_i\}$ on $\mathcal{G}$, and the second term denotes the summation of loss for these nodes on $\mathcal{G}_{-i}$. In this regard, $\tilde{L}_{\mathcal{V}_i}(\mathcal{G}_i, \hat{\boldsymbol{W}})$ generally depicts to what extent the loss summation changes for nodes in $\mathcal{V}'_i \backslash \{v_i\}$ on graph $\mathcal{G}$ compared with $\mathcal{G}_{-i}$. If v_i is down-weighted by a certain degree, the change of the loss summation for nodes in $\mathcal{V}'_i \backslash \{v_i\}$ can be depicted by a linearly re-scaled $\tilde{L}_{\mathcal{V}_i}(\mathcal{G}_i, \hat{\boldsymbol{W}})$, as described in Eq. (4).

Additionally, there could also be dependencies between v_i and test nodes in $\mathcal{G}_i$, as v_i can influence the representations of its neighboring test nodes due to the information propagation mechanism in GNNs during inference. Such a dependency could also influence the value of PDD when v_i is deleted from $\mathcal{G}$. Correspondingly, we introduce the characterization of the dependency between v_i and test nodes. Specifically, we present an upper bound to depict the normalized change

Algorithm 1: Node Influence on Model Bias Estimation

Input: $\mathcal{G}$: the graph data; $f_{\hat{\boldsymbol{W}}}$: the trained GNN model; $\mathcal{V}'$: the set of training nodes;
Output: $\mathcal{I}_\Gamma = \{\Gamma_{\frac{1}{m},v_i} - \Gamma_{0,v_i} : v_i \in \mathcal{V}'\}$;

```
Initialize I_Γ = ∅;
Compute {∂Γ/∂W : v_i ∈ V'} based on f_Ŵ;
while v_i ∈ V' do
    Compute dŴ_{ε,v_i}/dε |_{ε=0} according to Eq. (5) and (8);
    Compute I_Γ(v_i) according to Eq. (6);
    Compute Γ_{1/m,v_i} − Γ_{0,v_i} according to Eq. (7);
    Append element Γ_{1/m,v_i} − Γ_{0,v_i} onto I_Γ;
end while
return I_Γ;
```

magnitude of the neighboring test nodes' representations when a training node v_i is deleted. Here the analysis is based on the prevalent GCN model (Kipf and Welling 2017), and can be easily generalized to other GNNs. Following widely adopted assumptions in (Huang and Zitnik 2020; Xu et al. 2018), we have Proposition 1 (see the proofs in Appendix C).

Proposition 1. *Denote the representations of node $v_j (v_j \in \mathcal{V} \backslash \mathcal{V}')$ based on $\mathcal{G}$ and $\mathcal{G}_{-i}$ as $\boldsymbol{z}_j$ and $\boldsymbol{z}_j^\star$, respectively. Define $h^{(j,i)}$ and $q^{(j,i)}$ as the distance from v_j to v_i and the number of all possible paths from v_j to v_i, respectively. Define the set of geometric mean node degrees of $q^{(j,i)}$ paths as $\mathcal{D} = \{d_1^{(j,i)}, ..., d_{q^{(j,i)}}^{(j,i)}\}$. Define $d_{min}^{(j,i)}$ as the minimum value of $\mathcal{D}$. Assume the norms of all node representations are the same. We then have $\|\boldsymbol{z}_j^\star - \boldsymbol{z}_j\|_2 / \|\boldsymbol{z}_j\|_2 \leq q^{(j,i)} / (d_{min}^{(j,i)})^{h^{(j,i)}}$.*

From Proposition 1, we observe that (1) deleting v_i exerts an upper-bounded impact on the representations of other test nodes in its computation graph; and (2) this upper-bound exponentially decays w.r.t. the distance between v_i and test nodes. Hence the dependency between v_i and test nodes has limited influence on Γ during inference when v_i is deleted from the graph. On the contrary, considering that the dependency between v_i and other training nodes directly influences $\hat{\boldsymbol{W}}$ and thus influences the inference results of all nodes, such a dependency should not be neglected. Consequently, we argue that it is reasonable to estimate the influence of each training node on Γ by only considering the dependency between training nodes. We present the algorithmic routine of $\Delta\Gamma$ estimation in Algorithm 1.

Complexity Analysis

To better understand the computational cost, here we analyze the time complexity of estimating $\Delta\Gamma$ according to Algorithm 1. We denote the number of parameters in $\boldsymbol{W}$ and the average number of training nodes in the computation graph of an arbitrary training node as t and $\bar{r}$, respectively. For each node v_i, the time complexity to compute $\partial L_{v_i}\left(\mathcal{G}_i, \hat{\boldsymbol{W}}\right)/\partial \boldsymbol{W}$ and $\partial \tilde{L}_{\mathcal{V}_i'}(\mathcal{G}_i, \hat{\boldsymbol{W}})/\partial \boldsymbol{W}$ is $O(t)$ and $O(\bar{r}t)$, respectively. Hence the time complexity is $O(m\bar{r}t)$ to traverse all training nodes. For the Hessian matrix inverse, we employ a widely-used estimation approach (see Appendix A) with linear time complexity w.r.t t. Thus the time complexity of Eq. (5) and (8) is $O(m\bar{r}t)$. Additionally, the time complexity of Eq. (6) and (7) is $O(mt)$ and $O(m)$, respectively. To summarize, the time complexity of Algorithm 1 is $O(m\bar{r}t)$. Considering that $\bar{r} \leq m$, the algorithm has a quadratic time complexity w.r.t. training node number. This verifies the impressive time efficiency of our algorithm.

Experiments

We aim to answer the following research questions in experiments. **RQ1**: How efficient is BIND in estimating the influence of training nodes on the mode bias? **RQ2**: How well can BIND estimate the influence of training nodes on the model bias? **RQ3**: How well can we debias GNNs via deleting harmful training nodes based on our estimation? More details of experimental settings, supplementary experiments, and further analysis are in Appendix B.

Experimental Setup

Downstream Task & Datasets. Here the downstream task is node classification. Four real-world datasets are adopted in our experiments, including *Income*, *Recidivism*, *Pokec-z*, and *Pokec-n*. Specifically, *Income* is collected from *Adult Data Set* (Dua and Graff 2017). Each individual is represented by a node, and we establish connections (i.e., edges) between individuals following a similar criterion adopted in (Agarwal, Lakkaraju, and Zitnik 2021). The sensitive attribute is race, and the task is to classify whether the salary of a person is over \$50K per year or not. *Recidivism* is collected from (Jordan and Freiburger 2015). A node represents a defendant released on bail, and defendants are connected based on their similarity. The sensitive attribute is race, and the task is to classify whether a defendant is on bail or not. *Pokec-z* and *Pokec-n* are collected from *Pokec*, which is a popular social network in Slovakia (Takac and Zabovsky 2012). In both datasets, each user is a node, and each edge stands for the friendship relation between two users. Here the locating region of users is the sensitive attribute. The task is to classify the working field of users. More details are in Appendix B.

Baselines & GNN Backbones. We compare our method with three state-of-the-art GNN debiasing baselines, namely FairGNN (Dai and Wang 2021a), NIFTY (Agarwal, Lakkaraju, and Zitnik 2021), and EDITS (Dong et al. 2022a). To perform GNN debiasing, FairGNN employs adversarial training to filter out the information of sensitive attributes from node embeddings; NIFTY maximizes the agreement between the predictions based on perturbed sensitive attributes and unperturbed ones; EDITS pre-processes the input graph data to be less biased via attribute and structural debiasing. We mainly present the results of using GCN (Kipf and Welling 2017) as the backbone GNN model, while experiments with other GNNs are discussed in Appendix B.

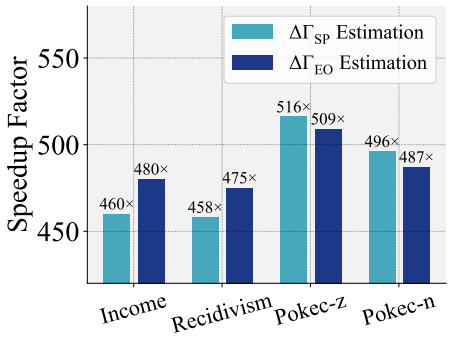


Figure 2: Evaluation of efficiency: speedup factors of $\Delta\Gamma_{\text{SP}}$ and $\Delta\Gamma_{\text{EO}}$ estimation over GNN re-training.

Evaluation Metrics. First, we employ running speedup factors to evaluate efficiency. Second, we use the widely adopted Pearson Correlation (Koh and Liang 2017; Chen et al. 2020) between the estimated and actual $\Delta\Gamma$ to evaluate the effectiveness of node influence estimation. Third, we adopt two traditional fairness metrics, namely Δ_{SP} (the metric for *Statistical Parity*) (Dwork et al. 2012) and Δ_{EO} (the metric for *Equal Opportunity*) (Hardt, Price, and Srebro 2016b), to evaluate the effectiveness of debiasing GNNs via harmful nodes deletion. Additionally, the classification accuracy is also employed to evaluate the utility-fairness trade-off.

Efficiency of Node Influence Estimation

To answer RQ1, we evaluate the efficiency of $\Delta\Gamma$ estimation by comparing its running time with that of GNN re-training. The running time of GNN re-training is computed as follows. We first delete the target node from the original input graph $\mathcal{G}$ and re-train the GCN to obtain $f_{\hat{\boldsymbol{W}}'}$. We then obtain $\Delta\Gamma$ based on the values of Γ given by $f_{\hat{\boldsymbol{W}}}$ and $f_{\hat{\boldsymbol{W}}'}$. The above running time is defined as the time cost of GNN re-training. The running time averaged across all training nodes is compared between GNN re-training and BIND, and we present the running speedup factors of BIND on the four real-world datasets in Fig. 2. We observe that the running speedup factors are over 450× on all four real-world datasets, which corroborates the efficiency superiority of BIND in estimating the value of $\Delta\Gamma$. Additionally, we observe that the estimation on Pokec-z and Pokec-n datasets has higher speedup factors on both $\Delta\Gamma_{\text{SP}}$ and $\Delta\Gamma_{\text{EO}}$ compared with the other two datasets. A reason could be that nodes in Pokec-z and Pokec-n have lower average degrees (see Appendix B). This facilitates the computation of $\tilde{L}_{\mathcal{V}_i'}(\mathcal{G}_i, \hat{\boldsymbol{W}})$ (the term that characterizes non-i.i.d.) and corresponding derivatives.

Effectiveness of Node Influence Estimation

We now evaluate the effectiveness of $\Delta\Gamma$ estimation. It is worth noting that the numerical values of the estimated influence on model bias are small for most of the nodes (see Appendix B). Here we introduce a strategy to evaluate the

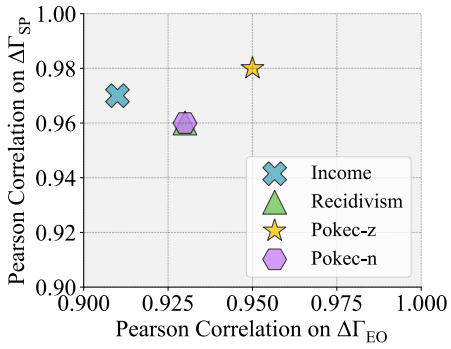


Figure 3: Evaluation of effectiveness: correlation between estimated and actual $\Delta\Gamma_{SP}$ and $\Delta\Gamma_{EO}$.

estimation effectiveness across a wider value range of $\Delta\Gamma$. The basic intuition here is that we select node sets and evaluate how well their estimated $\Delta\Gamma$ summation aligns with the actual one. Specifically, we first follow the widely adopted routine (Koh and Liang 2017; Chen et al. 2020) to truncate the helpful and harmful nodes with top-ranked $\Delta\Gamma$ values. We then construct a series of node sets associated with the largest positive and negative estimated $\Delta\Gamma$ summations under different set size thresholds. The range of these thresholds is between zero and a maximum possible value (determined by the training set size). It is worth noting that only nodes with non-overlapping computation graphs are selected in constructing each node set. This ensures that these nodes result in an estimated $\Delta\Gamma$ equivalent to the summation of their estimated $\Delta\Gamma$ (see Appendix C). We present the Pearson correlation of estimated $\Delta\Gamma_{SP}$ and $\Delta\Gamma_{EO}$ with the actual values on four datasets in Fig. 3. It is worth noting that achieving an exact linear correlation (i.e., Pearson correlation equals one) between the estimated and actual $\Delta\Gamma$ is almost impossible, since we only employ the first-order Taylor expansion in our estimation for $\Delta\Gamma$. From Fig. 3, we observe that the estimation achieves Pearson correlation values over 0.9 on both Γ_{SP} and Γ_{EO} across all datasets. Such consistencies between estimated and actual values verify the effectiveness of BIND.

Additionally, to understand how the non-i.i.d. characterization benefits the estimation, we also estimate $\Delta\Gamma$ with BIND after the non-i.i.d. characterization being disabled (i.e., setting the $\tilde{L}_{\mathcal{V}_i'}(\mathcal{G}_i, \boldsymbol{W})$ term in Eq. 4 as 0). We present the estimated $\Delta\Gamma$ v.s. actual $\Delta\Gamma$ on Income dataset with non-i.i.d. characterization being enabled and disabled in Fig. 4a and 4b, respectively. We observe the correlation decreases between the estimated and actual $\Delta\Gamma$ after the non-i.i.d. characterization is disabled. Such a decrease is also observed on other datasets in terms of both statistical parity and equal opportunity. Such an observation verifies the contribution of non-i.i.d. characterization to the estimation of $\Delta\Gamma$.

Finally, we evaluate how well the values of the proposed PDD matches the values of traditional fairness metrics. We collect the value pairs of (Δ_{SP}, Γ_{SP}) and (Δ_{EO}, Γ_{EO}) during the GNN re-training process. The values of Δ_{SP} v.s. actual Γ_{SP} are presented in Fig. 4c, and the values of Δ_{EO} v.s. actual Γ_{EO} are shown in Fig. 4d. We observe a satisfying match between Γ and traditional metrics, which corroborates that PDD is a valid indicator of the fairness level depicted by traditional fairness metrics.

Debiasing via Harmful Nodes Deletion

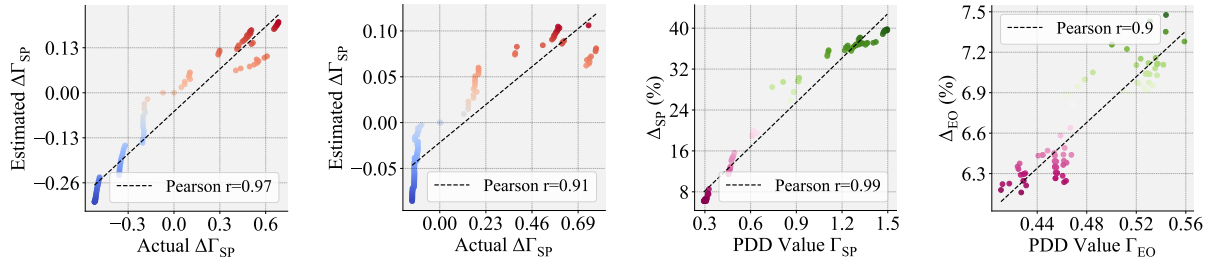
In this subsection, we demonstrate how BIND could be employed for GNN debiasing. The basic intuition here is to identify and delete those harmful nodes according to the estimated node influence on model bias, and evaluate whether GNNs can be debiased when they are trained on this new graph. Specifically, we set $\Gamma = \lambda\Gamma_{SP} + (1-\lambda)\Gamma_{EO}$ and estimate the node influence on Γ to consider both statistical parity and equal opportunity. We then set a budget k, and follow the strategy adopted in Section to select and delete a set of training nodes with the largest positive influence summation on Γ under this budget. We set $\lambda = 0.5$ to assign statistical parity and equal opportunity the same weight, and perform experiments with k being 1% (denoted as BIND 1%) and 10% (denoted as BIND 10%) of the total number of training nodes. We present the results on the four adopted datasets in Table 1. The following observations are made: (1) compared with other baselines, BIND achieves competitive performance (i.e., lower values) on both Δ_{SP} and Δ_{EO}. Hence training GNNs on a new graph after deleting harmful nodes (to fairness) is an effective approach for GNN debiasing; (2) there is no obvious performance decrease on the model utility of BIND compared with other baselines. We thus argue that deleting harmful nodes can also lead to a satisfying fairness-utility trade-off.

Related Work

Graph Neural Networks. GNNs can be divided into spectral-based and spatial-based ones (Wu et al. 2020; Zhou et al. 2020). Spectral GNNs inherit the insights from Convolutional Neural Networks (CNNs) (Bruna et al. 2013), and followed by many works (Defferrard, Bresson, and Vandergheynst 2016; Levie et al. 2018; Kipf and Welling 2017). Their goal is to design graph filters to extract task-related information from the input graphs (Chung and Graham 1997). Differently, spatial GNNs design message-passing mechanisms in the spatial domain to extract information from each node's neighbors (Wu et al. 2020; Zhou et al. 2020). Various aggregation strategies contribute to different tasks (Veličković et al. 2017; Xu et al. 2019b; Suresh et al. 2021; Park and Neville 2020).
Algorithmic Fairness. Algorithmic fairness can be defined from different perspectives (Pessach and Shmueli 2020; M. et al. 2021; Du et al. 2020; Caton and Haas 2020; Corbett-Davies and Goel 2019; Mitchell et al. 2021), where *Group Fairness* and *Individual Fairness* are two popular notions (Dwork et al. 2012). Generally, group fairness enforces similar statistics (e.g., positive prediction rate in binary classification tasks) across different demographic subgroups (Dwork et al. 2012). Typically, these demographic subgroups are described by certain sensitive attributes, such as gender, race, and religion. Individual fairness argues for similar outputs for similar individuals (Dwork et al. 2012). Algorithmic fairness can be considered in different stages of learning pipelines, including pre-processing (Dong et al. 2022a), in-processing (Dong et al. 2021; Lahoti, Gummadi, and Weikum 2019; Dai and Wang 2021b), and post-processing (Kang et al. 2020). Particularly, re-weighting train-

		Van. GCN	FairGNN	NIFTY	EDITS	BIND 1%	BIND 10%
Income	(↑) **Acc**	74.7 ± 1.4	69.1 ± 0.6	70.8 ± 0.9	68.3 ± 0.8	**75.2 ± 0.0**	71.7 ± 0.7
	(↓) Δ_{SP}	25.9 ± 1.9	**12.4 ± 4.7**	24.4 ± 1.6	24.0 ± 1.9	19.2 ± 0.6	14.7 ± 1.4
	(↓) Δ_{EO}	32.3 ± 0.8	**15.6 ± 6.8**	26.9 ± 3.7	24.9 ± 1.0	26.4 ± 0.4	16.2 ± 2.0
Recidivism	(↑) **Acc**	**89.8 ± 0.0**	89.7 ± 0.2	79.1 ± 0.9	89.6 ± 0.1	88.7 ± 0.0	88.5 ± 0.2
	(↓) Δ_{SP}	7.47 ± 0.2	7.31 ± 0.5	**1.82 ± 0.8**	5.02 ± 0.0	7.40 ± 0.0	6.57 ± 0.2
	(↓) Δ_{EO}	5.23 ± 0.1	5.17 ± 0.0	**1.28 ± 0.5**	2.89 ± 0.1	5.09 ± 0.1	4.23 ± 0.2
Pokec-z	(↑) **Acc**	63.2 ± 0.7	64.0 ± 0.7	**65.3 ± 0.2**	61.6 ± 0.9	63.5 ± 0.4	62.9 ± 0.4
	(↓) Δ_{SP}	7.32 ± 2.2	4.95 ± 0.8	2.34 ± 1.0	1.29 ± 0.8	6.75 ± 2.3	**1.02 ± 0.9**
	(↓) Δ_{EO}	7.60 ± 2.3	4.29 ± 0.7	**1.46 ± 1.3**	1.62 ± 1.6	5.41 ± 3.4	2.28 ± 1.5
Pokec-n	(↑) **Acc**	58.5 ± 0.8	60.3 ± 0.5	**61.1 ± 0.3**	56.8 ± 0.9	60.6 ± 0.8	58.8 ± 1.8
	(↓) Δ_{SP}	6.57 ± 2.6	5.30 ± 1.4	6.55 ± 0.7	2.75 ± 1.8	5.85 ± 2.0	**2.45 ± 0.9**
	(↓) Δ_{EO}	2.33 ± 0.5	1.67 ± 0.2	1.83 ± 0.6	2.24 ± 1.5	**1.15 ± 0.7**	2.22 ± 1.6

Table 1: Comparison on GNN utility and bias mitigation between BIND and baselines. BIND 1% and BIND 10% denote the node deletion budget k being 1% and 10% of the training node set size, respectively. (↑) denotes the larger, the better; (↓) denotes the opposite. Numerical results are in percentages. Best ones and runner-ups are in bold and underline, respectively.



(a) With non-i.i.d. term (b) Without non-i.i.d. term (c) Δ_{SP} v.s. Γ_{SP} (Income) (d) Δ_{EO} v.s. Γ_{EO} (Recid.)

Figure 4: In (a) and (b), we compare the estimation effectiveness of $\Delta\Gamma_{SP}$ with and without characterizing non-i.i.d.; in (c) and (d), we present the consistency between Γ and traditional fairness metrics (Δ_{SP} for statistical parity and Δ_{EO} for equal opportunity) under different node deletion budgets.

ing samples to mitigate model bias is a popular fairness-enhancing method during in-processing stage (Wang, Wu, and He 2022; Han, Baldwin, and Cohn 2021; Yan, Seto, and Apostoloff 2022; Jiang and Nachum 2020; Petrović et al. 2022). However, most of these methods only yield a set of weights for training samples to mitigate bias (Yan, Seto, and Apostoloff 2022; Wang, Wu, and He 2022), while to what extent each sample influences the exhibited bias is still unclear. Different from them, this work aims to understand the influence of each training node on model bias. To the best of our knowledge, this is a first-of-its-kind study. Moreover, most of existing methods based on re-weighting training samples are developed under the IID assumption. However, in this paper, we also analyze the non-IID characteristic between nodes to understand how each training node influences model bias.

Interpretation of Deep Learning Models. Deep learning models have huge parameter size and high complexity (Buhrmester, Münch, and Arens 2021; Samek, Wiegand, and Müller 2017; Fong and Vedaldi 2017; Xu et al. 2019a). To make these models more trustworthy and controllable, many studies have been devoted to improving their transparency (Fong and Vedaldi 2017). Generally, these works are divided into transparency design and post-hoc explanation (Xu et al. 2019a). The basic goal of transparency design is to understand the model in terms of model structure (Liu et al. 2021; Zhang et al. 2019) and training algorithms (Plumb et al. 2019), while post-hoc explanation aims to explain specific prediction results via visualization (Ding et al. 2017) and explanatory examples (Chen et al. 2018). In the realm of learning on graphs, some existing works aim to interpret GNNs (Ying et al. 2019; Luo et al. 2020; Yuan et al. 2020a), and they mainly focus on understanding the utility (e.g., node classification accuracy) of GNNs on the test set. Our work is different from them in two aspects: (1) we focus on interpreting the model bias instead of the utility for GNNs; (2) we aim to understand the model bias via attributing to the training set instead of only focusing on the test set.

Conclusion

In this paper, we study a novel problem of characterizing how each training node influences the bias exhibited in a trained GNN. We first propose a strategy named Probabilistic Distribution Disparity (PDD), which can be instantiated with different existing fairness notions, to quantify the node influence on the model bias. We then propose a novel framework named BIND to achieve an efficient influence estimation for each training node. We also develop a node deletion strategy to achieve GNN debiasing based on influence estimation. Extensive experiments verify (1) the consistency between the proposed PDD and traditional fairness metrics; (2) the efficiency and effectiveness of the influence estimation algorithm; and (3) the performance of the proposed strategy on GNN debiasing. We leave interpreting how the unfairness arises in other graph learning tasks as future works.

Acknowledgments

This work is supported by the National Science Foundation under grants IIS-2006844, IIS-2144209, IIS-2223768, IIS-2223769, CNS-2154962, and BCS-2228534, the JP Morgan Chase Faculty Research Award, and the Cisco Faculty Research Award.

References

Agarwal, C.; Lakkaraju, H.; and Zitnik, M. 2021. Towards a Unified Framework for Fair and Stable Graph Representation Learning. In *UAI*.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein GAN. In *ICML*.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Buhrmester, V.; Münch, D.; and Arens, M. 2021. Analysis of explainers of black box deep neural networks for computer vision: A survey. *MLKE*.

Caton, S.; and Haas, C. 2020. Fairness in machine learning: A survey. *CSUR*.

Chen, H.; Si, S.; Li, Y.; Chelba, C.; Kumar, S.; Boning, D.; and Hsieh, C.-J. 2020. Multi-stage influence function. *NeurIPS*.

Chen, J.; Song, L.; Wainwright, M.; and Jordan, M. 2018. Learning to explain: An information-theoretic perspective on model interpretation. In *ICML*.

Cheng, D.; Wang, X.; Zhang, Y.; and Zhang, L. 2020. Graph neural network for fraud detection via spatial-temporal attention. *TKDE*.

Chung, F. R.; and Graham, F. C. 1997. *Spectral graph theory*. 92. American Mathematical Soc.

Corbett-Davies, S.; and Goel, S. 2019. The measure and mismeasure of fairness: A critical review of fair machine learning. In *NeurIPS*.

Dai, E.; and Wang, S. 2021a. Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information. In *WSDM*.

Dai, E.; and Wang, S. 2021b. Towards Self-Explainable Graph Neural Network. In *CIKM*.

Dai, E.; and Wang, S. 2022. Learning fair graph neural networks with limited and private sensitive attribute information. *TKDE*.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *NeurIPS*.

Ding, Y.; Liu, Y.; Luan, H.; and Sun, M. 2017. Visualizing and understanding neural machine translation. In *ACL*.

Do, K.; Tran, T.; and Venkatesh, S. 2019. Graph transformation policy network for chemical reaction prediction. In *SIGKDD*.

Dong, Y.; Kang, J.; Tong, H.; and Li, J. 2021. Individual Fairness for Graph Neural Networks: A Ranking based Approach. In *SIGKDD*.

Dong, Y.; Liu, N.; Jalaian, B.; and Li, J. 2022a. EDITS: Modeling and Mitigating Data Bias for Graph Neural Networks. In *The Web Conf*.

Dong, Y.; Ma, J.; Chen, C.; and Li, J. 2022b. Fairness in Graph Mining: A Survey. *arXiv preprint arXiv:2204.09888*.

Dong, Y.; Wang, S.; Wang, Y.; Derr, T.; and Li, J. 2022c. On Structural Explanation of Bias in Graph Neural Networks. In *KDD*.

Du, M.; Yang, F.; Zou, N.; and Hu, X. 2020. Fairness in deep learning: A computational perspective. *IEEE Intelligent Systems*.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. Accessed: 2023-02-27.

Dwork, C.; Hardt, M.; Pitassi, T.; Reingold, O.; and Zemel, R. 2012. Fairness through awareness. In *ITCS*.

Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; and Yin, D. 2019. Graph neural networks for social recommendation. In *The Web Conf*.

Fong, R. C.; and Vedaldi, A. 2017. Interpretable explanations of black boxes by meaningful perturbation. In *ICCV*.

Guo, Z.; and Wang, H. 2020. A deep graph neural network-based mechanism for social recommendations. *TKDE*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NeurIPS*.

Han, X.; Baldwin, T.; and Cohn, T. 2021. Balancing out Bias: Achieving Fairness Through Balanced Training. *arXiv preprint arXiv:2109.08253*.

Hardt, M.; Price, E.; and Srebro, N. 2016a. Equality of Opportunity in Supervised Learning. In *NeurIPS*.

Hardt, M.; Price, E.; and Srebro, N. 2016b. Equality of opportunity in supervised learning. In *NeurIPS*.

Huang, K.; and Zitnik, M. 2020. Graph meta learning via local subgraphs. In *NeurIPS*.

Jiang, H.; and Nachum, O. 2020. Identifying and correcting label bias in machine learning. In *AISTATS*.

Jin, G.; Wang, Q.; Zhu, C.; Feng, Y.; Huang, J.; and Zhou, J. 2020. Addressing crime situation forecasting task with temporal graph convolutional neural network approach. In *ICMTMA*.

Jordan, K. L.; and Freiburger, T. L. 2015. The effect of race/ethnicity on sentencing: Examining sentence type, jail length, and prison length. *J Crim Justice*.

Kang, J.; He, J.; Maciejewski, R.; and Tong, H. 2020. Inform: Individual fairness on graph mining. In *SIGKDD*.

Kantorovich, L. V. 1960. Mathematical methods of organizing and planning production. *Management science*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Koh, P. W.; and Liang, P. 2017. Understanding black-box predictions via influence functions. In *ICML*.

Kwon, Y.; Lee, D.; Choi, Y.-S.; and Kang, S. 2022. Uncertainty-aware prediction of chemical reaction yields with graph neural networks. *Journal of Cheminformatics*.

Lahoti, P.; Gummadi, K. P.; and Weikum, G. 2019. iFair: Learning Individually Fair Data Representations for Algorithmic Decision Making. In *ICDE*.

Levie, R.; Monti, F.; Bresson, X.; and Bronstein, M. M. 2018. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Trans. Signal Process.*

Li, P.; Wang, Y.; Zhao, H.; Hong, P.; and Liu, H. 2021. On dyadic fairness: Exploring and mitigating bias in graph connections. In *ICLR*.

Liu, G.; Sun, X.; Schulte, O.; and Poupart, P. 2021. Learning Tree Interpretation from Object Representation for Deep Reinforcement Learning. *NeurIPS*.

Liu, N.; Feng, Q.; and Hu, X. 2022. Interpretability in Graph Neural Networks. *Graph Neural Networks: Foundations, Frontiers, and Applications*.

Loveland, D.; Pan, J.; Bhathena, A. F.; and Lu, Y. 2022. FairEdit: Preserving Fairness in Graph Neural Networks through Greedy Graph Editing. *arXiv preprint arXiv:2201.03681*.

Luo, D.; Cheng, W.; Xu, D.; Yu, W.; Zong, B.; Chen, H.; and Zhang, X. 2020. Parameterized explainer for graph neural network. In *NeurIPS*.

M., N.; M., F.; S., N.; L., K.; and G., A. 2021. A survey on bias and fairness in machine learning. *CSUR*.

Ma, J.; Deng, J.; and Mei, Q. 2021. Subgroup generalization and fairness of graph neural networks. In *NeurIPS*.

Mitchell, S.; Potash, E.; Barocas, S.; D'Amour, A.; and Lum, K. 2021. Algorithmic fairness: Choices, assumptions, and definitions. *Annu. Rev. Stat. Appl.*

Park, H.; and Neville, J. 2020. Role Equivalence Attention for Label Propagation in Graph Neural Networks. In *PAKDD*.

Pessach, D.; and Shmueli, E. 2020. Algorithmic fairness. *arXiv preprint arXiv:2001.09784*.

Petrović, A.; Nikolić, M.; Radovanović, S.; Delibašić, B.; and Jovanović, M. 2022. FAIR: Fair adversarial instance re-weighting. *Neurocomputing*.

Plumb, G.; Al-Shedivat, M.; Cabrera, A. A.; Perer, A.; Xing, E.; and Talwalkar, A. 2019. Regularizing black-box models for improved interpretability. *arXiv preprint arXiv:1902.06787*.

Pourhabibi, T.; Ong, K.-L.; Kam, B. H.; and Boo, Y. L. 2020. Fraud detection: A systematic literature review of graph-based anomaly detection approaches. *Decision Support Systems*.

Samek, W.; Wiegand, T.; and Müller, K.-R. 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*.

Shi, C.; Xu, M.; Guo, H.; Zhang, M.; and Tang, J. 2020. A graph to graphs framework for retrosynthesis prediction. In *ICML*.

Shumovskaia, V.; Fedyanin, K.; Sukharev, I.; Berestnev, D.; and Panov, M. 2020. Linking bank clients using graph neural networks powered by rich transactional data. In *DSAA*.

Song, W.; Dong, Y.; Liu, N.; and Li, J. 2022. GUIDE: Group Equality Informed Individual Fairness in Graph Neural Networks. In *KDD*.

Song, W.; Xiao, Z.; Wang, Y.; Charlin, L.; Zhang, M.; and Tang, J. 2019. Session-based social recommendation via dynamic graph attention networks. In *WSDM*.

Sun, Y.; Wang, S.; Tang, X.; Hsieh, T.-Y.; and Honavar, V. 2020. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *The Web Conf*.

Suresh, S.; Budde, V.; Neville, J.; Li, P.; and Ma, J. 2021. Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns. *arXiv preprint arXiv:2106.06586*.

Takac, L.; and Zabovsky, M. 2012. Data analysis in public social networks. In *Internation. scient. workshop*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, D.; Lin, J.; Cui, P.; Jia, Q.; Wang, Z.; Fang, Y.; Yu, Q.; Zhou, J.; Yang, S.; and Qi, Y. 2019. A semi-supervised graph attentive network for financial fraud detection. In *ICDM*. IEEE.

Wang, H.; Wu, Z.; and He, J. 2022. Training Fair Deep Neural Networks by Balancing Influence. *arXiv preprint arXiv:2201.05759*.

Wang, Y.; Zhao, Y.; Dong, Y.; Chen, H.; Li, J.; and Derr, T. 2022. Improving Fairness in Graph Neural Networks via Mitigating Sensitive Attribute Leakage. In *KDD*.

Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. 2019. Simplifying graph convolutional networks. In *ICML*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE Trans Neural Netw Learn Syst*.

Xu, F.; Uszkoreit, H.; Du, Y.; Fan, W.; Zhao, D.; and Zhu, J. 2019a. Explainable AI: A brief survey on history, research areas, approaches and challenges. In *NLPCC*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019b. How Powerful are Graph Neural Networks? In *ICLR*.

Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.-i.; and Jegelka, S. 2018. Representation learning on graphs with jumping knowledge networks. In *ICML*.

Yan, B.; Seto, S.; and Apostoloff, N. 2022. FORML: Learning to Reweight Data for Fairness. *arXiv preprint arXiv:2202.01719*.

Ying, R.; Bourgeois, D.; You, J.; Zitnik, M.; and Leskovec, J. 2019. Gnnexplainer: Generating explanations for graph neural networks. In *NeurIPS*.

Yuan, H.; Tang, J.; Hu, X.; and Ji, S. 2020a. Xgnn: Towards model-level explanations of graph neural networks. In *SIGKDD*.

Yuan, H.; Yu, H.; Gui, S.; and Ji, S. 2020b. Explainability in graph neural networks: A taxonomic survey. *arXiv preprint arXiv:2012.15445*.

Zhang, Q.; Yang, Y.; Ma, H.; and Wu, Y. N. 2019. Interpreting cnns via decision trees. In *CVPR*.

Zhang, S.; Zhu, F.; Yan, J.; Zhao, R.; and Yang, X. 2022a. DOTIN: Dropping Task-Irrelevant Nodes for GNNs. *arXiv preprint arXiv:2204.13429*.

Zhang, W.; Weiss, J. C.; Zhou, S.; and Walsh, T. 2022b. Fairness Amidst Non-IID Graph Data: A Literature Review. *arXiv preprint arXiv:2202.07170*.

Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2020. Graph neural networks: A review of methods and applications. *AI Open*.