# GNN Fairness through Graph Preprocessing

**Abhijit Gupta**
Yale University
abhijit.gupta@yale.edu

**Meili Gupta**
Yale University
meili.gupta@yale.edu

## 1 Introduction

While graph neural networks (GNNs) have emerged as a powerful tool for analyzing graph structured data, they have the tendency to perpetuate and even amplify social biases, leading to unfair and discriminatory outcomes. GNN fairness is concerned with eliminating harmful biases and ensuring models output predictions independent of protected features. While existing methods counteract attribute and structural bias in many ways, these methods are inefficient and costly, reducing real-world applicability. Unlike previous approaches, we prioritize efficiency and focus specifically on graph preprocessing to eliminate bias.

We separate the problem into node and edge debiasing. Within node debiasing, we propose two methods to learn fair node representations. First, we use linear algebra to remove the correlation between the features and the sensitive attribute. Second, we use adversarial debiasing to simultaneously train an encoder and classifier. The encoder learns to create similar embeddings that prevent the classifier from predicting the sensitive attribute. Within edge debiasing, we focus on heuristics to reduce excess homophily by removing a subset of edges between nodes of the same protected attribute. Our first method attempts to remove edges that are not likely to exist in the graph and our second method attempts to remove edges whose existence in the graph is heavily influenced by the sensitive attribute. Finally, we propose a combined method that uses both the adversarial debiasing node method and the second edge method.

As baselines, we consider a vanilla model both aware and unaware of the sensitive attribute. We also consider existing state-of-the-art methods FairGNN and EDITS. We experiment with five datasets, ultimately eliminating two due to unique graph properties. On the German Credit, Pokec-n, and Pokec-z datasets, we evaluate our methods for runtime, performance, and fairness. We find that both the node and edge methods improve fairness significantly with little to no impact on performance. While runtime increases, it is still significantly less than FairGNN and EDITS. Although the existing baselines have greater performance on these datasets, we believe our methods fill a valuable niche as starting points for debiasing large graph datasets. We plan to continue testing our method on larger datasets not previously considered for GNN fairness and explore additional modifications to our node and edge debiasing methods.

## 2 Related Works

### 2.1 Background

Suppose you have trained a GNN on a social network dataset, made several design decisions to optimize your performance metrics, and reached a stable version. Upon learning that one of the variables, say gender, is protected and cannot be used, you might just remove it from the input features and retrain the model. However, fairness through unawareness is not nearly strong enough, and the model may still be using the other features to estimate gender and act in biased and potentially harmful ways. As a simple example, the model may use height as a proxy for gender and systematically underestimate or overestimate the output based on that. We evaluate our fairness methods against the aware and unaware baselines, and hope to significantly outperform them.

Preprint. Under review.

## 2.2 State of the Art Models

When evaluating the performance of our fairness methods, we initially compare against the vanilla node classification model with no modifications. We also compare against two state-of-the-art fairness methods in literature.

1. **FairGNN** [1] The authors of FairGNN propose an adversarial GNN network to learn more fair node representations. They assumed that in many real-world scenarios, it will be difficult to obtain high amounts of sensitive attributes, and restricted their graphs to very small training splits. Their framework is composed of three models: the GNN classifier $f_G$, the GCN based sensitive attribute estimator $f_E$, and its adversary $f_A$. The sensitive attribute estimator circumvents the aforementioned problem by estimating the sensitive labels based on the biased input features. The remaining model architecture closely resembles a traditional adversarial model where the classifier and adversary have competing loss functions that encourage the node embeddings to be fair with respect to the sensitive attribute. The authors also conducted experiments that demonstrated that edges, in addition to nodes, introduce bias. Although their results are strong, the model takes several minutes to run on small datasets and must be run every time a new model is trained.

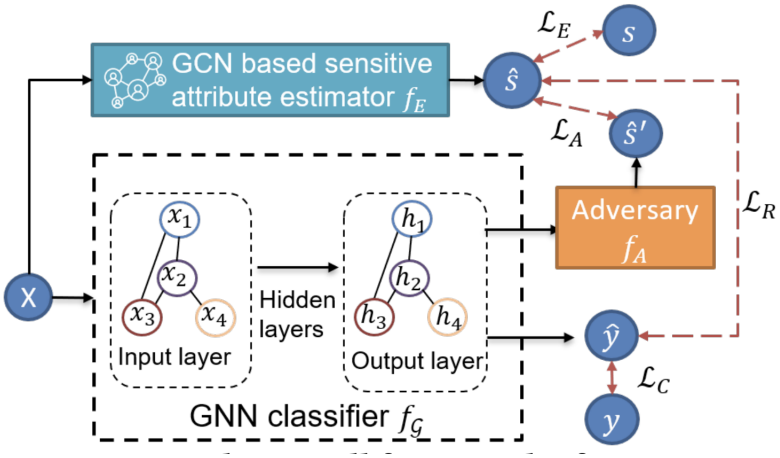


Figure 1: FairGNN Model Architecture

2. **EDITS** [2] EDITS takes a different approach to FairGNN by debiasing the graph instead of the GNN. This paper introduces two new measures of unfairness: attribute and structural bias. Attribute bias is the Wasserstein distance between the node feature distributions split by sensitive attribute, while structural bias is the same computation after two rounds of neighborhood aggregation. The model uses a continuous relaxation and directly optimizes these two metrics iteratively. The resulting graph is ideally fair regardless of the GNN run on it. While we see the benefits of graph preprocessing and adopt it in our methodology, we find EDITS to be computationally expensive and seek a more efficient method.

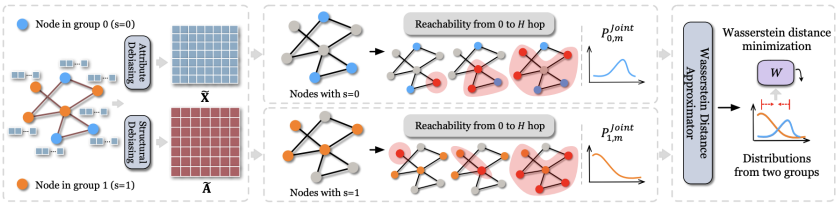


Figure 2: EDITS Model Architecture

# 3 Methods

## 3.1 Definitions

Let $G(V, E)$ be a homogeneous graph with $N = |V|$ nodes and $|E|$ edges. Each node has $F$ binary or continuous features, cumulatively represented by $X \in \mathbb{R}^{N \times F}$. The symmetric adjacency matrix $A \in \{0, 1\}^{N \times N}$ represents undirected edges in the graph. We denote the binary labels assigned to each node as $Y$ and the binary protected features of each node as $S$. A $GNN$ model $f$ accepts inputs $X$ and $A$ and outputs predictions $\hat{Y}$ close to $Y$. GNN Fairness is concerned with ensuring that $\hat{Y}$ is not systematically biased with respect to $S$.

## 3.2 Base Model

The base model architecture is a two-layer graph neural network classifier. For each dataset, we test the Graph Convolutional Network (GCN) and GraphSAGE blocks and select the one with best performance. The GCN block and GraphSAGE blocks are defined as follows

$$h_v^{(l)} = \sigma\Big( \sum_{u \in N(v)} W^{(l)} \frac{h_u^{(l-1)}}{\sqrt{|N(u)||N(v)|}} \Big) \tag{1}$$

$$h_v^{(l)} = \sigma(W^{(l)} \cdot \text{CONCAT}(h_v^{(l-1)}, \text{AGGREGATE}_k(\{h_u^{(l-1)} \forall u \in \mathcal{N}(v)\}))) \tag{2}$$

where $h_v^{(l)}$ is the embedding for node $v$ at layer $l$ of the model, $N(v)$ is the neighbor set of node $v$, and $W_l$ is the trainable weight matrix at layer $l$. $h_v^0$ is initialized as the features of node $v$. AGGREGATE can be any symmetric and trainable pooling operation, such as max or mean.

The model is composed of two GNN layers, with a ReLU nonlinearity in between and a sigmoid activation function at the end. The number of in-channels is the dimension of node features $F$, and there is 1 out-channel for the binary label classification objective. We use 16 nodes in the hidden channel. We use the Adam optimizer and gradient descent to learn an estimator of the node label. The loss function is the average across nodes of the binary cross entropy.

$$min_{\theta_G} \mathcal{L}_c = -\frac{1}{|V|} \sum_{v \in V} [y_v \log \hat{y}_v + (1 - y_v) \log(1 - \hat{y}_v)] \tag{3}$$

We train for 1000 epochs and use early stopping on the validation fold to avoid overfitting to the training data. We train both an "aware" variant where $S$ is included within $X$, and an "unaware" variant where $S$ is not included within $X$.

## 3.3 Goals

While FairGNN and EDITS report strong results on several datasets, their methods are computationally expensive, limiting their scalability and real-world applications. We seek to develop fairness methods that are not significantly more expensive than the baseline training process. While the EDITS method is slow, they do improve upon FairGNN by debiasing the graph instead of the GNN, reducing how often the method must be run. We therefore also primarily explore graph preprocessing fairness methods.

Our preprocessing methods can be separated into two categories: **node debiasing** and **edge debiasing**. Node debiasing attempts to modify the node features $X$ to obtain modified features $\tilde{X}$ that result in fairer GNNs. Similarly, edge debiasing attempts to modify the adjacency matrix $A$ to obtain $\tilde{A}$. We propose two approaches for each category. We also explore how the two methods can be combined for greater effect.

## 3.4 Node Debiasing

To improve computational efficiency, we discard all edges and treat node debiasing as a tabular data optimization problem. Given a set of rows each with $x \in \mathbb{R}^F$ and sensitive attribute $s \in \{0, 1\}$, we seek to produce debiased features $\tilde{x} \in \mathbb{R}^F$. We attempt two approaches: **Correlation Remover** and **Adversarial Debiasing**.

### 3.4.1 Correlation Remover

The correlation remover simply attempts to remove linear correlations between the node features and the sensitive attribute. Although this approach cannot target nonlinear effects and other statistical dependencies, we hypothesize that linear correlations can account for a significant portion of sensitive data leakage. Additionally, compared to minimizing mutual information or other more comprehensive metrics, we can remove the linear correlation without optimization, improving efficiency.

Mathematically, this method seeks to optimize the following expression

$$\min_{z_1,\ldots,z_n} \sum_{i=1}^{n} |z_i - x_i|^2 \text{ subject to } \frac{1}{n}\sum_{i=1}^{n} z_i(s_i - \bar{s})^T = 0 \tag{4}$$

We first normalize $X$ by subtracting the mean $\bar{X}$. We then use least squares regression to compute the correlation $\beta$ between $X - \bar{X}$ and $S$. We can then obtain the uncorrelated $\tilde{X}$ by $\tilde{X} = X - \beta S$. An $\alpha$ parameter is introduced to linearly scale the output between the fully correlated and uncorrelated extremes: $X_{out} = \alpha\tilde{X} + (1-\alpha)X$. Intuitively, a quick hyperparameter search showed that $\alpha = 1$ is most optimal as it completely removes linear correlations to the sensitive attribute.

### 3.4.2 Adversarial Debiasing

The adversarial debiasing method attempts to learn a mapping $f$ between $X$ and $\tilde{X}$ such that $\tilde{X}$ is close to $X$ and an adversary cannot classify a node's sensitive attribute based on $\tilde{X}$. Intuitively, this encourages the model to deviate slightly from a linear mapping to obfuscate any dependencies between $X$ and $S$. Although this method is more computationally expensive than the previous and requires optimization, it is able to target nonlinear dependencies in the data. Compared to the adversarial components in the FairGNN and EDITS papers, our adversarial debiasing is done outside of the graph structure and can be executed much faster.

Concretely, we train two multi-layer perceptrons (MLPs). The first MLP is the encoder and transforms $X$ to $\tilde{X}$. Unlike normal autoencoders, our encoder does not have a bottlenecks and accepts $F$ channels, has one hidden layer with $F$ nodes, and outputs $F$ channels. Our second MLP is the classifier and transforms $\tilde{X}$ to $\hat{S}$. The classifier has two hidden layers with $8$ nodes and a sigmoid activation function after the output layer. We use $25\%$ dropout to prevent overfitting.

We use mean squared error loss to guide the encoder to learn a faithful representation of the original features, and binary cross entropy to guide the classifier to learn a mapping from the modified embedding to the sensitive attribute. The encoder's loss function subtracts the classifier's loss to optimize against the classifier's target. We use two separate Adam optimizers and train the models in parallel. The encoder optimizer uses L1 regularization to encourage sparse connectivity while the classifier optimizer uses L2 regularization to learn a smooth function. We keep the learning rate small and train for 1000 epochs to allow both models to adapt to movements in the other.

## 3.5 Edge Debiasing

Although tabular deep learning fairness methods account for the unfairness in node features, the graph structure can also encode unfair biases. Higher homophily with respect to the sensitive attribute might lead to a node of the sensitive class aggregating more information from other nodes of the sensitive class. Even if a model is fair, as measured by statistical parity and equal opportunity, one sensitive class may be more likely classified into a particular label. If a node overweighs that information due to excess homophily, the model can become unfair. Given the discrete adjacency matrix, two operations are available to us: inserting and deleting edges. Inserting edges requires searching all possible $N^2$ edges for candidates, so we focus on edge deletion. We propose two heuristics to select which edges to delete. The retrained unaware node classification model is the "fair" output model of the algorithm.

### 3.5.1 Remove Unlikely Edges

The first heuristic reduces homophily by removing unlikely edges between nodes with the same sensitive attribute. To determine unlikely edges, we obtain the node embeddings in the hidden layer of the vanilla node classification GNN and take the dot product across edges to compute embedding

similarity. We remove the $20\%$ of homophilous edges whose vertices are most dissimilar. We run ablation study on the modification ratio to determine the optimal balance between removing bias and losing signal.

#### 3.5.2 Remove Impacted Edges

The second heuristic also debiases the graph by removing homophilous edges. Instead of removing the most dissimilar edges, we remove the edges whose similarity is most affected by the sensitive attribute. Intuitively, these edges were added to the graph due to the sensitive attribute, and should be removed to mask the sensitive attribute. To find these impacted edges, we train both an aware and unaware baseline models and compute the embedding similarities for both. We remove the $20\%$ of homophilous edges whose $(\text{aware similarity} - \text{unaware similarity})$ is highest. We run ablation study on the modification ratio to determine the optimal balance between removing bias and losing signal.

### 3.6 Node + Edge Debiasing

The ideal method would incorporate both edge and node debiasing. While we can run the two methods successively, we are interested in exploring a method that better meshes the two approaches. Based on experimental results (see Section 4.3), we focus on combining the adversarial node debiasing method and the impacted edge debiasing method.

The proposed method is as follows. Follow the methodology in Section 3.4.2 to obtain fair node embeddings via adversarial debiasing. Train an unaware node classification model on the fair node representations. Compute the embeddings of each node and take the dot product across edges to obtain the link likelihood with minimal impact from the sensitive attribute.

Next, train an aware node classification model on the original, unfair, node representations. Compute the embeddings of each node and take the dot product across edges to obtain the link likelihood with maximal impact from the sensitive attribute.

Take the difference between these two link likelihoods, and following the methodology in Section 3.5.2, remove the $20\%$ of homophilous edges with the greatest difference. Save the resulting edges. Finally, train a node classification model on the graph with fair node representations and modified edges. This method debiases both the nodes and edges of the input graph.

### 3.7 Metrics

#### 3.7.1 Fairness

We measure fairness with statistical parity and equal opportunity. Statistical parity (SP) represents whether the prediction Y is independent from protected attributes S.

$$\triangle_{SP} = |P(y = 1|s = 1) - P(y = 1|s = 0)| \tag{5}$$

Equal opportunity measures whether similar node embeddings with differing sensitive attributes S have similar outcomes as they should.

$$\triangle_{EO} = |P(y = 1|y = 1, s = 1) - P(y = 1|y = 1, s = 0)| \tag{6}$$

#### 3.7.2 Homophily

Homophily is defined as the property that similar nodes are more likely to share an edge than dissimilar nodes (similarity here defined by whether two nodes share the same protected attribute), and is a form of topological bias. The most common definitions of homophily are node homophily and edge homophily. Edge homophily is the fraction of edges that connect nodes of the same class:

$$h_{edge} = \frac{|u, v \in E : y_u = y_v|}{|E|} \tag{7}$$

Node homophily is the average over all nodes of the fraction of neighbors that have the same class as the node.

$$h_{node} = \frac{1}{n} \sum_{v \in V} \frac{|u \in N(v) : y_u = y_v|}{d(v)} \tag{8}$$

Excess homophily is defined as a higher level of homophily than would be expected by chance. For a sensitive attribute with probability $p$, the expected homophily is $p^2 + (1 - p)^2$.

# 4 Experiments

## 4.1 Datasets

Following existing literature [2, 1], we begin our analysis with the following datasets.

1. **German Credit**: The German Credit dataset represents 1,000 individuals. The binary classification goal is to classify individuals as at high credit or low credit risk, and the sensitive attribute is their gender. Links in the graph represent similar individuals.
2. **Pokec-z, Pokec-n**: Pokec is the most popular social network in Slovakia, similar to Facebook or Twitter. The anonymized network was collected in 2012. Our binary classification goal is to predict whether a user is employed or unemployed. The sensitive attribute will be the user's location.
3. **Recidivism**: The Recidivism dataset represents defendants released on bail (in the U.S. state court system from 1990 to 2009). Connections are related to similarity of crimes and past convictions. The goal is to predict the likelihood of a second criminal offense and the sensitive attribute is the gender.
4. **Credit**: The Credit Default Dataset consists of 30,000 nodes, with each node representing an individual who is using credit. The nodes are interconnected based on the spending and payment behaviors of the individuals. The objective of this dataset is to classify whether an individual is likely to default on their credit card payment, with the sensitive attribute as age.

In order to reflect real-world scenarios, FairGNN and EDITS significantly restricted the size of the labeled training set. We adopt identical data splits to ensure a consistent comparison with previous works. The table below summarizes key statistics for each dataset.

| Feature | German | Pokec-z | Pokec-n | Recidivism | Credit |
|---|---|---|---|---|---|
| # of Nodes | 1000 | 67796 | 66559 | 18876 | 30000 |
| # of Edges | 43484 | 1235916 | 1034094 | 623740 | 2843716 |
| # of Training Nodes | 100 | 500 | 500 | 100 | 6000 |
| # of Node Features | 27 | 277 | 266 | 18 | 13 |
| Sens Attr Frequency | 31% | 35% | 29% | 50% | 9% |
| Excess Edge Homophily | 23% | 41% | 36% | 2% | 12% |
| Excess Node Homophily | 24% | 37% | 34% | 2% | 10% |

Table 1: Dataset Summary Statistics

From Table 1, we see that the Recidivism dataset does not actually have significant excess homophily, rendering our edge debiasing methods inefficient. Additionally, the credit dataset has an extremely small sensitive attribute frequency, something we have not yet accounted for in our methods. As a result, we focus on the German, Pokec-z, and Pokec-n datasets for the time being. We hope to revisit the Recidivism and Credit datasets after modifying our methods to handle their unique features.

## 4.2 Implementation

We perform our experiments using Python, Pytorch, and Pytorch Geometric. Since the German and Pokec datasets are not available in Pytorch Geometric, we contribute Pytorch Geometric classes for both that apply the same preprocessing and train/val/test split performed in the FairGNN and EDITS papers.

Section 3.2 describes the hyperparmeters used for the base model. We use GraphSAGE for the German dataset and the Graph Convolutional Network for the Pokec-z and Pokec-n datasets. Regarding node debiasing, the correlation remover has no hyperparameters except $\alpha$ (set to 1). For the adversarial debiasing method, the encoder learning rate is $1 \times 10^{-3}$, the encoder L1 penalty is $1 \times 10^{-3}$, the classifier learning rate is $3 \times 10^{-3}$ and the classifier L2 penalty is $3 \times 10^{-3}$. Regarding the edge debiasing methods, the only hyperparameter is the modification ratio. This is initially set to $20\%$ and explored further in our ablation studies.

### 4.3 Results

We focus on three aspects of a fairness method: runtime, performance, and fairness. We measure runtime by the overhead each method adds with respect to the baseline model. Performance is measured using accuracy and F1-score. Finally, fairness is measured with statistical parity and equal opportunity.

As the German dataset is the smallest of the three, we evaluate several of our methods on it to determine the best methods in each category. We then run just those methods on the Pokec dataset and compare those methods to FairGNN and EDITS on all three datasets.

#### 4.3.1 German Dataset

Tests are run on a 2022 MacBook M2 Air. Each experiment is run 10 times and 95% confidence intervals are shown for each metric.

| Method | Runtime | Accuracy | F1-Score | Parity | Equality |
|---|---|---|---|---|---|
| Vanilla Aware | $2.16 \pm 0.03$s | $0.64 \pm 0.02$ | $0.71 \pm 0.03$ | $0.48 \pm 0.08$ | $0.43 \pm 0.08$ |
| Vanilla Unaware | $2.13 \pm 0.02$s | $0.65 \pm 0.02$ | $0.72 \pm 0.03$ | $0.36 \pm 0.07$ | $0.28 \pm 0.06$ |
| Node Correlation | $3.62 \pm 0.02$s | $0.61 \pm 0.03$ | $0.69 \pm 0.04$ | $0.39 \pm 0.12$ | $0.36 \pm 0.12$ |
| Node Adversarial | $6.53 \pm 0.10$s | $0.65 \pm 0.03$ | $0.73 \pm 0.03$ | $0.12 \pm 0.03$ | $0.04 \pm 0.02$ |
| Edge Unlikely | $5.67 \pm 0.02$s | $0.65 \pm 0.02$ | $0.72 \pm 0.04$ | $0.35 \pm 0.06$ | $0.27 \pm 0.04$ |
| Edge Impactful | $7.62 \pm 0.04$s | $0.65 \pm 0.02$ | $0.72 \pm 0.04$ | $0.31 \pm 0.05$ | $0.22 \pm 0.05$ |
| Node + Edge | $16.9 \pm 0.1$s | $0.63 \pm 0.02$ | $0.70 \pm 0.03$ | $0.06 \pm 0.03$ | $0.05 \pm 0.02$ |

Table 2: German Dataset Raw Results

Table 2 shows the results on the German dataset. Starting with runtime, the vanilla models are the simplest and have the lowest runtime, taking two seconds to train. As the methods get more complicated and require multiple trained models, the runtime increases up to almost 8 seconds. This is still far more efficient than FairGNN and EDITS, which each take several minutes to run on the German dataset. Looking at accuracy and F1-score, we see that the only fairness method that noticeably reduces performance is the node correlation method. This method also fails to reduce bias, likely overfitting to the extremely small training data. The remaining methods all reduce the bias metrics compared to the vanilla models. The Node + Edge method results in the most fair GNN.

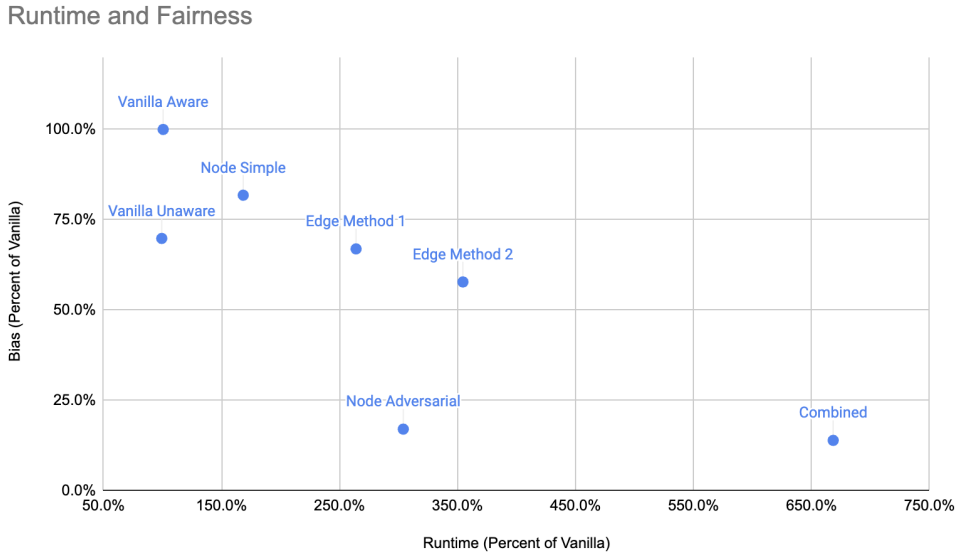


Figure 3: Scatter Plot of Bias vs Runtime. Methods that run longer are often less biased. Vanilla Unaware, Node Adversarial, and Combined Node + Edge are Pareto-Optimal

#### 4.3.2 Pokec-n Dataset

Tests are run on a T80 using the Grace cluster. Each experiment is run 3 times and 95% confidence intervals are shown for each metric.

| Method | Runtime | Accuracy | F1-Score | Parity | Equality |
|---|---|---|---|---|---|
| Vanilla Aware | $11.0 \pm 1.0$s | $0.65 \pm 0.01$ | $0.63 \pm 0.01$ | $0.18 \pm 0.01$ | $0.15 \pm 0.01$ |
| Vanilla Unaware | $10.4 \pm 1.1$s | $0.65 \pm 0.01$ | $0.63 \pm 0.01$ | $0.09 \pm 0.01$ | $0.08 \pm 0.01$ |
| Node Correlation | $18.6 \pm 0.9$s | $0.65 \pm 0.01$ | $0.63 \pm 0.01$ | $0.09 \pm 0.01$ | $0.08 \pm 0.01$ |
| Node Adversarial | $37.6 \pm 0.3$s | $0.65 \pm 0.01$ | $0.63 \pm 0.01$ | $0.09 \pm 0.01$ | $0.08 \pm 0.01$ |
| Edge Unlikely | $26.8 \pm 1.0$s | $0.65 \pm 0.01$ | $0.64 \pm 0.01$ | $0.07 \pm 0.01$ | $0.07 \pm 0.01$ |
| Edge Impactful | $37.4 \pm 0.2$s | $0.67 \pm 0.01$ | $0.64 \pm 0.01$ | $0.02 \pm 0.02$ | $0.03 \pm 0.02$ |
| Node + Edge | $73.9 \pm 1.9$s | $0.66 \pm 0.01$ | $0.64 \pm 0.01$ | $0.02 \pm 0.02$ | $0.03 \pm 0.02$ |

Table 3: Pokec-n Dataset Raw Results

Similar to the German Dataset, the runtime steadily increases as the methods become more complex. Due to a high proportion of categorical input features, the node methods struggle to debias the input in a way that a GNN cannot undo (since the methods only minorly perturb the features, the GNN can use a $0.5$ cutoff to recover the original features). The edge debiasing methods are more effective, with the second approach outperforming the first. Although not statistically significant, the Node + Edge method slightly outperformed either Edge method for each random seed.

#### 4.3.3 Pokec-z Dataset

Tests are run on a T80 using the Grace cluster. Each experiment is run 3 times and 95% confidence intervals are shown for each metric.

| Method | Runtime | Accuracy | F1-Score | Parity | Equality |
|---|---|---|---|---|---|
| Vanilla Aware | $11.5 \pm 1.0$s | $0.68 \pm 0.01$ | $0.69 \pm 0.02$ | $0.13 \pm 0.01$ | $0.11 \pm 0.01$ |
| Vanilla Unaware | $10.9 \pm 1.1$s | $0.68 \pm 0.01$ | $0.69 \pm 0.01$ | $0.08 \pm 0.01$ | $0.07 \pm 0.01$ |
| Node Correlation | $19.8 \pm 0.5$s | $0.68 \pm 0.01$ | $0.69 \pm 0.01$ | $0.08 \pm 0.01$ | $0.07 \pm 0.01$ |
| Node Adversarial | $38.0 \pm 0.3$s | $0.68 \pm 0.01$ | $0.69 \pm 0.01$ | $0.08 \pm 0.01$ | $0.07 \pm 0.01$ |
| Edge Unlikely | $28.8 \pm 0.3$s | $0.66 \pm 0.01$ | $0.68 \pm 0.01$ | $0.06 \pm 0.01$ | $0.01 \pm 0.01$ |
| Edge Impactful | $38.9 \pm 0.8$s | $0.68 \pm 0.02$ | $0.68 \pm 0.02$ | $0.04 \pm 0.02$ | $0.05 \pm 0.01$ |
| Node + Edge | $75.1 \pm 0.9$s | $0.68 \pm 0.02$ | $0.68 \pm 0.01$ | $0.04 \pm 0.03$ | $0.04 \pm 0.01$ |

Table 4: Pokec-z Dataset Raw Results

Results are very similar to for the Pokec-z dataset unsurprisingly. The main difference is that the first edge debiasing method outperforms the second, a reversal of the Pokec-n dataset results. Further study is required to explore why this happened, and if any actionable insights can be gained. Like the previous two datasets, the Node + Edge model improves fairness the most, though at the cost of a larger runtime (though still significantly faster than FairGNN and EDITS).

### 4.4 Comparison with Baselines

We successfully cloned both FairGNN and EDITS locally and verified results from the papers. While we found no issues with FairGNN, we did notice that node features for the German dataset were not correctly normalized in the EDITS paper. Additionally, due to outdated frameworks and undocumented code, we do not run on either model any dataset not originally tested by the authors. Instead, we have ensured that our implementation of their datasets use the same random seed and result in an identical train/val/test split.

| Dataset | Metric | Vanilla Aware | Vanilla Unaware | FairGNN | EDITS | Ours |
|---|---|---|---|---|---|---|
| German | AUC | 71.02% | 70.56% | - | 73.21% | 64.85% |
| | F1 | 71.29% | 71.83% | - | 80.62% | 71.49% |
| | SP | 44.83% | 45.84% | - | 8.30% | 8.43% |
| | EO | 43.06% | 27.88% | - | 3.75% | 4.41% |
| Pokec-n | AUC | 69.21% | 69.64% | 74.9% | 61.82% | 70.22% |
| | F1 | 62.59% | 62.61% | - | 52.84% | 63.82% |
| | SP | 17.79% | 9.43% | 0.80% | 0.91% | 2.06% |
| | EO | 15.25% | 8.44% | 1.90% | 1.10% | 2.75% |
| Pokec-z | AUC | 73.30% | 73.13% | 76.70% | 67.83% | 72.11% |
| | F1 | 68.92% | 68.66% | - | 61.91% | 67.76% |
| | SP | 12.99% | 8.08% | 0.90% | 2.74% | 3.85% |
| | EO | 10.66% | 6.57% | 1.70% | 2.87% | 4.12% |

Table 5: Correctness and Fairness metrics for five GNN models on three node classification datasets.

Starting with the performance metrics, we see that EDITS has the highest AUC and F1-score in the German dataset (likely due to the bug mentioned above regarding data preprocessing), and FairGNN has the highest AUC for both Pokec datasets (but doesn't report F1-score). As the vanilla aware and unaware models shown are ours, we hypothesize that differences in the data preprocessing and/or base model account for most of the difference in the performance metrics. All the models report minimal performance degradation as a result of the fairness methods.

Moving to the fairness metrics, we see that all three models drastically outperform both the vanilla aware and unaware baselines. On the German dataset, our method is slightly more biased than the EDITS method (within error bars). On the Pokec-n and Pokec-z datasets, our methods are slightly more biased than both FairGNN and EDITS. Despite requiring significantly less time to train and run, our Node + Edge model is fairly competitive with the two state-of-the-art methods.

### 4.5 Ablation Study

We view testing the 7 variations of our model as the primary ablation study, with the result being that the adversarial node and impactful edge debiasing methods are critical to the method's success. We find that the Node + Edge method performs the best on all three datasets tested. In addition, we conduct a brief ablation study on the optimal modification ratio for the edge debiasing methods. The table below summarizes the modification ratio and the performance and fairness metrics of the "Edge Impactful" model. We use this model instead of the Node + Edge model to better isolate the effect due to the modification ratio.

| Modification Ratio | Excess Homophily | Accuracy | F1-Score | Parity | Equality |
|---|---|---|---|---|---|
| 0 | 0.2326 | 0.6560 | 0.7244 | 0.4008 | 0.3193 |
| 0.1 | 0.2155 | 0.6560 | 0.7221 | 0.3667 | 0.2668 |
| 0.2 | 0.1952 | 0.6680 | 0.7365 | 0.3348 | 0.2132 |
| 0.3 | 0.1705 | 0.6520 | 0.7203 | 0.2921 | 0.1796 |
| 0.4 | 0.1399 | 0.6240 | 0.6948 | 0.2729 | 0.1639 |
| 0.5 | 0.1012 | 0.6280 | 0.6990 | 0.2623 | 0.1723 |
| 0.6 | 0.0504 | 0.6360 | 0.7111 | 0.1984 | 0.1008 |
| 0.7 | -0.0192 | 0.6400 | 0.7078 | 0.1877 | 0.1019 |
| 0.8 | -0.1202 | 0.6560 | 0.7226 | 0.1664 | 0.0746 |
| 0.9 | -0.2802 | 0.6480 | 0.7197 | 0.1920 | 0.1355 |
| 1.0 | -0.5720 | 0.6600 | 0.7284 | 0.2197 | 0.1176 |

Table 6: Modification Ratio Ablation Study

This table shows that as the modification ratio increases, parity and equality decreases, indicating a more fair graph. Additionally, the F1-score initially hold steady and slightly increases before decreasing as the modification ratio increases. After the modification ratio passes $0.5$, the results are mostly constant as the graph devolves into disconnected nodes. These results indicate that a modification ratio of $0.2$ or $0.3$ is optimal.

## 5 Conclusion

We investigated the problem of GNN fairness, with a specific emphasis on efficiency and scalability to large datasets. We chose to focus on graph preprocessing for model-agnostic fairness, and created several node and edge debiasing approaches. After experimenting with multiple combinations of methods, we created a single method that combines adversarial node debiasing and removes homophilous edges most influenced by the sensitive attribute.

We evaluated our baseline models, node debiasing models, edge debiasing models, and combined debiasing model on three real-world datasets with up to $60,000$ nodes and $1.2$ million edges. We evaluated our methods on computational cost, performance, and fairness. We found that both our node and edge methods improved fairness on a majority of the datasets, with the combined method having the least bias on all three datasets. Our methods also did not significantly reduce performance and only marginally increased runtime, especially compared to the existing baselines.

Compared to the FairGNN and EDITS state-of-the-art results, our combined node and edge debiasing approach falls slightly short in performance and fairness. However, given it's computational cost, we believe it is the simplest and most effective fairness approach for large scale datasets. Moving forward, we hope to continue refining our methodology and test our graph preprocessing on larger social network datasets not previous considered for GNN fairness.

## 6 Reproduciblity

Visit the GitHub at `https://github.com/avgupta456/cpsc680-final`.

## References


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

[2] Yushun Dong, Ninghao Liu, Brian Jalaian, and Jundong Li. Edits: Modeling and mitigating data bias for graph neural networks. In *Proceedings of the ACM Web Conference 2022*, pages 1259–1269, 2022.