FairEGM: Fair Link Prediction and Recommendation via Emulated Graph Modification

SEAN CURRENT, Ohio State University, USA
YUNTIAN HE, Ohio State University, USA
SAKET GURUKAR, Ohio State University, USA
SRINIVASAN PARTHASARATHY, Ohio State University, USA

As machine learning becomes more widely adopted across domains, it is critical that researchers and ML engineers think about the inherent biases in the data that may be perpetuated by the model. Recently, many studies have shown that such biases are also imbibed in Graph Neural Network (GNN) models if the input graph is biased, potentially to the disadvantage of underserved and underrepresented communities. In this work, we aim to mitigate the bias learned by GNNs by jointly optimizing two different loss functions: one for the task of link prediction and one for the task of demographic parity. We further implement three different techniques inspired by graph modification approaches: the Global Fairness Optimization (GFO), Constrained Fairness Optimization (CFO), and Fair Edge Weighting (FEW) models. These techniques mimic the effects of changing underlying graph structures within the GNN and offer a greater degree of interpretability over more integrated neural network methods. Our proposed models emulate microscopic or macroscopic edits to the input graph while training GNNs and learn node embeddings that are both accurate and fair under the context of link recommendations. We demonstrate the effectiveness of our approach on four real world datasets and show that we can improve the recommendation fairness by several factors at negligible cost to link prediction accuracy.

Additional Key Words and Phrases: graph representation learning, demographic parity, group fairness, graph convolution neural networks, graph neural networks, graph fairness, link prediction, link recommendation

ACM Reference Format:
Sean Current, Yuntian He, Saket Gurukar, and Srinivasan Parthasarathy. 2022. FairEGM: Fair Link Prediction and Recommendation via Emulated Graph Modification. In *Equity and Access in Algorithms, Mechanisms, and Optimization (EAAMO '22), October 6–9, 2022, Arlington, VA, USA.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3551624.3555287

1 INTRODUCTION

The rapid development and widespread application of machine learning (ML) models in the day-to-day life of individuals highlights the pressing need to incorporate fairness in model formulation and development. An increasing number of such ML models operate on graph data. Of these ML models, graph neural networks (GNNs) [17] have been shown to be effective on several machine learning tasks on graphs such as node classification [26], link prediction [11], and graph visualization [34]. Due to their effectiveness, GNNs have been utilized in various applications such as recommendation engines [36], drug discovery [37], and predicting social network relations [18].

While GNNs demonstrate excellent performance on graph machine learning, these models can learn and propagate bias present in the input graph and amplify it. Given their prevalence, it is important to study and rectify biases these models learn from implicit or explicit biases in the input graph. For example, recent studies have shown that standard

Manuscript submitted to ACM

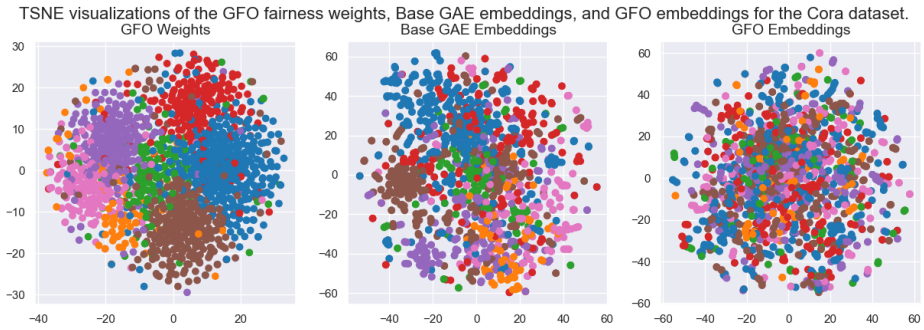

Fig. 1. A TSNE[33] visualization of weights learned by the GFO method (Section 4.1) as well as node embeddings produced by unfair and fair embedding methods on the Cora dataset. The color of a data-point indicates the value of the sensitive attribute for the corresponding node. Left: $\boldsymbol{W}_f$ weights learned by the GFO method. Middle: embeddings produced by a standard GAE autoencoder. Right: embeddings produced by the GFO embedding method. We observe that with the Base GAE embedding method, points with the same sensitive attribute are grouped more closely together, and thus links between points with the same sensitive attribute are more likely to be suggested by a link recommendation algorithm. In contrast, sensitive attributes in the GFO embeddings are more uniformly distributed, leading to more diversified recommendations with respect to the target sensitive attribute. This is achieved by learning weights which are added to nodes in the network to encourage fair representations; these weights inherently group according to the sensitive attribute features that they "correct" in order to make the node embeddings fair, as visualized in the left figure.

GNNs for downstream tasks like link prediction learn node representations that perform poorly on various fairness measures [1, 21]. A common source of bias in this context is the homophilic effect, which observes that nodes tend to associate with similar peers [22] based on demographic characteristics such as race, ethnicity, sex, and religion. Homophily has recently been pointed out in the social sciences as a leading cause of the *glass ceiling effect* [3, 31]. More specifically, it can lead to unfair treatment of historically disadvantaged and underserved communities in multiple cases, including ranking [15], social perception [20], and job promotion [6, 32]. Apart from the structural properties of graphs, feature information in the graphs is also a source of bias. Even if a sensitive feature is excluded, bias can still be inherited from other features closely correlated with the sensitive one [25]. Thus, it is paramount to explicitly model for and measure such sources of bias in downstream tasks like link prediction.

One can learn fair node representations and mitigate disparate sources of biases present in the input graph by incorporating fairness constraints in GNN training. Given the proliferation of graph-based recommendation models [36], we incorporate fairness constraints through demographic parity [8]. Informally, demographic parity seeks to ensure that each group with a particular sensitive attribute receives the positive outcome at the same rate as other groups with different values for the same sensitive attribute [8]. For example, a professional networking website can utilize a GNN-based recommendation engine to recommend a job opening to a certain number of individuals. Demographic parity based recommendations could ensure that no individuals of a particular race, ethnicity, or gender are less likely to receive such a recommendation. The same can be said for social networking sites; if a recommendation algorithm is not fair under the definition of demographic parity, then the algorithm will suggest individuals with similar demographics to be linked, reinforcing preexisting ties and limiting diversity in a social setting. In contrast, an algorithm that is fair under the context of demographic parity would guarantee that recommendations are made with such diversity in mind. We also acknowledge that there exist contexts where biased recommendations that prioritize certain demographics can be beneficial (such as for the purpose of strengthening minority communities)[7]. In these instances, the application of

demographic parity is not suitable; however, for the general level of link recommendations on a global scale, optimizing demographic parity can be a useful way to improve fairness and diversity in downstream applications.

We incorporate demographic parity in GNN-based recommendation models in an model-agnostic manner by proposing a novel *Link Divergence* loss function that can be used in tandem with pre-existing link recommendation losses. We incorporate the optimization of Link Divergence through three separate methods inspired by graph modifications. Our techniques emulate the effect of adding new nodes and edges to the input graph without the computational overhead of actually doing so, offering both efficiency and interpretability. When used in tandem with the proposed Link Divergence loss, these methods facilitate the learning of fair graph node embeddings for downstream tasks of link prediction and recommendation. We present **E**mulated **G**raph **M**odifications for **Fair**ness (**FairEGM**), a collection of three methods which emulate the effects of a variety of graph modifications for the purpose of improving graph fairness. The Global Fairness Optimization (GFO) method introduces a fairness-oriented bias to every node in the dataset, which is optimized for fairness separately from the link prediction task. The Constrained Fairness Optimization (CFO) method introduces a rank-deficient weight matrix which is added as a bias to nodes in the graph, and is similarly optimized for fairness without regard to the link prediction task. Finally, the Fair Edge Weighting (FEW) method introduces edge weights to existing edges in the graph, and optimizes the edge weights to debias the input graph.

In our experiments, we concretely observe that applying models to biased data without regard to fairness can result in biased node representations that cluster based on the sensitive attribute under consideration, as shown in Figure 1, middle. In contrast, our fair embedding models result in node representations that do not cluster by sensitive attribute (Figure 1, right) while maintaining performance on downstream tasks by introducing corrective weights specific to nodes in the graph (Figure 1, left). We show that each of or methods is capable of improving fairness across four real-world datasets and can significantly reduce the biases learned by GNNs. The reduced bias (or increased fairness) comes with a slight reduction in the link prediction performance. We also show that **FairEGM** is fairer than other graph learning baselines on the link prediction task.

2 RELATED WORK

Graph Representation Learning. Various methods for graph representation learning for node embeddings have been proposed, such as Node2Vec [11], DeepWalk [26], matrix factorization approaches [10], and graph neural network (GNN) approaches [17]. Methods such as Node2Vec and DeepWalk construct embeddings using techniques inspired by word embeddings in natural language processing; these methods first generate random walks throughout the graph structure, forming sequences of nodes. These sequences are treated as "sentences", with the nodes as "words" in the sentence, allowing for the use of traditional word embedding algorithms to learn node embeddings. In contrast, matrix embedding approaches operate on the adjacency matrix of the graph $\boldsymbol{A}$, hoping to learn a set of embeddings $\boldsymbol{\Phi}$ such that $\boldsymbol{\Phi}\boldsymbol{\Phi}^\top = \boldsymbol{A}$.

However, when applied to attributed graphs, these methods are generally inferior to graph convolution networks. GCNs utilize the adjacency matrix $\boldsymbol{A}$ and a layer input $\boldsymbol{H}$ (usually the feature matrix $\boldsymbol{F}$ in the first layer of a GCN or the output of the previous layer $\boldsymbol{H}^{(\ell)}$) to learn node embeddings. Node embeddings $\boldsymbol{\Phi} = \boldsymbol{H}^{(\ell+1)}$ are extracted from the final layer in the GCN, and are learned by directly optimizing a weight matrix $\boldsymbol{W}$ for the task such that

$$\boldsymbol{H}^{(\ell+1)} = \sigma(\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(\ell)}\boldsymbol{W}),$$

where σ is the sigmoid function (or some other nonlinear activation function, often ReLU) and $\boldsymbol{D} = \sum_j A_{ij}$ is a normalization vector for A. These methods have the added benefit of utilizing message-passing and semi-supervised

learning in the embedding process, strengthening their performance compared to other methods on graph representation learning tasks. We will focus on GCNs as the primary embedding method for this paper.

Fairness in Machine Learning. Recent work in machine learning has demonstrated the capability of ML models to learn implicit biases present in the data, such as systemic racism in criminal recidivism models [2], facial recognition technology [5], and gender biases in machine translation [30]. As the usage of ML models across domains becomes increasingly common, it remains critical that researchers consider the fairness of their models.

Several formulations of fairness have been proposed in the ML community [23], [8]. The first and most basic formulation is to ignore sensitive attributes with attribute unaware fairness: if a model cannot use a sensitive attribute in its prediction, the model is fair [8]. However, this formulation is flawed due to correlations that may exist between sensitive attributes and other features in the dataset, such as zip-code when determining credit approval; due to systemic racism and historic segregation, some zip-codes are more strongly associated with specific races than others, which results in zip-code becoming a proxy for race.

This leads to the formulations of demographic parity and equalized odds [8], which propose specific constraints on the performance of a model. Demographic parity requires that members of different protected classes appear in the positive class at the same rate; the distribution of protected attributes of members in the positive class should match the population distribution. On the other hand, equalized odds is less focused on the model outcome, but rather on model performance; the true positive rates should be equal across protected attributes. This guarantees that a model achieves similar performance across protected attributes. In contrast to the prior definitions of fairness, the individual fairness [8] definition does not depend on sensitive attributes, but rather the similarity of members. Individual fairness dictates that similar individuals should have similar outcomes in the model.

Fairness in Graph Embeddings. The analysis of fairness in graph mining has received much attention in recent years. To address the bias in graph learning models, several models have been proposed [1, 4, 14, 21]. Bose et al. [4] propose an adversarial method to ensure graph embeddings do not contain information that can be used to discern an individual's protected class. Rahman et al. [28] implement the FairWalk algorithm, which improves upon random walk algorithms by more fairly traversing the graph structure based upon the sensitive attributes of nodes.

In the realm of individual fairness, InFoRM [14] recognizes three approaches to implementing individual fairness constraints: debiasing the input graph, debiasing the mining model, and debiasing the mining result. To debias the input graph, Kang et al. construct an algorithm to optimize the adjacency matrix of graph data to improve individual fairness. Comparatively, the recent work of Li et al. [21] proposes the FairAdj algorithm for group fairness metrics, which attempts to learn a fair adjacency matrix while preserving predictive accuracy for a dyadic link prediction task under structural constraints. Similarly, FairDrop [29] randomly drops edges in the graph while improving fairness based on dyadic attributes, also modifying the adjacency matrix.

Much of the previous work on debiasing the input graph focuses purely on modifying the existing adjacency matrix, in contrast to our work, which additionally considers debiasing the input graph by emulating the addition of new artificial nodes and edges. Not only is this approach unique, but it also allows for new information to be added to the network, contrasting approaches centered around optimizing the adjacency matrix, which are constricted to leveraging information already present in the graph. Additionally, there is opportunity to learn more about the features of the network through the analysis of added nodes: by understanding how the added information debiases the input graph, we can better understand biases present in the original graph.

3 PROBLEM STATEMENT

Our formulations consider an undirected graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ with vertex set $\boldsymbol{V}$ containing n nodes and edge set $\boldsymbol{E} \subseteq \boldsymbol{V} \times \boldsymbol{V}$. We notate the self-connected adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, the degree matrix $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ and the normalized adjacency matrix $\hat{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A} \boldsymbol{D}^{-\frac{1}{2}}$. The node feature matrix is written as $\boldsymbol{F} \in \mathbb{R}^{n \times m}$, where m is the dimensionality of the feature set, and the one-hot encoded sensitive attribute matrix as $\boldsymbol{S} \in \mathbb{N}^{n \times k}$ where k is the number of possible values the sensitive attribute can take on (note that in the current formulation, only a single sensitive attribute with k possible values is considered). The GNN weight matrix for layer i is notated $\boldsymbol{W}^{(i)} \in \mathbb{R}^{l_{i-1} \times l_i}$, where l_i is the hidden layer size and l_{i-1} is the size of the previous layer (m if $i = 0$). The embedding matrix $\boldsymbol{\Phi}^{(i)} \in \mathbb{R}^{n \times l_i}$ is equal to the output of layer i in the GNN. $\boldsymbol{\Phi} \in \mathbb{R}^{n \times d}$ is the output of the final layer of the GNN. We further represent a generic GNN architecture as $\boldsymbol{\mathcal{G}}$.

Definition 1 (Graph Representation Learning [13])**.** Given a graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ with feature matrix $\boldsymbol{F}$ and a dimensionality $d \ll |V|$, graph representation learning aims to learn a function $h_{\boldsymbol{G}} : V \rightarrow \mathbb{R}^d$ such that $\boldsymbol{\Phi} = h_{\boldsymbol{G}}(V)$ is a matrix of d-dimensional vector representations for nodes in the graph such that the similarity among nodes in the graph space is approximated by similarity between nodes in the embedding space.

The learned node embeddings can be used as latent features in various downstream tasks. In this work, we focus on the fair link prediction task. We select demographic parity [8] as our fairness criteria. Informally, demographic parity is satisfied if the output of the model is not dependent on a given sensitive attribute [23]. Formally, we define the demographic parity fairness criteria on the link prediction problem as follows.

Definition 2 (Link Prediction with Demographic Parity Fairness)**.** Given a graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, a node v, and an embedding model M, let $L_M(v) = (u_1, ..., u_n)$ be the set of nodes that have the highest likelihood to form a link with node v where $L_M(v)$ is computed using the model M. The link prediction problem with demographic parity fairness for node v has the following constraint: $D(P_{L_M(v)}, P_S) = 0$ where D is the distance metric between distributions, $P_{L_M(v)}$ is the distribution of sensitive attributes over the recommended nodes set $L_M(v)$, and P_S is the distribution of sensitive attributes on the overall graph.

The link prediction problem with demographic parity fairness states that the distribution of sensitive attributes over recommended nodes ($L_M(v)$) should not be distinguishable from the distribution of sensitive attributes on the overall graph (P_S). Note that the model can perform link recommendation by performing either K-nearest neighbors or by using a classifier.

Problem Statement: We want to learn a graph neural network (GNN) model M such that M performs well on the link prediction task while satisfying the demographic parity fairness constraint (defined in Definition 2).

4 METHODOLOGY

To address the problem defined in Section 3, we train our proposed models with two objective functions. In the first objective function, we optimize the model for utility (better performance on the link prediction task), while in the second objective function, we optimize the model for fairness (demographic parity based recommendations). Our objective functions are described as follows:

Utility objective. We use matrix reconstruction loss as our utility objective function due to its effectiveness in link prediction [18] and recommendation systems [24, 27]. Specifically, following Kipf et al. [18], we use the sigmoid of the dot product between node embeddings to reconstruct the matrix. This reconstruction task acts as a proxy for the link

prediction task. We minimize the difference between the reconstructed and original adjacency matrix by optimizing the node embeddings.

Mathematically, our utility objective is described as follows:

$$L_R(\mathbf{\Phi}) = \boldsymbol{W}_{pos} \circ H(\boldsymbol{A}, \sigma(\mathbf{\Phi}\mathbf{\Phi}^\top)), \tag{1}$$

where H is the binary cross entropy function and $\boldsymbol{W}_{pos}$ is an element-wise weighting term, defined as

$$\boldsymbol{W}_{pos} = (\boldsymbol{A} \cdot \frac{\|\boldsymbol{V}\|^2 - 2\|\boldsymbol{E}\|}{2\|\boldsymbol{E}\|} + (1 - \boldsymbol{A})). \tag{2}$$

$\boldsymbol{W}_{pos}$ balances positive and negative edges in the graph, placing more weight on positive edges when the graph is sparse, and less weight on positive edges when it is highly connected.

Fairness objective. In order to achieve demographic parity for the link prediction task, we require each group present in the sensitive attribute set to receive the positive outcome at the same rate. In other words, the distribution of sensitive attributes in the positive outcome should match the population distribution of sensitive attributes. In this case, we define a positive outcome as a positive prediction of a link between nodes: $\sigma(\mathbf{\Phi}_v\mathbf{\Phi}_u^\top) > 0.5$ indicates a positive outcome for node u given node v. We first define a function $f(v)$ to evaluate the distribution of similarity scores for sensitive attributes in relation to the node $v \in \boldsymbol{V}$. Specifically, $f(v)$ computes the total similarity score for all nodes $u \in \boldsymbol{V}\backslash\{v\}$ grouped by a specific sensitive attribute value $\boldsymbol{S}_u$ divided by the total number of nodes in that group. Based on this insight, we propose the link divergence loss function L_D to measure the sum of KL-divergences between the population distribution P_S of the sensitive attributes and $f(v)$ for each node:

$$L_D(\mathbf{\Phi}) = \sum_{v \in \boldsymbol{V}} D_{KL}\left(P_S \parallel f(v)\right), \tag{3}$$

where D_{KL} is the KL-divergence function and

$$f(v) = \frac{1}{\|\boldsymbol{V}\| - 1} \sum_{u \in \boldsymbol{V}\backslash\{v\}} \sigma(\mathbf{\Phi}_v\mathbf{\Phi}_u^\top) \cdot \boldsymbol{S}_u, \tag{4}$$

where $\boldsymbol{S}_u$ is a one-hot encoded vector representing the sensitive attribute of node u. Note that $f(v)$ is normalized before the KL-divergence is calculated. The L_D loss function encourages weights to learn demographic parity: when L_D is minimized, the two distributions $f(v)$ and P_s are equal, directly enforcing demographic parity.

Graph Convolution: Each of our approaches uses a graph convolution operation with the symmetrically normalized adjacency matrix containing self-loops as the initial step. Given a node v with neighbors $\mathcal{N}(v)$ and initial feature/embedding matrix $\mathbf{\Phi}^{(i)}$, the graph convolution operation is defined as the aggregative step

$$\mathbf{\Phi}_v^{(i+1)} = \sigma\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{e_{u,v}}{\sqrt{\hat{d}_u \hat{d}_v}} \mathbf{\Phi}_v^{(i)} \cdot W^{(i)}\right) \tag{5}$$

where $\mathbf{\Phi}_v^{(i+1)}$ is the embedding of node v at layer $(i+1)$ and $\hat{d}_v = 1 + sum_{u \in \mathcal{N}(v)} \cdot e_{u,v}$. Rewriting this operation in matrix form, we have

$$\mathbf{\Phi}^{(i+1)} = \sigma(\hat{\boldsymbol{A}}\mathbf{\Phi}^{(i)}\boldsymbol{W}), \tag{6}$$

where $\hat{\boldsymbol{A}}$ is the symmetrically normalized adjacency matrix containing self-loops. We use this representation when defining our formulations.

Algorithm 1 Joint optimization procedure.

Input: node features $\boldsymbol{F}$, normalized adjacency matrix $\hat{\boldsymbol{A}}$, sensitive attribute matrix $\boldsymbol{S}$, GNN utility weights $\boldsymbol{W}$, GNN fairness weights $\boldsymbol{W}_f$, learning rate η, and fairness weight λ_f.

while $\boldsymbol{W}$ or $\boldsymbol{W}_f$ has not converged **do**
  Compute L_R according to Eqn. 1
  Set $g_{\boldsymbol{W}} \leftarrow \nabla_{\boldsymbol{W}} L_R$
  Set $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \cdot \text{Adam}(\boldsymbol{W}, g_{\boldsymbol{W}})$
  Compute L_D according to Eqn. 3
  Set $g_{\boldsymbol{W}_f} \leftarrow \lambda_f \cdot \nabla_{\boldsymbol{W}_f} L_D$
  Set $\boldsymbol{W}_f \leftarrow \boldsymbol{W}_f - \eta \cdot \text{Adam}(\boldsymbol{W}, g_{\boldsymbol{W}})$
end while

Output: Fair node embeddings $\boldsymbol{\Phi} \leftarrow \text{GNN}_{\boldsymbol{W},\boldsymbol{W}_f}(\boldsymbol{F}, \hat{\boldsymbol{A}})$

Optimization Process: In our formulations, we want to constrain optimization of the reconstruction loss and link divergence to separate weight terms. To do this, each formulation constructs two optimization problems dependent on separate weight terms solved using a joint optimization. The reconstruction loss weights are only changed according to the gradient of the reconstruction loss and similarly the link divergence weights are changed only according to the gradient of link divergence.

Additionally, we introduce a hyper-parameter $\lambda_f \in (0, \infty)$ to weigh the impact of the fairness loss in the above optimization process. By scaling the gradients of the link divergence by a factor of λ_f, we retain a level of control over the impact of the fairness optimizations on the model. Naturally, due to inherent biases present in data, many models suffer performance losses when fairness is highly weighted [19]. As such, λ_f allows one to balance the importance of fairness and utility in model application. Larger values of λ_f will scale up the gradients of the link divergence loss, placing a larger weight on the fairness optimization during training, while lower values of λ_f will let the optimization place more emphasis on the reconstruction loss. This optimization process is outlined in algorithm 1.

4.1 Global Fairness Optimization (GFO)

We first consider emulating a graph modification resulting from the introduction of a new partner node for each node in the graph, which is only connected to the node it is partnered with. On a conceptual level, the partner node v^* is responsible for balancing out biases present in the original node v.

We introduce a new set of n nodes $\boldsymbol{V}^*$ with accompanying n edges $\boldsymbol{E}^* = \{(v_i, v_i^*)\}_{i=1}^n$ each with edge weight 1 and features $\boldsymbol{F}^*$ initialized from a Glorot normal distribution [9]. We add $\boldsymbol{V}^*$ and $\boldsymbol{E}^*$ to the original graph $\boldsymbol{G}$ to construct a modified graph $\tilde{\boldsymbol{G}} = (\boldsymbol{V} \cup \boldsymbol{V}^*, \boldsymbol{E} \cup \boldsymbol{E}^*)$ with feature matrix $\tilde{\boldsymbol{F}} = \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{F}^* \end{bmatrix}$ and adjacency matrix $\tilde{\boldsymbol{A}} = \begin{bmatrix} \hat{\boldsymbol{A}} & \boldsymbol{A}^* \\ \boldsymbol{A}^* & 0 \end{bmatrix}$, where $\boldsymbol{A}^*$ is the $n \times n$ matrix with edge weights on the diagonal and 0 elsewhere.

Applying the graph convolution operation on the modified graph $\tilde{\boldsymbol{G}}$, we obtain node embeddings

$$\tilde{\boldsymbol{\Phi}} = \begin{bmatrix} \boldsymbol{\Phi} \\ \boldsymbol{\Phi}^* \end{bmatrix} = \mathcal{G}(\sigma(\tilde{\boldsymbol{A}}\tilde{\boldsymbol{F}}\boldsymbol{W})), \tag{7}$$

where σ is a nonlinear activation and $\mathcal{G}$ represents subsequent GNN layers. Since we do not need to learn resulting node embeddings for the introduced artificial nodes, we can simplify Equation 7 to only calculate embeddings for nodes

in the original graph:

$$\Phi = \mathcal{G}(\sigma(\left[\hat{A}\ A^*\right]\tilde{F}W)) = \mathcal{G}(\sigma((\hat{A}F + A^*F^*)W)). \tag{8}$$

We can further simplify the equation by distributing the edge weights of A^* to construct the fairness optimization weights $W_f = A^*F^*$. Substituting W_f into Equation 8, we obtain the final GFO embedding formulation

$$\Phi = \mathcal{G}(\sigma((\hat{A}F + W_f)W)). \tag{9}$$

Note that since there are no constraints on the values of W_f, the GFO formulation equates to direct modification of node features following the convolution operation.

This results in the following optimization problems, dependent on $\mathcal{G}, W$ (weights for the utility objective), and W_f (weights for the fairness objective):

$$\min_{\mathcal{G},W}(L_R(\Phi)) \text{ and } \min_{W_f}(L_D(\Phi)) \tag{10}$$

4.2 Constrained Fairness Optimization (CFO)

Next, we consider a generalization of the GFO approach. Here, instead of introducing a partner node for every node in the graph, we introduce a finite set of c new nodes connected to each node in the original graph. On a conceptual level, the c new nodes form a basis by which the biases of nodes can be corrected through new connections to a diverse set of nodes. We use the notation CFO_c to represent the CFO method using c additional nodes.

We introduce V^* with c nodes, $E^* = \{(v_i, v_j^*)\}_{i=1}^{n}{}_{j=1}^{c}$ with $n \cdot c$ edges, and features F^*. The features and edge weights are initialized with Glorot normal initialization [9]. We add V^* and E^* to the original graph G to construct a modified graph $\tilde{G} = (V \cup V^*, E \cup E^*)$ with feature matrix $\tilde{F} = \begin{bmatrix} F \\ F^* \end{bmatrix}$ and adjacency matrix $\tilde{A} = \begin{bmatrix} \hat{A} & A^* \\ A^{*\top} & 0 \end{bmatrix}$, where A^* is the $n \times c$ matrix of edge weights.

Following the same formulation of the graph convolution operation as the GFO method, we obtain the following output of the first GNN layer for CFO:

$$\Phi = \mathcal{G}(\sigma((\hat{A}F + A^*F^*)W)). \tag{11}$$

There are now two separate formulations that can take place depending on the number of added nodes c. If c is less than both the number of original nodes n and the number of features m, the $n \times m$ matrix A^*F^* is guaranteed to be rank deficient:

$$\begin{aligned} \text{rank}(A^*F^*) &\leq \min(\text{rank}(A^*), \text{rank}(F^*)) \\ &\leq \min(\min(n, c), \min(c, m)) \\ &= c < \min(n, m). \end{aligned}$$

Due to the rank deficiency of A^*F^*, we cannot generalize the product (A^*F^*) into a single weight matrix (W_f) as we did in Equation 9. As a result, one needs to maintain the inherent constraints of rank deficiency during the optimization process in the CFO formulation. Hence, the matrices A^* and F^* must be optimized separately to ensure A^*F^* cannot achieve full rank. This results in the following optimization problems, dependent on $\mathcal{G}, W, A^*$, and F^*:

$$\min_{\mathcal{G},W}(L_R(\Phi)) \text{ and } \min_{A^*,F^*}(L_D(\Phi)) \tag{12}$$

In contrast, we can consider the case where c is greater than or equal to either the number of original nodes n or the number of features m. When $c \geq n$ or $c \geq m$, the rank of the matrix product A^*F^* is limited by $\min(n, m)$. Because this

formulation is unconstrained by the value of c, we can simplify Equation 11 to match the GFO solution (Equation 9) by introducing the same weight matrix, $\boldsymbol{W}_f = \boldsymbol{A}^* \boldsymbol{F}^*$. Thus, we observe that the GFO formulation is a special case of CFO.

4.3 Fair Edge Weighting (FEW)

In both GFO and CFO methods, we mitigate the bias present in the input graph by emulating the introduction of new nodes. In the FEW method, we mitigate bias learned by the GCN by editing edge weights in the existing graph. Edge weights in a graph act as a weighting function when learning the node embedding for a node v from its neighbors $u \in \mathcal{N}(v)$. The edge weight for edge (u, v) determines the degree to which the features of node u contribute to the embedding of node v. By scaling the edge weights in the adjacency matrix, we can place more emphasis on particular edges that could correct the bias for nodes in the graph. On a conceptual level, FEW balances edge weights in the existing graph to correct the bias present in the input data.

We introduce an edge weight matrix, $\boldsymbol{W}_f$, to modify the existing normalized adjacency matrix $\hat{\boldsymbol{A}}$ prior to the graph convolution operation:

$$\boldsymbol{\Phi} = \mathcal{G}(\sigma((\hat{\boldsymbol{A}} \circ \boldsymbol{W}_f)\boldsymbol{F}\boldsymbol{W})). \tag{13}$$

Because the normalized adjacency matrix $\hat{\boldsymbol{A}}$ is expected to have values of 0 for non-existent edges, the element-wise multiplication operation will only introduce weights on existing edges. This results in the following optimization problems, dependent on $\mathcal{G}$, $\boldsymbol{W}$, and $\boldsymbol{W}_f$:

$$\min_{\mathcal{G}, \boldsymbol{W}}(L_R(\boldsymbol{\Phi})) \text{ and } \min_{\boldsymbol{W}_f}(L_D(\boldsymbol{\Phi})) \tag{14}$$

Note that in this formulation of FEW, there are no constraints on $\boldsymbol{W}_f$, so introduced weights have a range of $(-\infty, \infty)$ and are not necessarily symmetric, allowing FEW to construct a directed graph.

5 EXPERIMENTS

5.1 Dataset

We conduct our experiments on a set of four real-world datasets (see Table 1 for details, including sensitive attribute information). Edges in the Citeseer, Cora, and Pubmed datasets (https://linqs.soe.ucsc.edu/data) represent paper citations and edges in the Facebook-1684 ego-network dataset (https://snap.stanford.edu/data/) represent Facebook friendships. Citeseer, Cora, and Pubmed have bag-of-word feature vectors for each node while Facebook-1684 has anonymized features for each node representative of various attributes of a person's Facebook profile.

Dataset	Nodes	Edges	Features	Sensitive Attribute	Clustering Coefficient
Citeseer	3,327	4,732	3,703	Topic (6)	0.2407
Cora	2,708	5,278	1,433	Topic (7)	0.1426
Facebook	786	14,024	317	Gender (2)	0.4757
Pubmed	19,717	44,327	500	Topic (3)	0.0602

Table 1. Dataset statistics for the four experimental datasets. The number (n) next to the sensitive attribute label indicates how many values the sensitive attribute may take on.

5.2 Experimental Setup

For each dataset, we train a basic graph convolution network (GCN), a GCN with GFO optimization, two GCNs with CFO_{10} ($c = 10$) and CFO_{100} ($c = 100$) optimization, and a GCN with FEW optimization. All models use a two-layer GAE autoencoder[18] as the GCN model. Following prior work [18], each GAE has a 32-dim hidden layer and a 16-dim embedding layer. We compare our methods to FairWalk [28] and FairAdj [21] using similar parameters. We offer an additional experiment in Appendix A, which compares our models to the base GAE model using an augmented loss function $L(\mathbf{\Phi}) = L_R(\mathbf{\Phi}) + \lambda L_D(\mathbf{\Phi})$.

Link predictions models are trained using 20 randomized train-test splits. Following prior works [12] for link prediction tasks, we split the dataset into training and test data using the following procedure. We first randomly sample 20% of edges and add them to the test set. From the subgraph consisting of the remaining edges, we extract its largest connected component as the training data. Finally, we remove the nodes that are not in the training data but are present in the test set from the test set.

Training hyperparameters are chosen separately for each model based on reconstruction loss optimization. All values of the fairness weight λ are set to 1. Losses are optimized with an Adam optimizer [16] with β_1 and β_2 kept at the default values of 0.9 and 0.999, respectively, while the learning rate α and number of training epochs are tuned according to the datasets. Hyperparameters for each dataset are listed in the Table 2. Any hyperparameters not listed are kept at their default values from the source code.

Experiments for the Pubmed dataset are run on a high-performance computer with Intel Xeon E5-2680 v4 CPUs (128GB memory) and NVIDIA Tesla P100 GPUs (16GB memory). All other datasets are run on a laptop with Intel Core i7-9750H CPUs (16GB memory) and an NVIDIA GeForce GTX 1650 GPU (4GB memory).

5.3 Metrics

Loss: We evaluate all methods based on reconstruction loss L_R and link divergence L_D, as discussed previously.
Quality: To evaluate embedding quality on the link prediction task, we use the AUROC and F1-Score metrics following the procedures described in [12]. We train a logistic regression model as a classifier for positive/negative edges using an equal number of positive and negative edges randomly selected from the training set. The AUROC and F1-Score are then recorded using the logistic regression predictions for positive edges and an equal number of randomly selected negative edges from the test set.
Fairness: To evaluate performance on the demographic parity fairness task for link recommendation, we compute DP@k as follows: for each node u in graph $\boldsymbol{G}$, we utilize the learned d-dimension embedding vector to find the k-nearest nodes (denoted as kNN(u)) in the embedding space using the sigmoid of the dot-product similarity score. From the k-nearest nodes kNN(u), we calculate the distribution π(kNN(u)) of the observed sensitive attributes and compare to

Dataset	GCN Models		FairAdj		FairWalk	
	Learning Rate	Epochs	Learning Rate	T2	Learning Rate	Epochs
Citeseer	0.0001	300	0.005	10	0.01	1
Cora	0.0001	300	0.001	10	0.1	1
Facebook	0.0001	300	0.01	10	0.1	1
Pubmed	0.001	200	0.005	10	0.1	1

Table 2. Model hyperparameters for the four experimental datasets.

Dataset	Model	$L_R \downarrow$	$L_D \downarrow$	AUROC ↑	F1 ↑
Citeseer	Base	1.31 ± 0.0399	0.000743 ± 0.00247	**0.74** ± 0.0295	0.616 ± 0.0259
	FairWalk	6.05 ± 0.308	0.000134 ± 3.47e-05	0.697 ± 0.0302	**0.628** ± 0.0366
	FairAdj	**0.832** ± 0.00447	0.00227 ± 0.00021	0.678 ± 0.0262	0.596 ± 0.0264
	GFO	1.34 ± 0.0142	**3.12e-08** ± 4.74e-08	0.714 ± 0.0263	0.599 ± 0.0263
	CFO_{10}	1.34 ± 0.0125	9.88e-07 ± 9.14e-07	0.714 ± 0.0339	0.599 ± 0.0354
	CFO_{100}	1.34 ± 0.014	6.71e-08 ± 6.66e-08	0.706 ± 0.0544	0.595 ± 0.0339
	FEW	1.33 ± 0.0242	0.000124 ± 0.000161	0.718 ± 0.0579	0.608 ± 0.0345
Cora	Base	1.29 ± 0.0188	0.000117 ± 0.000159	0.73 ± 0.0205	0.608 ± 0.0199
	FairWalk	6.89 ± 0.135	0.000453 ± 6.55e-05	0.624 ± 0.0158	0.573 ± 0.0141
	FairAdj	**0.937** ± 0.0109	0.00547 ± 0.000376	0.573 ± 0.0309	0.554 ± 0.0215
	GFO	1.31 ± 0.00939	**1.04e-07** ± 6.5e-08	**0.731** ± 0.0182	**0.613** ± 0.0183
	CFO_{10}	1.3 ± 0.0155	7.71e-06 ± 5.44e-06	**0.731** ± 0.0204	**0.613** ± 0.019
	CFO_{100}	1.3 ± 0.0116	4.88e-07 ± 7.19e-07	0.729 ± 0.0207	0.609 ± 0.022
	FEW	1.29 ± 0.0188	0.000101 ± 0.000112	0.726 ± 0.0226	0.607 ± 0.0242
Facebook	Base	1.25 ± 0.0295	1.43e-05 ± 1.65e-05	0.786 ± 0.0117	0.72 ± 0.0116
	FairWalk	2.26 ± 0.0333	0.000182 ± 3.07e-05	0.723 ± 0.00861	0.693 ± 0.00743
	FairAdj	**0.837** ± 0.00208	0.000686 ± 5.58e-05	0.759 ± 0.00851	0.711 ± 0.00733
	GFO	1.27 ± 0.0189	6.89e-08 ± 5.99e-08	**0.788** ± 0.0094	0.72 ± 0.00808
	CFO_{10}	1.26 ± 0.0185	4.45e-07 ± 3.96e-07	0.786 ± 0.00998	0.72 ± 0.00948
	CFO_{100}	1.26 ± 0.0177	**3.01e-08** ± 1.66e-08	0.787 ± 0.00966	**0.721** ± 0.00699
	FEW	1.24 ± 0.0409	1.65e-05 ± 2.34e-05	0.787 ± 0.00998	**0.721** ± 0.00918
Pubmed	Base	1.31 ± 0.00417	2.95e-06 ± 2.72e-06	**0.847** ± 0.00431	**0.724** ± 0.00499
	FairWalk	8.58 ± 0.0568	0.000302 ± 1.92e-05	0.724 ± 0.00612	0.622 ± 0.00585
	FairAdj	**0.875** ± 0.00224	0.0045 ± 0.000141	0.63 ± 0.00934	0.562 ± 0.0104
	GFO	1.31 ± 0.0179	2.22e-09 ± 3.17e-09	0.829 ± 0.0717	0.716 ± 0.0314
	CFO_{10}	1.31 ± 0.0181	4.56e-09 ± 3.44e-09	0.83 ± 0.0758	0.718 ± 0.0323
	CFO_{100}	1.32 ± 0.0242	1.84e-09 ± **1.5e-09**	0.812 ± 0.104	0.711 ± 0.0444
	FEW	1.31 ± 0.0176	5.41e-08 ± 7.84e-08	0.834 ± 0.053	0.716 ± 0.0318

Table 3. Utility in Link Prediction: losses and link prediction metrics for all datasets. The highest performing model for each dataset and metric is bolded.

the global distribution P_S of sensitive attributes of the dataset. The metric is defined for a node u as follows:

$$\text{DP@}k(u) = D_{KL}\left(P_S \parallel \pi(\text{kNN}(u))\right) \tag{15}$$

where $\pi(\text{kNN}(u))$ represents the normalized distribution of sensitive attribute values in the nearest neighbors of u and P_S is the distribution of the sensitive attributes in the overall dataset. The final overall metric DP@k is the average DP@$k(u)$ for all nodes u in the dataset:

$$\text{DP@}k = \frac{1}{|V|}\sum_u \text{DP@}k(u). \tag{16}$$

Ideally, this value should be as close to zero as possible. When S is a binary sensitive attribute, the DP@k metric is additionally a suitable metric for dyadic fairness, and is similar to the ΔDP metric proposed by [21].

5.4 Results

Results for link prediction are reported in Table 3. Across the board, our methods consistently perform near or above state-of-the-art levels for link prediction while additionally optimizing for link divergence. While FairAdj reports a lower reconstruction loss for all datasets, the AUROC and F1-scores do not behave similarly. For the Citeseer dataset, the AUROC and F1-scores are slightly better for FairAdj compared to the GFO, CFO_{10}, CFO_{100} and FEW models, while the reverse is true for Cora, Facebook, and Pubmed, which report the fair autoencoder models scoring significantly

higher on AUROC, particularly for the Cora and Pubmed datasets. We further note that the GFO, CFO_{10}, CFO_{100}, and FEW models do not consistently perform significantly better or worse than the base model they are built on, indicating that the fairness optimizations made by the models do not seem to significantly impact embedding performance.

Results for demographic parity in link recommendation with the DP@k metrics are presented in Table 4. We observe that for the Citeseer dataset, the GFO method offers the greatest improvement in the DP@k metrics compared to the base GCN model while maintaining similar AUROC and F1 scores. All of the fair autoencoders improve upon the base GCN method. This is similarly true for the Cora, Facebook, and Pubmed datasets, where the GFO method consistently ranks among the top models for the DP@k metrics, though is arguably out-shined by the CFO_{100} method for the Cora dataset. We additionally note that as k increases, the DP@k metrics decrease for all methods, indicating that as more of the nearest nodes are considered, the distribution of sensitive attributes becomes fairer and more representative of the population.

Additionally, our models perform better than Fairwalk and FairAdj for the Citeseer, Cora, and Facebook datasets, and better than FairAdj for the Pubmed dataset. Fairwalk achieves generally stronger DP@k scores on the Pubmed dataset; however, this performance is offset by weaker AUROC and F1 scores, representative of the inherent trade-off between fairness and utility.

The FEW method does not appear as capable of improving fairness as the GFO and CFO methods. This is further observed in Figure 2, which documents the reconstruction and link divergence losses during training for the various methods on the Pubmed dataset. As the FEW model continues training, Link Divergence asymptotically converges to a higher value than the GFO and CFO methods. In contrast, the GFO and CFO methods are better able to optimize Link Divergence in the same number of epochs.

T-SNE[33] is a visualization technique that maps high-dimensional data to lower dimensions while preserving natural clusters. Using T-SNE, we observe that the GFO method produces node embeddings that do not naturally organize according to the value of the sensitive attribute in contrast to standard GCN autoencoder methods, which appear more visually separable under a T-SNE representation

Dataset	Model	DP@10 ↓	DP@20 ↓	DP@40 ↓
Citeseer	Base	4.88 ± 1.79	3.01 ± 1.69	1.85 ± 1.4
	FairWalk	6.96 ± 0.79	5.31 ± 0.742	2.9 ± 0.557
	FairAdj	6.54 ± 0.747	4.55 ± 0.762	2.16 ± 0.445
	GFO	2.3 ± 1.23	**0.706** ± 0.506	**0.172** ± 0.201
	CFO_{10}	**1.94** ± 0.791	0.879 ± 0.527	0.234 ± 0.285
	CFO_{100}	2.34 ± 1.17	0.71 ± 0.692	0.271 ± 0.516
	FEW	4.17 ± 1.35	2.2 ± 1.28	1.17 ± 0.937
Cora	Base	3.79 ± 1.06	1.89 ± 0.934	0.914 ± 0.652
	FairWalk	7.35 ± 0.137	5.4 ± 0.208	3.37 ± 0.312
	FairAdj	7.09 ± 0.189	4.73 ± 0.175	2.51 ± 0.167
	GFO	3.47 ± 0.826	1.05 ± 0.348	**0.132** ± 0.0486
	CFO_{10}	3.38 ± 0.84	1.23 ± 0.536	0.239 ± 0.123
	CFO_{100}	**3.32** ± 0.808	**0.96** ± 0.424	0.147 ± 0.0713
	FEW	3.81 ± 0.874	1.87 ± 0.618	0.719 ± 0.42
Facebook	Base	0.136 ± 0.161	0.0418 ± 0.0204	0.0226 ± 0.0102
	FairWalk	0.239 ± 0.0764	0.0451 ± 0.00856	0.0226 ± 0.00254
	FairAdj	0.411 ± 0.114	0.0802 ± 0.0264	0.0329 ± 0.00335
	GFO	**0.0661** ± 0.0484	**0.0191** ± 0.0114	0.0076 ± 0.00446
	CFO_{10}	0.17 ± 0.329	0.0226 ± 0.00984	**0.0072** ± 0.00366
	CFO_{100}	0.101 ± 0.0643	0.0246 ± 0.0133	0.00783 ± 0.00441
	FEW	0.217 ± 0.391	0.0413 ± 0.0252	0.0171 ± 0.00914
Pubmed	Base	2.43 ± 1.36	0.64 ± 1.09	0.185 ± 0.0859
	FairWalk	**1.12** ± 0.0701	0.368 ± 0.0405	0.176 ± 0.0184
	FairAdj	5.72 ± 0.108	4.46 ± 0.136	3.3 ± 0.148
	GFO	1.52 ± 1.56	0.327 ± 0.72	0.196 ± 0.0851
	CFO_{10}	2.61 ± 1.27	**0.174** ± 0.0407	0.203 ± 0.0694
	CFO_{100}	2.06 ± 1.51	0.182 ± 0.0322	**0.168** ± 0.0906
	FEW	2.31 ± 1.38	0.328 ± 0.697	0.196 ± 0.0686

Table 4. Fairness in Link Recommendations: DP@k metrics for all datasets. The highest performing model for each dataset and metric is bolded.

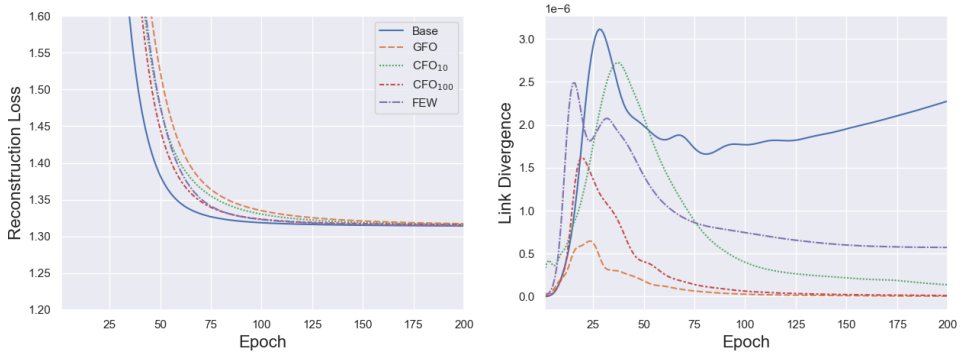

Fig. 2. Left: Reconstruction loss during training for the Pubmed dataset. Right: Link divergence loss during training for the Pubmed dataset.

as shown in Figure 1. Additionally, a T-SNE analysis of the learned $\boldsymbol{W}_f$ weights for GFO demonstrates that the method learns similar biases for nodes which share a sensitive attribute, as evidenced by the strong clustering of node-specific weight values observed in the left plot of Figure 1.

Comparing the CFO_{10} and CFO_{100} methods, we observe that the CFO_{100} method consistently achieves a lower L_D loss and generally performs better than the CFO_{10} method in regards to the fairness metrics. This comparison suggests that higher values of c (the number of added nodes in CFO) are more capable of optimizing for fairness; this claim is further supported by the GFO performance, which can be considered a bound on the performance of CFO models due to the relation between the solution spaces for CFO and GFO fair weight matrices discussed in section 4.2. To investigate this further, we run the CFO method with a wider range of values of c.

5.5 Impact of the parameter c in CFO

Results for a wider variety of CFO_c models are shown in Figure 3 of Appendix B for the Cora dataset. As c increases, we see a slight increase in reconstruction loss; however, this does not correlate with an increase in AUROC, which steadily hovers around 0.73, indicating models with higher c still perform well on the link prediction task. However, higher c models generally exhibit stronger performance on the fairness-related metrics. As the number of introduced nodes c increases, the Link Divergence loss decays exponentially, indicating CFO models with higher c are more capable of optimizing for fairness. This is paralleled in the observed DP@40 metric, which similarly decreases as c increases. After $c \approx 40$, Link Divergence and DP@40 stop decreasing significantly, indicating the number of introduced nodes has saturated and any further increases in c will offer minimal returns for the Cora dataset.

5.6 Efficiency of Approaches

We first acknowledge that the time complexity of training a base L-layer GCN model is $O(L(|E|d + nd^2))$ [35], where $|E|$ is the number of edges, d is the size of the embedding dimension, and n is the number of nodes. In each layer, GCN performs $O(|E|)$ convolution operations with a sparse implementation then followed by a non-linear transformation, which takes $O(nd^2)$ time.

The time complexity of our GFO approach remains $O(L(|E|d + nd^2))$. In each layer, GFO introduces an additive bias term into each node's representation after the convolution, which takes $O(nd)$ time. Since $n \ll |E|$, this computation does not increase the time complexity.

For the CFO approach, the time complexity is $O(max(|E|, cn)d + (L-1)|E|d + Lnd^2)$. CFO adds an additional $n \cdot c$ edges into the first layer of the graph adjacency matrix, where c is the number of additional nodes. With a sparse implementation of GCN, the additional cn convolutions in the first layer takes $O(ncd)$ time. When c is not large, the total time complexity varies little compared to the base GCN.

Similar to the GFO approach, the FEW method does not affect the overall efficiency of the graph convolution algorithm. The FEW method introduces a weight for each edge in the input graph, which scale the adjacency matrix before the graph convolution step. This does not affect the big-O runtime, which will remain $O(L(|E|d + nd^2))$ for a sparse matrix implementation.

6 CONCLUSION

Standard graph representation learning methods built for the task of link recommendation can learn node representations that are unfair and biased towards historically disadvantaged and underserved communities, leading to unfair treatment in ranking [15], social perception [20], and job promotion [6, 32]. In order to support equity and opportunity for said communities, we propose a set of three methods built to encourage diversity and equity in graph representation learning for link recommendation.

While many works have expanded the graph representation learning domain, few have considered fair graph representations and the graph modifications necessary to construct such embeddings. Existing works such as InFoRM [14], FairWalk [28], and FairAdj [21] focus on introducing fairness by adjusting the input adjacency matrix; in contrast, our work emulates the effects of three unique graph modification methods, including methods that introduce new nodes into the graph, along with a novel loss function to address demographic parity in the graph embedding domain.

Notably, all three of our fairness methods demonstrate significant ability to improve demographic parity on the link recommendation task with negligible loss (and in some cases small gains) in link prediction performance. We show that for a set of four datasets, each formulation is able to increase graph fairness under the definition of demographic parity through effects similar to the introduction of new nodes or edge weights. Additionally, our methods are separable from the base GCN architecture used in the embedding learning, allowing users to extract information about the emulated graph modifications from the embedding model. In the future, we hope to demonstrate the flexibility of our formulations on a wider variety of GCN architectures, such as VGAE [18].

We additionally plan to investigate the types of modifications our methods learn and emulate, so that we may better understand how bias is represented and remedied in these datasets. We aspire to explore ways we might impose constraints on our fair learning methods to better simulate the introduction of realistic nodes and edges; while our current methods significantly improve demographic parity, introduced artificial nodes are free to take on whatever features will do the job, regardless of if those features could exist. Finally, we intend to construct new losses for optimization in order to utilize our emulated graph modification techniques to improve fairness on other embedding tasks, such as node classification, or for other forms of fairness, such as equalized odds or individual fairness.

Acknowledgements: This material is partially supported by the National Science Foundation (NSF) under grants OAC-2018627, CCF-2028944, and CNS-2112471 and the Air Force Office of Scientific Research (AFOSR) under grant FA8650-19-2-2204. Any opinions, findings, and conclusions in this material are those of the author(s) and may not reflect the views of the respective funding agencies.

REFERENCES

[1] Chirag Agarwal, Himabindu Lakkaraju, and Marinka Zitnik. 2021. Towards a Unified Framework for Fair and Stable Graph Representation Learning. *arXiv preprint arXiv:2102.13186* (2021).

[2] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. 2016. Machine bias. (2016).

[3] Chen Avin, Barbara Keller, Zvi Lotker, Claire Mathieu, David Peleg, and Yvonne-Anne Pignolet. 2015. Homophily and the glass ceiling effect in social networks. In *Proceedings of the 2015 conference on innovations in theoretical computer science*. 41–50.

[4] Avishek Bose and William Hamilton. 2019. Compositional fairness constraints for graph embeddings. In *International Conference on Machine Learning*. PMLR, 715–724.

[5] Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*. PMLR, 77–91.

[6] Sara M Clifton, Kaitlin Hill, Avinash J Karamchandani, Eric A Autry, Patrick McMahon, and Grace Sun. 2019. Mathematical model of gender bias and homophily in professional hierarchies. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 29, 2 (2019), 023135.

[7] Francesco Fabbri, Francesco Bonchi, Ludovico Boratto, and Carlos Castillo. 2020. The effect of homophily on disparate visibility of minorities in people recommender systems. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 14. 165–175.

[8] Pratik Gajane and Mykola Pechenizkiy. 2017. On formalizing fairness in prediction with machine learning. *arXiv preprint arXiv:1710.03184* (2017).

[9] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 9)*, Yee Whye Teh and Mike Titterington (Eds.). PMLR, Chia Laguna Resort, Sardinia, Italy, 249–256. https://proceedings.mlr.press/v9/glorot10a.html

[10] Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* 151 (2018), 78–94. https://doi.org/10.1016/j.knosys.2018.03.022

[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[12] Saket Gurukar, Priyesh Vijayan, Aakash Srinivasan, Goonmeet Bajaj, Chen Cai, Moniba Keymanesh, Saravana Kumar, Pranav Maneriker, Anasua Mitra, Vedang Patel, Balaraman Ravindran, and Srinivasan Parthasarathy. 2019. Network Representation Learning: Consolidation and Renewed Bearing. arXiv:1905.00987 [cs.LG]

[13] William L. Hamilton. [n.d.]. Graph Representation Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 14, 3 ([n. d.]), 1–159.

[14] Jian Kang, Jingrui He, Ross Maciejewski, and Hanghang Tong. 2020. InFoRM: Individual Fairness on Graph Mining. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 379–389.

[15] Fariba Karimi, Mathieu Génois, Claudia Wagner, Philipp Singer, and Markus Strohmaier. 2018. Homophily influences ranking of minorities in social networks. *Scientific reports* 8, 1 (2018), 1–12.

[16] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[17] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[18] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. arXiv:1611.07308 [stat.ML]

[19] Jon Kleinberg. 2018. Inherent trade-offs in algorithmic fairness. In *Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems*. 40–40.

[20] Eun Lee, Fariba Karimi, Claudia Wagner, Hang-Hyun Jo, Markus Strohmaier, and Mirta Galesic. 2019. Homophily and minority-group size explain perception biases in social networks. *Nature human behaviour* 3, 10 (2019), 1078–1087.

[21] Peizhao Li, Yifei Wang, Han Zhao, Pengyu Hong, and Hongfu Liu. 2021. On dyadic fairness: Exploring and mitigating bias in graph connections. In *Proceedings of International Conference on Learning Representations*.

[22] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27, 1 (2001), 415–444.

[23] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2019. A Survey on Bias and Fairness in Machine Learning. *CoRR* abs/1908.09635 (2019). arXiv:1908.09635 http://arxiv.org/abs/1908.09635

[24] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th International Conference on Data Mining*. IEEE, 497–506.

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

[26] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[27] Adityanarayanan Radhakrishnan, George Stefanakis, Mikhail Belkin, and Caroline Uhler. 2021. Simple, Fast, and Flexible Framework for Matrix Completion with Infinite Width Neural Networks. arXiv:2108.00131 [cs.LG]

[28] Tahleen Rahman, Bartlomiej Surma, Michael Backes, and Yang Zhang. 2019. Fairwalk: Towards Fair Graph Embedding. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 3289–3295. https://doi.org/10.24963/ijcai.2019/456

[29] Indro Spinelli, Simone Scardapane, Amir Hussain, and Aurelio Uncini. 2021. Biased Edge Dropout for Enhancing Fairness in Graph Representation Learning. arXiv:2104.14210 [cs.LG]

[30] Gabriel Stanovsky, Noah A Smith, and Luke Zettlemoyer. 2019. Evaluating gender bias in machine translation. *arXiv preprint arXiv:1906.00591* (2019).

[31] Ana-Andreea Stoica, Christopher Riederer, and Augustin Chaintreau. 2018. Algorithmic Glass Ceiling in Social Networks: The effects of social recommendations on network diversity. In *Proceedings of the 2018 World Wide Web Conference*. 923–932.

[32] Bonnie J Tesch, Helen M Wood, Amy L Helwig, and Ann Butler Nattinger. 1995. Promotion of women physicians in academic medicine: glass ceiling or sticky floor? *Jama* 273, 13 (1995), 1022–1025.

[33] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 86 (2008), 2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html

[34] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 1225–1234.

[35] 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* 32, 1 (2020), 4–24.

[36] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Jul 2018). https://doi.org/10.1145/3219819.3219890

[37] Marinka Zitnik and Jure Leskovec. 2017. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics* 33, 14 (2017), i190–i198.

A AUGMENTED LOSS MODEL

We present an additional experiment to compare the GFO, CFO, and FEW models to a standard GAE autoencoder trained using an augmented loss function of the form $L(\mathbf{\Phi}) = L_R(\mathbf{\Phi}) + \lambda L_D(\mathbf{\Phi})$. This modified loss function encourages the GAE model to learn both the utility and fairness tasks without introducing additional weights or terms into the GNN. Because the combined loss is highly sensitive to λ for balancing utility and fairness, we test the AUG models with values of $\lambda = 10^i$ for $i = 0, 1, ..., 5$. This serves as an additional baseline for our models.

Tables 5 and 6 present the results of the augmented loss models (AUG_λ) on the utility and fairness metrics, respectively. In contrast to the GFO, CFO, and FEW models, the AUG model with $\lambda = 1$ is not able to offer significantly fairer embeddings over the base GAE model, and offers only slight improvements in the fairness metrics while maintaining a similar degree of utility as the base model. However, as λ increases, there is a noticeable shift in the balance of fairness and utility in the AUG models. Models with higher values of λ tend to have stronger performance on fairness metrics.

Comparing the AUG to the GFO, CFO, and FEW models, we find that the AUG models perform similarly to GFO, CFO, and FEW on the link recommendation task, supporting earlier findings in Section 5.4 that the fairness optimizations learned by these models do not significantly impact embedding performance. In regards to fairness related performance, at least one of the GFO, CFO, and FEW models performs better or comparable to the AUG models on the Citeseer, Facebook, and Pubmed datasets. However, we do observe stronger performance for the $\text{AUG}_1$0000 and $\text{AUG}_1$00000 models on the Cora dataset for the DP10 and DP20 metrics. For DP40, the GFO and CFO_{100} models are once again the strongest performers.

Overall, these results show that restricting the optimization of the Link Divergence loss to a specific set of weights in the GFO and CFO methods does not significantly impede the ability of these models to learn fairer embeddings. Additionally, these experimental results offer a fourth method of improving fairness with respect to demographic parity in graph embeddings via the augmented loss function $L(\mathbf{\Phi}) = L_R(\mathbf{\Phi}) + \lambda L_D(\mathbf{\Phi})$, which perform comparably or slightly better than the GFO, CFO, and FEW models at the cost of reduced interpretability.

Dataset	Model	$L_R \downarrow$	$L_D \downarrow$	AUROC ↑	F1 ↑
Citeseer	Base	**1.31** ± 0.0399	0.000741 ± 0.00246	**0.74** ± 0.0295	**0.616** ± 0.0261
	GFO	1.34 ± 0.0145	3.21e-08 ± 4.85e-08	0.714 ± 0.0263	0.599 ± 0.0261
	CFO_{10}	1.34 ± 0.0125	9.88e-07 ± 9.14e-07	0.714 ± 0.0339	0.599 ± 0.0354
	CFO_{100}	1.34 ± 0.014	6.7e-08 ± 6.65e-08	0.706 ± 0.0544	0.596 ± 0.0339
	FEW	1.33 ± 0.0242	0.000124 ± 0.000161	0.718 ± 0.0579	0.608 ± 0.0345
	AUG_1	1.33 ± 0.0237	0.000121 ± 0.000133	0.703 ± 0.0687	0.602 ± 0.0377
	AUG_{10}	1.33 ± 0.0208	0.000105 ± 0.000106	0.714 ± 0.0604	0.605 ± 0.0339
	AUG_{100}	1.34 ± 0.0187	2.29e-05 ± 1.6e-05	0.71 ± 0.053	0.597 ± 0.0278
	AUG_{1000}	1.34 ± 0.0128	2.54e-06 ± 1.34e-06	0.707 ± 0.0518	0.597 ± 0.0252
	AUG_{10000}	1.35 ± 0.014	1.78e-07 ± 1.39e-07	0.704 ± 0.057	0.599 ± 0.0345
	AUG_{100000}	1.35 ± 0.0195	**2.29e-08** ± 2.17e-08	0.665 ± 0.0932	0.584 ± 0.036
Cora	Base	**1.29** ± 0.0188	0.000117 ± 0.000159	0.73 ± 0.0205	0.608 ± 0.0199
	GFO	1.31 ± 0.00939	**1.04e-07** ± 6.5e-08	**0.731** ± 0.0182	**0.613** ± 0.0183
	CFO_{10}	1.3 ± 0.0155	7.71e-06 ± 5.44e-06	**0.731** ± 0.0204	**0.613** ± 0.0188
	CFO_{100}	1.3 ± 0.0116	4.88e-07 ± 7.2e-07	0.729 ± 0.0207	0.609 ± 0.022
	FEW	**1.29** ± 0.0188	0.000101 ± 0.000112	0.726 ± 0.0226	0.607 ± 0.0242
	AUG_1	**1.29** ± 0.0165	9.75e-05 ± 7.34e-05	0.73 ± 0.0186	0.613 ± 0.0214
	AUG_{10}	**1.29** ± 0.0227	0.000109 ± 9.66e-05	0.73 ± 0.0192	0.606 ± 0.0183
	AUG_{100}	1.3 ± 0.0136	3.68e-05 ± 1.95e-05	0.727 ± 0.0196	0.611 ± 0.0203
	AUG_{1000}	1.3 ± 0.0145	9.53e-06 ± 5.09e-06	0.728 ± 0.0239	0.606 ± 0.0236
	AUG_{10000}	1.32 ± 0.00953	1.52e-06 ± 9.64e-07	0.729 ± 0.0195	0.61 ± 0.0206
	AUG_{100000}	1.32 ± 0.0106	2.05e-07 ± 1.57e-07	0.727 ± 0.0216	0.603 ± 0.0165
Facebook	Base	1.25 ± 0.0295	1.43e-05 ± 1.65e-05	0.786 ± 0.0117	0.72 ± 0.0116
	GFO	1.27 ± 0.0189	6.89e-08 ± 5.99e-08	0.788 ± 0.0094	0.72 ± 0.00813
	CFO_{10}	1.26 ± 0.0185	4.45e-07 ± 3.96e-07	0.786 ± 0.00998	0.72 ± 0.00944
	CFO_{100}	1.26 ± 0.0177	**3.01e-08** ± 1.66e-08	0.787 ± 0.00966	0.721 ± 0.00704
	FEW	**1.24** ± 0.0409	1.65e-05 ± 2.34e-05	0.787 ± 0.00998	0.721 ± 0.00921
	AUG_1	1.25 ± 0.0308	1.38e-05 ± 1.18e-05	0.788 ± 0.00639	0.721 ± 0.00508
	AUG_{10}	**1.24** ± 0.0325	1.65e-05 ± 1.33e-05	0.787 ± 0.00911	0.721 ± 0.00746
	AUG_{100}	**1.24** ± 0.0455	2.19e-05 ± 4.08e-05	0.787 ± 0.00953	0.721 ± 0.00897
	AUG_{1000}	1.25 ± 0.0256	8.77e-06 ± 8.68e-06	0.788 ± 0.00907	0.721 ± 0.00942
	AUG_{10000}	1.27 ± 0.0122	8.94e-07 ± 3.53e-07	**0.791** ± 0.00893	**0.724** ± 0.00678
	AUG_{100000}	1.28 ± 0.0133	1.15e-07 ± 9.5e-08	0.788 ± 0.00686	0.723 ± 0.00444
Pubmed	Base	1.31 ± 0.00417	2.95e-06 ± 2.72e-06	**0.847** ± 0.00431	**0.724** ± 0.00499
	GFO	1.31 ± 0.0179	2.22e-09 ± 3.17e-09	0.829 ± 0.0717	0.716 ± 0.0314
	CFO_{10}	1.31 ± 0.0181	4.56e-09 ± 3.44e-09	0.83 ± 0.0758	0.718 ± 0.0323
	CFO_{100}	1.32 ± 0.0242	**1.84e-09** ± 1.5e-09	0.812 ± 0.104	0.711 ± 0.0444
	FEW	1.31 ± 0.0176	5.41e-08 ± 7.84e-08	0.834 ± 0.053	0.716 ± 0.0318
	AUG_1	1.32 ± 0.0241	2.68e-06 ± 2.83e-06	0.81 ± 0.104	0.708 ± 0.0428
	AUG_{10}	1.31 ± 0.00723	3.54e-06 ± 3.3e-06	0.846 ± 0.00606	**0.724** ± 0.00671
	AUG_{100}	1.32 ± 0.0238	2.36e-06 ± 4.04e-06	0.81 ± 0.104	0.709 ± 0.0446
	AUG_{1000}	1.31 ± 0.0244	2.93e-07 ± 2.36e-07	0.813 ± 0.104	0.71 ± 0.0434
	AUG_{10000}	**1.3** ± 0.00639	4.24e-08 ± 4.45e-08	**0.847** ± 0.00438	**0.724** ± 0.00496
	AUG_{100000}	1.32 ± 0.0236	1.12e-08 ± 9.8e-09	0.816 ± 0.0923	0.71 ± 0.0418

Table 5. Utility in Link Prediction: losses and link prediction metrics for the augmented loss model on all datasets. Prior results for the GFO, CFO, and FEW methods are repeated for ease of comparison. The highest performing model for each dataset and metric is bolded.

Dataset	Model	DP@10 ↓	DP@20 ↓	DP@40 ↓
Citeseer	Base	4.88 ± 1.79	3.01 ± 1.69	1.85 ± 1.4
	GFO	2.3 ± 1.23	**0.703** ± 0.497	**0.17** ± 0.197
	CFO_{10}	**1.94** ± 0.791	0.879 ± 0.527	0.234 ± 0.285
	CFO_{100}	2.35 ± 1.17	0.71 ± 0.692	0.271 ± 0.516
	FEW	4.17 ± 1.35	2.2 ± 1.28	1.17 ± 0.937
	AUG_1	4.28 ± 1.35	2.26 ± 1.27	1.38 ± 1.12
	AUG_{10}	4.76 ± 1.87	2.76 ± 1.86	1.57 ± 1.34
	AUG_{100}	3.15 ± 1.33	1.65 ± 0.902	0.593 ± 0.433
	AUG_{1000}	2.25 ± 0.892	0.98 ± 0.8	0.398 ± 0.585
	AUG_{10000}	2.09 ± 0.962	0.861 ± 0.667	0.395 ± 0.504
	AUG_{100000}	2.11 ± 1.39	0.933 ± 0.993	0.574 ± 0.802
Cora	Base	3.79 ± 1.06	1.89 ± 0.934	0.914 ± 0.652
	GFO	3.47 ± 0.826	1.05 ± 0.348	**0.132** ± 0.0486
	CFO_{10}	3.38 ± 0.84	1.23 ± 0.536	0.239 ± 0.123
	CFO_{100}	3.32 ± 0.808	0.96 ± 0.424	0.147 ± 0.0716
	FEW	3.81 ± 0.874	1.87 ± 0.618	0.719 ± 0.42
	AUG_1	3.9 ± 0.925	1.82 ± 0.8	0.798 ± 0.552
	AUG_{10}	4.12 ± 0.966	2.05 ± 0.84	0.918 ± 0.569
	AUG_{100}	3.64 ± 0.654	1.37 ± 0.531	0.454 ± 0.284
	AUG_{1000}	3.65 ± 0.861	1.28 ± 0.45	0.237 ± 0.132
	AUG_{10000}	3.24 ± 1.1	1.06 ± 0.695	0.178 ± 0.194
	AUG_{100000}	**2.76** ± 1.14	**0.862** ± 0.539	0.206 ± 0.277
Facebook	Base	0.136 ± 0.161	0.0418 ± 0.0204	0.0226 ± 0.0102
	GFO	**0.0661** ± 0.0484	**0.0191** ± 0.0114	0.0076 ± 0.00446
	CFO_{10}	0.17 ± 0.329	0.0226 ± 0.00984	**0.0072** ± 0.00366
	CFO_{100}	0.101 ± 0.0643	0.0246 ± 0.0132	0.00783 ± 0.00441
	FEW	0.217 ± 0.391	0.0413 ± 0.0252	0.0171 ± 0.00914
	AUG_1	0.103 ± 0.0815	0.0416 ± 0.0245	0.0251 ± 0.0116
	AUG_{10}	0.109 ± 0.0552	0.042 ± 0.0253	0.0249 ± 0.013
	AUG_{100}	0.0987 ± 0.0549	0.0365 ± 0.022	0.0245 ± 0.0131
	AUG_{1000}	0.108 ± 0.0697	0.036 ± 0.0208	0.0195 ± 0.00992
	AUG_{10000}	0.148 ± 0.345	0.026 ± 0.0168	0.00878 ± 0.00496
	AUG_{100000}	0.179 ± 0.229	0.0324 ± 0.0279	0.00978 ± 0.00621
Pubmed	Base	2.43 ± 1.36	0.64 ± 1.09	0.185 ± 0.0859
	GFO	**1.52** ± 1.56	0.327 ± 0.72	0.196 ± 0.0851
	CFO_{10}	2.61 ± 1.27	**0.174** ± 0.0407	0.203 ± 0.0694
	CFO_{100}	2.06 ± 1.51	0.182 ± 0.0322	0.168 ± 0.0906
	FEW	2.31 ± 1.38	0.328 ± 0.697	0.196 ± 0.0686
	AUG_1	1.66 ± 1.51	0.507 ± 0.865	**0.165** ± 0.0907
	AUG_{10}	2.78 ± 1.13	0.735 ± 1.19	0.225 ± 0.0776
	AUG_{100}	2.48 ± 1.36	0.331 ± 0.69	0.19 ± 0.0914
	AUG_{1000}	2.22 ± 1.44	0.323 ± 0.498	0.217 ± 0.0705
	AUG_{10000}	2.5 ± 1.35	0.485 ± 0.924	0.207 ± 0.0744
	AUG_{100000}	2.27 ± 1.37	0.672 ± 1.04	0.195 ± 0.0712

Table 6. Fairness in Link Recommendations: DP@k metrics for the augmented loss model on all datasets. Prior results for the GFO, CFO, and FEW methods are repeated for ease of comparison. The highest performing model for each dataset and metric is bolded.

B ADDITIONAL RESULTS FOR CFO

We present an additional experiment to test the impact of the number of introduced nodes c in the CFO method, as discussed in section 5.5. Experiments were run over 300 epochs using 5-fold cross validation. As c increases, the achieved link divergence and DP@40 metrics decay exponentially while the reconstruction loss trends upward slightly. There is no consistent change in the average AUROC score with increased c.

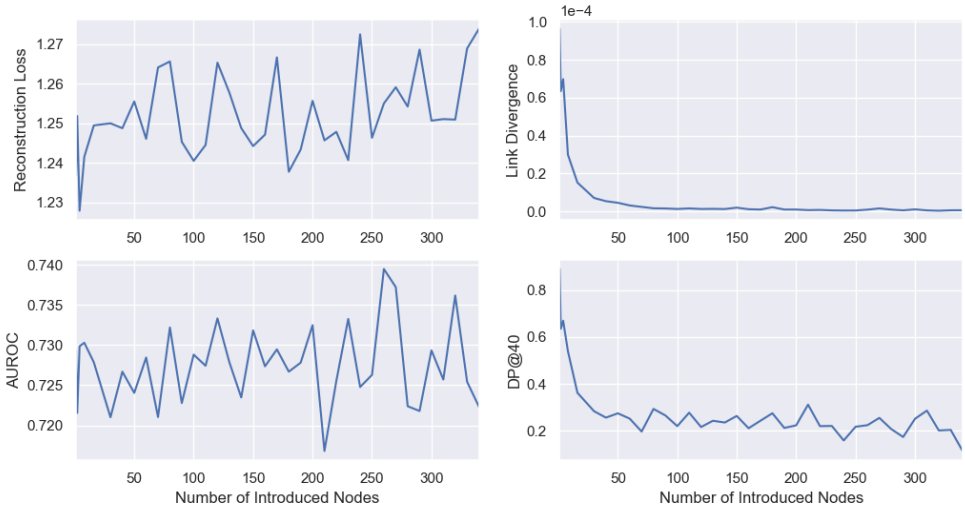

Fig. 3. Top left: Reconstruction loss of the CFO_c model for varying values of c. Top right: Link divergence of the CFO_c model for varying values of c. Bottom left: AUROC metric of the CFO_c model for varying values of c. Bottom right: DP@40 metric of the CFO_c model for varying values of c. All models were trained for 300 epochs with 5-fold cross validation on the Cora dataset. Values of c taken into consideration were 1, 2, 4, 8, 16, 30, 40, ..., 340.