Individual Fairness for Graph Neural Networks: A Ranking based Approach

Yushun Dong
University of Virginia
yd6eb@virginia.edu

Jian Kang
University of Illinois at Urbana-Champaign
jiank2@illinois.edu

Hanghang Tong
University of Illinois at Urbana-Champaign
htong@illinois.edu

Jundong Li
University of Virginia
jundong@virginia.edu

ABSTRACT

Recent years have witnessed the pivotal role of Graph Neural Networks (GNNs) in various high-stake decision-making scenarios due to their superior learning capability. Close on the heels of the successful adoption of GNNs in different application domains has been the increasing societal concern that conventional GNNs often do not have fairness considerations. Although some research progress has been made to improve the fairness of GNNs, these works mainly focus on the notion of group fairness regarding different subgroups defined by a protected attribute such as gender, age, and race. Beyond that, it is also essential to study the GNN fairness at a much finer granularity (i.e., at the node level) to ensure that GNNs render similar prediction results for similar individuals to achieve the notion of individual fairness. Toward this goal, in this paper, we make an initial investigation to enhance the individual fairness of GNNs and propose a novel ranking based framework—REDRESS. Specifically, we refine the notion of individual fairness from a ranking perspective, and formulate the ranking based individual fairness promotion problem. This naturally addresses the issue of Lipschitz constant specification and distance calibration resulted from the Lipschitz condition in the conventional individual fairness definition. Our proposed framework REDRESS encapsulates the GNN model utility maximization and the ranking-based individual fairness promotion in a joint framework to enable end-to-end training. It is noteworthy mentioning that REDRESS is a plug-and-play framework and can be easily generalized to any prevalent GNN architectures. Extensive experiments on multiple real-world graphs demonstrate the superiority of REDRESS in achieving a good balance between model utility maximization and individual fairness promotion. Our open source code can be found here: https://github.com/yushundong/REDRESS.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Applied computing** → **Law, social and behavioral sciences**.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467266

KEYWORDS

graph neural networks, individual fairness, ranking

ACM Reference Format:
Yushun Dong, Jian Kang, Hanghang Tong, and Jundong Li. 2021. Individual Fairness for Graph Neural Networks: A Ranking based Approach. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467266

1 INTRODUCTION

Graph structured data is ubiquitous in today's increasingly connected world. Examples include social networks, biological networks, knowledge graphs, and critical infrastructure systems, to name a few. To gain deep insights from these graphs, a plethora of sophisticated graph mining algorithms have been developed in the past few decades [1, 11, 30, 37]. Among these efforts, Graph Neural Networks (GNNs) have emerged as a promising learning paradigm recently and demonstrated superior learning performance in diverse settings [12, 19, 35], which makes GNNs play an increasingly important role in various high-stake decision-making scenarios, e.g., credit scoring [32], recommendation [9], and medical diagnosis [9]. Nevertheless, close on the heels of the successful adoption of GNNs in various real-world scenarios has been the increasing societal concerns that these algorithms often do not have the fairness consideration, resulting in discriminatory actions toward specific groups or populations [2, 7, 14, 24]. For example, there is a growing practice of credit scoring using social network information [38], in which graph neural networks have become a de facto solution [13, 36]. Although these practices have shown to broaden opportunities for a larger portion of the population, they still yield unfair decisions for people in certain protected groups (e.g., low-income people) [7, 24].

To date, a wide spectrum of fairness measures has been developed to quantify and mitigate the bias of underlying learning algorithms [8, 22, 29, 41]. Existing fairness measures can be mainly divided into *group fairness measures* and *individual fairness measures* [24]. On the one hand, group fairness ensures that members of different protected groups (e.g., gender, race, and income) bear similar outcome statistics regarding model predictions [10, 14, 14, 40]. On the other hand, individual fairness scrutinizes potential bias and discrimination at a much finer granularity, and ensures similar individuals should yield similar prediction outcomes [8, 41]. Although much progress has been made in the field of algorithmic fairness, the studies of fairness issues in graph mining algorithms are fairly

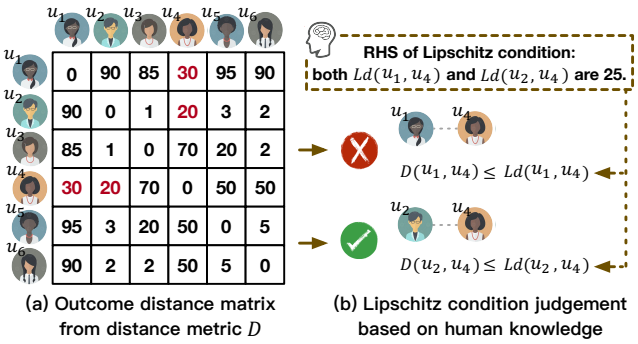


Figure 1: (a) Outcome distance for all instance pairs. (b) Lipschitz condition sanity check for pairs (u_1, u_4) and (u_2, u_4).

recent. Specifically, in the context of graph representation learning, a vast majority of existing works focus on the notion of group fairness, aiming to learn node embeddings that are independent of any protected attributes [3, 4, 29]. However, as graph data is naturally heterogeneous, different data modalities (i.e., graph structure and node attributes) are often coupled together. Thus bias and discrimination can exist in various shapes and formats. In this regard, beyond the notion of group fairness regarding protected attributes, it is also desired to dig into the atomic components of graphs (i.e., a node) to ensure that graph representation learning renders similar results for similar individuals—in achieving *individual fairness*.

Despite the fundamental importance of achieving individual fairness for graph mining algorithms, the related studies are still in their infancy. In this work, we make an initial investigation to enhance the individual fairness of graph neural networks for decision-making. However, it remains a daunting problem mainly because of the following challenges: (1) ***Constraint Formulation.*** Formulating proper constraints to improve individual fairness is non-trivial. Traditionally, given a pair of instances, such constraint is achieved via the Lipschitz condition[1] [8, 17]. Nevertheless, the Lipschitz constant is often hard to be specified due to distance metric difference between the input and outcome space. (2) ***Distance Calibration.*** The absolute distance comparison in the Lipschitz condition fails to calibrate the differences between different instances. For example in Fig. 1, on the one hand for instance u_1, instance u_4 is the closest one to it in the outcome space. However, the distance between them violates the Lipschitz condition and thus we do not impose the individual fairness constraint between them (although we should since u_1 is much closer to u_4 than other instances). On the other hand for u_2, although u_4 is the second farthest one to it, Lipschitz condition is still satisfied and individual fairness constraint is imposed (in fact we may not need to do that since u_2 is much further to u_4 than u_2, u_5, and u_6). (3) ***End-to-End Learning Paradigm.*** A major advantage of GNNs over traditional unsupervised graph embedding algorithms is their end-to-end learning mechanism, i.e., the node embeddings are tailored for specific downstream learning tasks. In this regard, how to incorporate the individual fairness constraint seamlessly into the learning process without jeopardizing its end-to-end paradigm is another challenge that needs to be tackled.

[1]Given a pair of instance x and y, their distance in the outcome space is upper bounded by their distance in the input space such that $D(f(x), f(y)) \leq Ld(x, y)$, where $f(.)$ maps instances to the output space, and L is the Lipschitz constant. $D(.,.)$ and $d(.,.)$ are two functions that measure the distance of instances in the output space and input space, respectively.

Table 1: Symbols and descriptions.

Symbols	Definitions or Descriptions
$\mathcal{M}$	backbone GNN model
$\mathcal{G}$	input graph
$\mathbf{A}$	adjacency matrix of graph $\mathcal{G}$
$\mathbf{X}$	node feature matrix of graph $\mathcal{G}$
$\mathbf{Y}$	ground truth of downstream learning task
$\hat{\mathbf{Y}}$	prediction of downstream learning task
$\mathbf{S}_{\mathcal{G}}$	oracle pairwise similarity matrix
$\mathbf{S}_{\hat{\mathbf{Y}}}$	pairwise similarity matrix from $\hat{\mathbf{Y}}$
n	number of nodes
d	number of node features
l	layer number in the backbone GNN model

In this paper, to tackle these challenges, we propose a principled framework REDRESS (short for Ranking basEd inDividual faiRnESS) to promote the individual fairness of graph neural networks. Specifically, to tackle the first two challenges, we refine the definition of individual fairness from a ranking perspective, and formulate the individual fairness constraint as "*for each instance u_i, the two ranking lists of other instances (based on their distances to u_i) in the input space and outcome space should be as similar as possible*". As such, we can avoid the delicate distance comparison between two different distance metrics in the Lipschitz condition, and the relative ranking comparison can also naturally alleviate the issue of uncalibrated distance. To tackle the third challenge, two optimization modules are encapsulated in REDRESS to improve model utility and individual fairness, respectively. To fit into the end-to-end training process, the two optimization modules are designed to adapt to the gradient-based optimization techniques. The main contributions of this paper can be summarized as follows.

- ***Problem Formulation.*** We study a novel problem of promoting individual fairness for graph neural networks from a ranking perspective.
- ***Algorithmic Design.*** We address the limitations of existing individual fairness constraints and propose a novel plug-and-play framework to mitigate the individual biases without jeopardizing the utility of underlying graph neural networks.
- ***Experimental Evaluations.*** We perform comprehensive experimental evaluations on real-world datasets to demonstrate the superiority of our proposed framework in terms of both bias mitigation and model utility maximization.

2 PROBLEM STATEMENT

In this section, we firstly present the notations used throughout this paper. Then we introduce the definition of *individual fairness from a ranking perspective*, followed by the problem formulation of *promoting ranking based individual fairness of GNNs*.

Notations. We use bold uppercase letters (e.g., $\mathbf{S}$), bold lowercase letters (e.g., $\mathbf{s}$), and normal lowercase letters (e.g., s) to denote matrices, vectors and scalars, respectively. Also, for any matrix, e.g., $\mathbf{S}$, we represent its i-th row as $\mathbf{s}_i$, its (i,j)-th entry as $\mathbf{S}_{ij}$ or s_{ij}, and its transpose as $\mathbf{S}^\top$. For any scalar, $|\cdot|$ is the absolute value operator.

Let $\mathcal{G} = (\mathbf{A}, \mathbf{X})$ be an input graph, where $\mathbf{A} \in \mathbb{R}^{n \times n}$ denotes the adjacency matrix of the graph and $\mathbf{X} \in \mathbb{R}^{n \times d}$ denotes the matrix for node features (n nodes and d node features). $\mathbf{Y}$ and $\hat{\mathbf{Y}}$ represent the ground truth and predictions for a specific downstream task, respectively. For example, if the downstream task is *node classification*, $\mathbf{Y} \in \{0, 1\}^{n \times c}$ and $\hat{\mathbf{Y}} \in \mathbb{R}^{n \times c}$ are the ground truth and predicted class membership matrix (c classes), respectively.

To tackle the aforementioned challenges of *Constraint Formulation* and *Distance Calibration*, we refine the definition of individual fairness from a ranking perspective (Definition 1)[2]. We follow similar settings in [17, 21, 22], where the oracle pairwise similarity matrix $\mathbf{S}_{\mathcal{G}}$ is given apriori (e.g., assigned by specialists in [22]).

Definition 1. ***Individual fairness from a ranking perspective.*** *Given the oracle pairwise similarity matrix $\mathbf{S}_{\mathcal{G}}$ of the input graph $\mathcal{G}$, and the similarity matrix $\mathbf{S}_{\hat{\mathbf{Y}}}$ among instances in the outcome space (defined upon a similarity metric), we say the predictions are individually fair if for each instance i, the two ranking lists that encode the relative order of other instances (ranked based on the similarity between instance i and other instances in descending order) from $\mathbf{S}_{\mathcal{G}}$ and $\mathbf{S}_{\hat{\mathbf{Y}}}$ are consistent with each other.*

Example: Given a graph $\mathcal{G}$ with five nodes, suppose the ranking list that encodes the similarity between node u_1 and other nodes from $\mathbf{S}_{\mathcal{G}}$ is $\{u_4, u_3, u_2, u_5\}$, we say the predictions are are individually fair for node u_1 if the ranking list that encodes the similarity between u_1 and other nodes from $\mathbf{S}_{\hat{\mathbf{Y}}}$ is also $\{u_4, u_3, u_2, u_5\}$.

Based on Definition 1, we formulate the problem of *promoting ranking based individual fairness of GNNs* as follows.

Problem 1. ***Promoting ranking based individual fairness of GNNs.*** *Given an input graph $\mathcal{G}$, a backbone GNN model $\mathcal{M}$ (e.g., GCN [19]), the ground truth $\mathbf{Y}$ and the predictions $\hat{\mathbf{Y}}$ corresponding to a specific downstream task (e.g., node classification), oracle pairwise similarity matrix $\mathbf{S}_{\mathcal{G}}$ from $\mathcal{G}$, and pairwise similarity matrix $\mathbf{S}_{\hat{\mathbf{Y}}}$ obtained from $\hat{\mathbf{Y}}$, our goal is to promote the individual fairness of each node in the graph $\mathcal{G}$ according to Definition 1 without jeopardizing the utility of the model predictions (i.e., making $\hat{\mathbf{Y}}$ close to $\mathbf{Y}$).*

3 THE PROPOSED FRAMEWORK—REDRESS

In this section, we firstly introduce the overall structure of the proposed framework REDRESS. Then details of utility maximization and individual fairness promotion are presented, followed by the overall objective function formulation for training.

3.1 Overall Framework Structure

As the main focus of this work is to promote the individual fairness of GNNs during the decision-making process while maximally preserve the utility of the underlying learning models, we formulate these two desiderata as two separate modules and encapsulate them together with the GNN backbone into an end-to-end learning framework—REDRESS. The overall architecture of REDRESS is shown in Fig. 2. Essentially, it consists of three key parts: GNN backbone model, utility maximization (Module 1), and individual fairness promotion (Module 2).

[2]For the ease of presentation convenience, we use similarity measures instead of distance metrics. Similarity measure can be considered as an inverse distance metric.

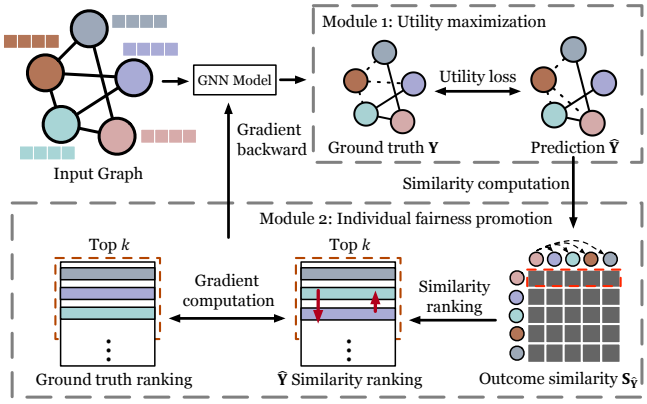


Figure 2: An illustration of the proposed REDRESS structure. Module 1 and 2 is utilized for model utility maximization and individual fairness promotion, respectively.

- **GNN backbone model.** It provides a basic GNN architecture for downstream learning tasks. Some prevalent choices include GCN [19], GAE [20], and SGC [39].
- **Utility maximization.** To maximize the utility of the backbone model for a specific learning task, this module aims to minimize the prediction loss of the corresponding task.
- **Individual fairness optimization.** To relieve the individual bias toward fair decision-making, this module enforces the similarity ranking lists from $\mathbf{S}_{\hat{\mathbf{Y}}}$ and $\mathbf{S}_{\mathcal{G}}$ of each instance to be consistent according to Definition 1.

3.2 GNN Backbone Model

Acting as the backbone of the proposed framework, the GNN model takes the input $\mathcal{G}$ and outputs $\hat{\mathbf{Y}}$ as the predictions for a specific downstream learning task. The basic operation of GNN between l-th and $(l+1)$-th layer can be summarized as

$$\mathbf{h}_v^{(l+1)} = \sigma(\text{COMBINE}(\mathbf{h}_v^{(l)}, f(\{\mathbf{h}_u^{(l)} : u \in \mathcal{N}(v)\}))), \quad (1)$$

where $\mathbf{h}_v^{(l)}$ and $\mathbf{h}_v^{(l+1)}$ represent the embedding of node v at l-th and $(l+1)$-th layer, respectively. For a graph with node feature matrix $\mathbf{X}$, $\mathbf{h}_v^{(0)}$ can be initialized as the input node feature $\mathbf{x}_v$. $\mathcal{N}(v)$ indicates the neighbor set of node v according to the adjacency matrix $\mathbf{A}$. $f(.)$ denotes the aggregating function, e.g., weighted sum. COMBINE(.) indicates the combining function for output of $f(.)$ and $\mathbf{h}_v^{(l)}$, which combines the representation from the centering node and the representations of its neighbors. $\sigma(.)$ represents the activation function (e.g., ReLU). Denote the output of the last GNN layer as matrix $\mathbf{Z} \in \mathbb{R}^{n \times c}$, then the predictions $\hat{\mathbf{Y}}$ of GNN can then be obtained as $\text{softmax}(\mathbf{Z}) \in \mathbb{R}^{n \times c}$ for *node classification* [19] and $\text{sigmoid}(\mathbf{Z}^\top \mathbf{Z}) \in \mathbb{R}^{n \times n}$ for *link prediction* [20].

3.3 Model Utility Maximization

To maximize the utility of the backbone GNN model in advancing downstream learning tasks, we need to enforce the predictions $\hat{\mathbf{Y}}$ to be closer to the ground truth $\mathbf{Y}$. To this end, a loss function corresponding to the specific learning task should be defined in Module 1 between $\mathbf{Y}$ and $\hat{\mathbf{Y}}$. For example, for the *node classification* and the *link prediction* tasks, the corresponding cross-entropy loss

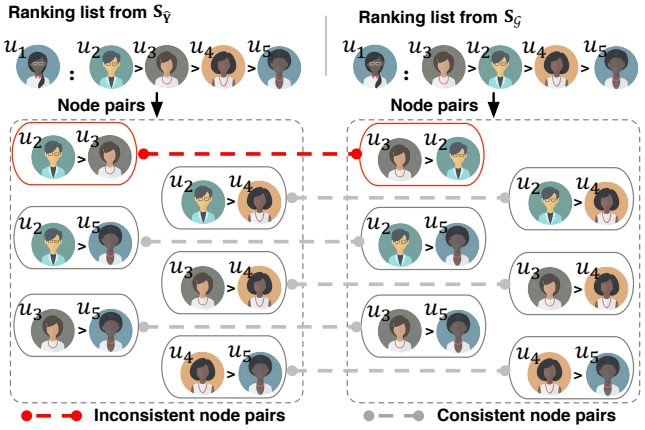


Figure 3: An illustrative example of the relative ranking order comparison for our proposed framework.

can be used to quantify the utility of the GNN model

$$\mathcal{L}_{\text{utility}} = -\sum_{(i,j)\in\mathcal{T}} \mathrm{Y}_{ij} \ln \hat{\mathrm{Y}}_{ij}. \tag{2}$$

Here $\mathcal{T}$ represents the (node, class) tuple set for training nodes in the *node classification* task and (node, node) tuple set for the vertices of training edges in the *link prediction* task. The utility maximization can be achieved by minimizing the cross-entropy loss in Equ. (2). It should be noted that for this module, gradient-based optimization techniques can be directly applied for end-to-end training as the loss function is differentiable w.r.t. the model parameters.

3.4 Individual Fairness Promotion

As mentioned above, this module aims to promote the ranking based individual fairness for GNN such that for each node, the two ranking lists obtained from the oracle similarity matrix $\mathrm{S}_{\mathcal{G}}$ and the outcome similarity matrix $\mathrm{S}_{\hat{\mathrm{Y}}}$ are consistent with each other. Since the ranking list from the oracle similarity matrix $\mathrm{S}_{\mathcal{G}}$ is fixed, and $\mathrm{S}_{\hat{\mathrm{Y}}}$ is derived from the prediction outcome $\hat{\mathrm{Y}}$ via certain similarity metric, the ranking list from $\mathrm{S}_{\hat{\mathrm{Y}}}$ should be optimized via learning more appropriate $\hat{\mathrm{Y}}$—we refer it as *ranking optimization*. One straightforward solution to achieve this goal is to derive two ranking lists from $\mathrm{S}_{\mathcal{G}}$ and $\mathrm{S}_{\hat{\mathrm{Y}}}$ for each node, then define a loss function to quantify the difference between these two ranking lists. After that, we can combine the loss function over all nodes together and minimize the overall loss for a better $\hat{\mathrm{Y}}$ that can promote individual fairness. However, such a straightforward solution is often impractical as the ranking operations for the ranking lists will make the overall loss function not differentiable (w.r.t. the GNN model parameters) anymore, in a way the prevalent gradient-based optimization techniques cannot be directly applied. In other words, there is a gap between the involvement of ranking operations pertaining to Definition 1 and the need for gradient-based optimization techniques. To bridge the gap, we propose a novel ranking optimization strategy and we will elaborate more details in the following part.

3.4.1 Ranking Optimization. As mentioned above, minimizing the difference between two ranking lists (for each node) from $\mathrm{S}_{\hat{\mathrm{Y}}}$ and $\mathrm{S}_{\mathcal{G}}$ with gradient-based optimization techniques is difficult because of the non-differentiable ranking operations. Instead of formulating the loss based on ranking lists, here we propose a new loss formulation directly upon the outcome similarity matrix $\mathrm{S}_{\hat{\mathrm{Y}}}$ and oracle similarity matrix $\mathrm{S}_{\mathcal{G}}$. Since the new loss formulation does not rely on the ranking lists, gradient-based optimization techniques can then be applied. The new loss formulation is based on a probabilistic approach inspired by [28]. For each node, the new loss formulation will enforce the relative order of each node pair decided by $\mathrm{S}_{\hat{\mathrm{Y}}}$ and that decided by S_{G} to be consistent. More specifically, for each node u_i, if it is with higher similarity value to u_j than u_m in $\mathrm{S}_{\hat{\mathrm{Y}}}$ (i.e., $\hat{s}_{i,j} > \hat{s}_{i,m}$, $i \neq j \neq m$, where $\hat{s}_{i,j}$ denotes the (i,j)-th entry of $\mathrm{S}_{\hat{\mathrm{Y}}}$), then it should also be with higher similarity value to u_j than u_m in the oracle similarity matrix $\mathrm{S}_{\mathcal{G}}$ (i.e., $s_{i,j} > s_{i,m}$, $i \neq j \neq m$, where $s_{i,j}$ denotes the (i,j)-th entry of $\mathrm{S}_{\mathcal{G}}$). In other words, the loss aims to penalize the node pairs whose relative similarity order are not consistent across the predicted similarity matrix $\mathrm{S}_{\hat{\mathrm{Y}}}$ and the oracle similarity matrix $\mathrm{S}_{\mathcal{G}}$ (as shown in Fig. 3). We then introduce the details of the loss formulation below.

To illustrate the formulation of the loss function, we take the loss computation of the node pair (u_j, u_m) centered on node u_i as an example. For node u_i, we define $\hat{P}_{j,m}(\hat{s}_{i,j}, \hat{s}_{i,m})$ as the predicted probability that the node u_i is more similar to node u_j than to node u_m. Here $\hat{s}_{i,j}$ and $\hat{s}_{i,m}$ represent the similarity score between node pairs (u_i, u_j) and (u_i, u_m) from the outcome similarity matrix $\mathrm{S}_{\hat{\mathrm{Y}}}$, respectively. To formulate it as a probability score between 0 and 1, we make use of the sigmoid function:

$$\hat{P}_{j,m}(\hat{s}_{i,j}, \hat{s}_{i,m}) = \frac{1}{1 + e^{-\alpha(\hat{s}_{i,j} - \hat{s}_{i,m})}}, \tag{3}$$

where α here is a scalar. Accordingly, define the known probability that the node u_i is more similar to node u_j than to node u_m as $P_{j,m}(s_{i,j}, s_{i,m})$, which can be formulated as follows:

$$P_{j,m}(s_{i,j}, s_{i,m}) = \begin{cases} 1, & s_{i,j} > s_{i,m}, \\ 0.5, & s_{i,j} = s_{i,m}, \\ 0, & s_{i,j} < s_{i,m}. \end{cases} \tag{4}$$

Here $s_{i,j}$ and $s_{i,m}$ denote the similarity between node pairs (u_i, u_j) and (u_i, u_m) from the oracle similarity matrix $\mathrm{S}_{\mathcal{G}}$, respectively. To promote individual fairness via ranking optimization, it is necessary to quantify and minimize the difference between the predicted probability distribution and the known one. Here, we make use of cross-entropy loss for the difference quantification between the two distributions. For example, the cross-entropy loss of node pair (u_j, u_m) centered on u_i can be expressed as

$$\mathcal{L}_{j,m}(i) = -P_{j,m} \log \hat{P}_{j,m} - (1 - P_{j,m}) \log(1 - \hat{P}_{j,m}). \tag{5}$$

Then, the total loss function over all node pairs centered on node u_i can be formulated as

$$\mathcal{L}_{\text{fairness}}(i) = \sum_{j,m} \mathcal{L}_{j,m}(i), \tag{6}$$

where $i \neq j \neq m$. By minimizing the above loss function, the relative order of all node pairs (centered on node u_i) decided by $\mathrm{S}_{\hat{\mathrm{Y}}}$ will be enforced to be consistent with the corresponding order decided by $\mathrm{S}_{\mathcal{G}}$. When the loss of all nodes is aggregated and minimized, the ranking based individual fairness can be achieved.

3.4.2 Training Facilitation. Minimizing Equ. (6) for node u_i requires the ranking of all other nodes centered on node u_i being optimal (i.e., the ranking follows a descending order according to the similarity score from $\mathbf{S}_{\mathcal{G}}$). This is usually hard to achieve, especially for graphs with a large number of nodes. Consequently, for each node u_i, we propose to focus on the ranking optimization of the top-k nodes given by the outcome similarity matrix $\mathbf{S}_{\hat{\mathbf{Y}}}$. This strategy is motivated by the basic principle of individual fairness [8], which is only emphasizing the outcome of "similar people" to be similar. Motivated by existing research on learning to rank [28], we achieve this goal by developing a simple but effective approach. Specifically, we define $z_{@k}(\cdot,\cdot)$ as the similarity metric (e.g., *NDCG@k* [16] or *ERR@k* [5]) between the top-k ranking list derived from $\mathbf{S}_{\mathcal{G}}$ and the predicted top-k ranking list derived from $\mathbf{S}_{\hat{\mathbf{Y}}}$ for each node. For the loss of each node pair given by Equ. (5), we scale the loss term by the absolute value change of $z_{@k}$ if the ranking positions of the corresponding node pair u_j and u_m in the predicted ranking list are swapped. Then the loss for each node u_i can be presented as

$$\mathcal{L}_{\text{fairness}}(i) = \sum_{j,m} \mathcal{L}_{j,m}(i)|\Delta z_{@k}|_{j,m}\,, \tag{7}$$

where $i \neq j \neq m$. To illustrate the computation of $|\Delta z_{@k}|_{j,m}$, we take $k = 4, i = 1, j = 4$ and $m = 2$ as an example. Assume the top-4 ranking of node u_1 with other nodes in $\mathbf{S}_{\hat{\mathbf{Y}}}$ is $\hat{List}_{u_i} = \{u_4, u_3, u_2, u_5\}$, and the corresponding ranking in $\mathbf{S}_{\mathcal{G}}$ is $List_{u_i} = \{u_4, u_3, u_5, u_2\}$. Then the $|\Delta z_{@k}|_{4,2}$ corresponding to node pair (u_4, u_2) centered on node u_1 is $|z_{@k}(List_{u_1}, \hat{List}_{u_1}) - z_{@k}(List_{u_1}, \hat{List}'_{u_1})|$. Here $\hat{List}'_{u_1} = \{u_2, u_3, u_4, u_5\}$, i.e., $\hat{List}_{u_1}$ with the positions of node u_2 and u_4 swapped. With such method, the ranking optimization will be enforced to focus more on the top-k nodes for each node u_i. Here, k is often specified as a very number ($k \ll n$).

Besides, $|\Delta z_{@k}|_{j,m}$ is always zero if neither node u_j nor u_m is from the top-k nodes of u_i according to $\mathbf{S}_{\hat{\mathbf{Y}}}$. Consequently, this strategy also reduces the time complexity from $O(n\cdot\binom{n-1}{2}) = O(n^3)$ to $O(n\cdot\binom{n-1}{1}\binom{k}{1}) = O(n^2k)$ for the total loss computation. Nevertheless, the time complexity of $O(n^2k)$ is still expensive for the training on large graphs. To further reduce the time complexity, here we constrain that both nodes in the pair (u_j, u_m) are from the top-k ranked nodes of u_i according to $\mathbf{S}_{\hat{\mathbf{Y}}}$. Then the total fairness loss can be formulated as

$$\mathcal{L}_{\text{fairness}} = \sum_{i} \sum_{j,m:j,m\in\mathcal{K}(i)} \mathcal{L}_{j,m}(i)|\Delta z_{@k}|_{j,m}\,, \tag{8}$$

where $i \neq j \neq m$, and $\mathcal{K}(i)$ represents the top-k ranked node set for node u_i. In this way, the computational complexity can be further reduced from $O(n^2k)$ to $O(n\cdot\binom{k}{2}) = O(nk^2)$, which facilitates the training process of REDRESS.

3.5 Overall Objective Function

Now we have $\mathcal{L}_{\text{utility}}$ for the model utility maximization (formulated in Section 3.3), and $\mathcal{L}_{\text{fairness}}$ for the model individual fairness promotion (formulated in Section 3.4). Then the overall objective function of the proposed framework REDRESS can be attained by combining the two formulations together:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{utility}} + \gamma\mathcal{L}_{\text{fairness}}. \tag{9}$$

Here γ is a tunable hyperparameter controlling the strength of individual fairness constraint. For training of the proposed framework, the gradient-based techniques can be directly adopted to minimize the total objective function $\mathcal{L}_{\text{total}}$.

4 EXPERIMENTAL EVALUATIONS

In this section, we first introduce the adopted downstream learning tasks and the used datasets. Then we present the experimental settings and the implementation details. At last, we show the empirical evaluation results of REDRESS.

4.1 Downstream Tasks and Datasets

Downstream Tasks. To assess the performance of our proposed framework REDRESS, we choose the widely adopted *semi-supervised node classification* task [19, 36, 39] and *link prediction* task [20, 26] as the downstream learning tasks. Both of these two tasks are of high practical significance in many areas.

Datasets. To comprehensively explore how REDRESS promotes the individual fairness of GNNs from a ranking perspective, we adopt three different real-world datasets for each of the chosen downstream task. Specifically, for the *semi-supervised node classification* task, we adopt one citation network (ACM [33]) and two co-authorship networks (Co-author-CS and Co-author-Phy [31] from the KDD Cup 2016 challenge). For the *link prediction* task, three social networks (BlogCatalog [34], Flickr [15], and Facebook [23]) are used for evaluation. All of these datasets are publicly available. For citation networks, each node represents a paper, and an edge between two nodes denotes the citation relationship between two papers. For co-author networks, nodes represent authors, and an edge between two nodes indicates that the linked two authors have co-authored a paper together. Node attributes of both citation and co-author networks are generated by the bag-of-words model based on the abstract of the published paper. For social networks, each node represents a user, and links represent the corresponding interactions between users. The attributes here are constructed by the profiles or descriptions of users. Here, CS and Phy are short for the datasets Co-author-CS and Co-author-Phy, respectively. The detailed statistics of these datasets are shown in Table 2.

4.2 Experiment Settings

GNN Backbone Models. As mentioned previously, our proposed REDRESS is a plug-and-play framework which can be easily generalized to any prevalent GNN architectures. Hence, we choose two different backbone GNN architectures for each downstream learning task in our experiments. For the *semi-supervised node classification* task, Graph Convolutional Network (GCN) [19] and Simplifying Graph Convolutional Network (SGC) [39] are adopted as our backbones. For the *link prediction* task, GCN and Variational Graph Auto-Encoders (GAE) [20] are employed.

Oracle Similarity Matrix. Following the settings of [17, 21, 22], the oracle similarity matrix $\mathbf{S}_{\mathcal{G}}$ of the input graph $\mathcal{G}$ is a given apriori. In practice, the oracle similarity matrix is often problem-specific

Table 2: Detailed statistics of the used datasets for node classification (short as NC) and link prediction (short as LP).

	Dataset	# Nodes	# Edges	# Features	# Classes
NC	ACM	16,484	71,980	8,337	9
	CS	18,333	81,894	6,805	15
	Phy	34,493	247,962	8,415	5
LP	BlogCatalog	5,196	171,743	8,189	N/A
	Flickr	7,575	239,738	12,047	N/A
	Facebook	4,039	88,234	1,406	N/A

and is determined by humans. To show the generalization capability of REDRESS in handling different types of oracle similarity matrix, we construct two different types of oracle similarity matrix from the feature perspective and the structural perspective. From the feature perspective, we compute the cosine similarity between input node features as the $S_{\mathcal{G}}$; while from the structural perspective, we compute the Jaccard similarity between node pairs as the $S_{\mathcal{G}}$. For the outcome similarity matrix $S_{\hat{Y}}$, we utilize the cosine distance, which is the most widely adopted distance metric to measure node pair similarity in the embedding space.

Baselines. To demonstrate the superiority of our proposed ranking-based individual fairness framework, we compare REDRESS with the following individual fairness promotion approaches on top of the backbone GNN models. It should be noted that the existing group fairness graph embedding methods (such as [3, 29]) cannot be used for comparison as they achieve fairness for subgroups determined by specific protected attributes while we focus on the notion of individual fairness without such protected attributes.

- **PFR** [22]: PFR aims to learn fair representations to achieve the notion of individual fairness. It has demonstrated to outperform traditional approaches such as [14, 21, 41] on individual fairness promotion. Since PFR can be considered as a pre-processing strategy and is not tailored for graph data, we employ it on the input node features to generate a new fair node feature representation and feed it into the backbone GNN models for learning.
- **InFoRM** [17]: InFoRM is a recently proposed individual fairness framework for conventional graph mining tasks (e.g., PageRank, Spectral Clustering) based on the Lipschitz condition. Here, we adapt InFoRM to different backbone GNN models by combing its individual fairness promotion loss and the unity loss of the backbone GNN model together, and then optimize the final loss in an end-to-end manner.

Evaluation Metrics. For model utility evaluation, we adopt classification accuracy (*ACC*) and area under receiver operating characteristic curve (*AUC*) for the node classification task and the link prediction task, respectively. For individual fairness evaluation, we adopt two widely used ranking metrics *NDCG@k* [16] and *ERR@k* [5] to measure the similarity between the ranking list from $S_{\hat{Y}}$ (outcome similarity matrix) and $S_{\mathcal{G}}$ (oracle similarity matrix) for each node. The average value of *NDCG@k* and *ERR@k* across all nodes[3] are reported. $k = 10$ is adopted for quantitative performance comparison, but different choices of k are also studied. The quantitative performance and corresponding discussion based on *ERR@k* is provided in the Appendix.

[3] all nodes in the test set for node classification and all nodes for link prediction

4.3 Implementation Details

REDRESS is implemented in Pytorch [27]. For all GNN backbones adopted in our experiments (i.e., GCN[4], SGC[5] and GAE[6]), we utilize their released implementations. We set the learning rate of all experiments as 0.01 for both the node classification task and the link prediction task. For GCN and SGC based models, the layer and hidden unit number is set as 2 and 16, respectively. For GAE based models, we set the graph convolutional layer number as 3, with the two hidden unit number being 32 and 16. For the training of REDRESS in all experiments, we set γ and k in the loss function as 1 and 10, respectively. All models are optimized with Adam optimizer [18]. For both of the two downstream tasks, datasets are randomly shuffled, and only training data is visible for all models. More details, including dataset split and hyper-parameter settings, are provided in the Appendix.

4.4 Effectiveness of REDRESS

In this section, we perform experiments on real-world networks to validate the effectiveness of the proposed REDRESS framework. In particular, we aim to answer the following research questions:

- **RQ1:** How well can REDRESS balance the GNN model utility and individual fairness compared with other baselines?
- **RQ2:** How will the individual fairness promotion hyperparameter γ affect the performance of REDRESS?
- **RQ3:** How will the choice of parameter k affect the performance of REDRESS?

RQ1: Performance on Balancing Utility and Fairness. First, we investigate the effectiveness of the proposed REDRESS framework by comparing its performance on balancing the model utility and individual fairness against state-of-the-art alternatives. For generalization purpose, the performance of REDRESS is compared with other baselines under different settings of oracle similarity matrices (i.e., feature similarity and structure similarity) and different GNN backbones (i.e., GCN, SGC, and GAE). Quantitative results for the *node classification* task and the *link prediction* task are shown in Table 3 and Table 4, respectively. In these two tables, higher *ACC* and *AUC* represents better performance on model utility, and higher *NDCG*@10 indicates better performance on individual fairness. We can make the following observations from these two tables:

- From the perspective of model utility (i.e., *ACC* in node classification and *AUC* in link prediction), our proposed framework REDRESS provides competitive performance compared with other state-of-the-art baselines. Besides, our proposed framework REDRESS achieves better utility performance compared with the vanilla GNN backbones in some cases. We conjecture that this is partly because the individual fairness promotion term plays the role of regularization to prevent over-fitting of the backbone GNN models.
- From the perspective of ranking based individual fairness, our framework outperforms all baseline methods in all cases with different levels of improvement w.r.t. the fairness evaluation metric *NDCG*@10. This verifies the effectiveness of the individual fairness promotion of our proposed framework

[4] https://github.com/tkipf/pygcn
[5] https://github.com/Tiiiger/SGC
[6] https://github.com/tkipf/gae

Table 3: Node classification results on ACM, Co-author-CS (CS) and Co-author-Phy (Phy) datasets. BB represents the backbone GNN model. Vanilla denotes the vanilla GNN. All values are reported in percentage. The relative improvement of each entry compared with the corresponding backbone performance is denoted in the parentheses. Best performance is marked in bold.

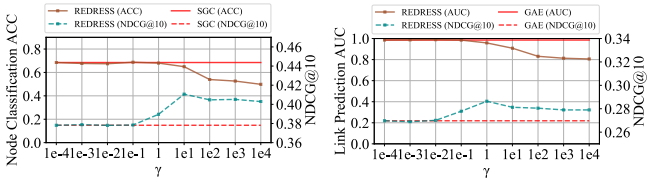
	BB	Model	Feature Similarity		Structural Similarity	
			Utility: ACC	Fairness: NDCG@10	Utility: ACC	Fairness: NDCG@10
ACM	GCN	**Vanilla**	**72.49 ± 0.6** (—)	47.33 ± 1.0 (—)	**72.49 ± 0.6** (—)	25.42 ± 0.6 (—)
		InFoRM	68.03 ± 0.3 (−6.15%)	39.79 ± 0.3 (−15.9%)	69.13 ± 0.5 (−4.64%)	12.02 ± 0.4 (−52.7%)
		PFR	67.88 ± 1.1 (−6.36%)	31.20 ± 0.2 (−34.1%)	69.00 ± 0.7 (−4.81%)	23.85 ± 1.3 (−6.18%)
		REDRESS (Ours)	71.75 ± 0.4 (−1.02%)	**49.13 ± 0.4** (+3.80%)	72.03 ± 0.9 (−0.63%)	**29.09 ± 0.4** (+14.4%)
	SGC	**Vanilla**	68.40 ± 1.0 (—)	55.75 ± 1.1 (—)	**68.40 ± 1.0** (—)	37.18 ± 0.6 (—)
		InFoRM	**68.81 ± 0.5** (+0.60%)	48.25 ± 0.5 (−13.5%)	66.71 ± 0.6 (−2.47%)	28.33 ± 0.6 (−23.8%)
		PFR	67.97 ± 0.7 (−0.62%)	34.71 ± 0.1 (−37.7%)	67.78 ± 0.1 (−0.91%)	37.15 ± 0.6 (−0.08%)
		REDRESS (Ours)	67.16 ± 0.2 (−1.81%)	**58.64 ± 0.4** (+5.18%)	67.77 ± 0.4 (−0.92%)	**38.95 ± 0.1** (+4.76%)
CS	GCN	**Vanilla**	90.59 ± 0.3 (—)	50.84 ± 1.2 (—)	**90.59 ± 0.3** (—)	18.29 ± 0.8 (—)
		InFoRM	88.66 ± 1.1 (−2.13%)	53.38 ± 1.6 (+5.00%)	87.55 ± 0.9 (−3.36%)	19.18 ± 0.9 (+4.87%)
		PFR	87.51 ± 0.7 (−3.40%)	37.12 ± 0.9 (−27.0%)	86.16 ± 0.2 (−4.89%)	11.98 ± 1.3 (−34.5%)
		REDRESS (Ours)	**90.70 ± 0.2** (+0.12%)	**55.01 ± 1.9** (+8.20%)	89.16 ± 0.3 (−1.58%)	**21.28 ± 0.3** (+16.4%)
	SGC	**Vanilla**	87.48 ± 0.8 (—)	74.00 ± 0.1 (—)	87.48 ± 0.8 (—)	32.36 ± 0.3 (—)
		InFoRM	88.07 ± 0.1 (+0.67%)	74.29 ± 0.1 (+0.39%)	88.65 ± 0.4 (+1.34%)	32.37 ± 0.4 (+0.03%)
		PFR	88.31 ± 0.1 (+0.94%)	48.40 ± 0.1 (−34.6%)	84.34 ± 0.3 (−3.59%)	28.87 ± 0.9 (−10.8%)
		REDRESS (Ours)	**90.01 ± 0.2** (+2.89%)	**76.60 ± 0.1** (+3.51%)	**89.35 ± 0.1** (+2.14%)	**34.24 ± 0.2** (+5.81%)
Phy	GCN	**Vanilla**	**94.81 ± 0.2** (—)	34.83 ± 1.1 (—)	**94.81 ± 0.2** (—)	1.57 ± 0.1 (—)
		InFoRM	89.33 ± 0.8 (−5.78%)	31.25 ± 0.0 (−10.3%)	94.46 ± 0.2 (−0.37%)	1.77 ± 0.0 (+12.7%)
		PFR	89.74 ± 0.5 (−5.35%)	24.16 ± 0.4 (−30.6%)	87.26 ± 0.2 (−7.96%)	1.20 ± 0.1 (−23.6%)
		REDRESS (Ours)	94.63 ± 0.7 (−0.19%)	**43.64 ± 0.5** (+25.3%)	93.94 ± 0.3 (−0.92%)	**1.93 ± 0.1** (+22.9%)
	SGC	**Vanilla**	**94.45 ± 0.2** (—)	49.63 ± 0.1 (—)	**94.45 ± 0.2** (—)	3.61 ± 0.1 (—)
		InFoRM	92.01 ± 0.1 (−2.58%)	43.87 ± 0.2 (−11.6%)	94.27 ± 0.3 (−0.19%)	3.64 ± 0.0 (+0.83%)
		PFR	89.74 ± 0.3 (−4.99%)	28.54 ± 0.1 (−42.5%)	89.73 ± 0.3 (−5.00%)	2.62 ± 0.1 (−27.4%)
		REDRESS (Ours)	94.30 ± 0.1 (−0.16%)	**53.40 ± 0.1** (+7.60%)	93.94 ± 0.2 (−0.54%)	**4.03 ± 0.0** (+11.6%)

REDRESS. PFR and InFoRM do not improve *NDCG*@10 in some cases due to the fact that their algorithms are not designed for ranking based individual fairness optimization.

- From the perspective of balancing the model utility and individual fairness, our framework achieves both competitive model utility performance and superior individual fairness promotion in all cases compared with other baselines. Based on such observations, we argue that our framework achieves better performance on balancing the model utility and individual fairness compared with other alternatives.

RQ2: Influence of Individual Fairness Promotion Hyperparameter γ. In our framework, the strength of individual fairness promotion is controlled by hyperparameter γ as defined in Equ. (9). To explore how γ affects the performance of REDRESS, we vary it among {1e-4, 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3, 1e4} and report the performance on utility and individual fairness within fixed training epochs. Due to space limit, here we only present the results from (a) ACM with SGC backbone based on Jaccard similarity, and (b) Facebook with GAE backbone based on cosine similarity as Fig. (4). We can make the following observations (we also have similar observations in other datasets):

- When γ is relatively small (e.g., smaller than 1e-1 for ACM and 1e-2 for Facebook), the individual fairness constraint makes little difference to the performance of REDRESS on the model utility and *NDCG*@10 for both tasks.
- When γ is a modest value (e.g., between 1e-1 and 1e1 for ACM or between 1e-2 and 1 for Facebook), *NDCG*@10 can be improved with little sacrifice on *ACC* or *AUC*. In other words, an appropriate γ helps to achieve better individual fairness performance without jeopardizing the model utility compared with the vanilla SGC and GAE. This shows that REDRESS achieves a proper balance between promoting individual fairness and maintaining the model utility.
- When γ is relatively large (e.g., larger than 1e1 for ACM and 1 for Facebook), *ACC* and *AUC* will be affected by the strength of individual fairness promotion. At the same time, *NDCG*@10 also drops as γ gets larger. This is because when γ falls in this area, the top-10 ranked nodes are far from optimal, and individual fairness promotion module can hardly achieve better performance within fixed epochs.



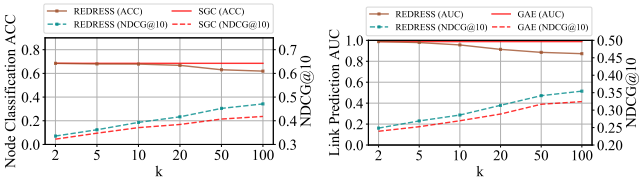
(a) ACM (b) Facebook

Figure 4: Study on individual fairness constraint strength: (a) REDRESS with SGC backbone and Jaccard similarity on ACM; (b) REDRESS with GAE backbone and cosine similarity on Facebook.

RQ3: Influence of the Choice of k. At last, we investigate what the performance of REDRESS will be like under different choices of k. We also present the model utility and individual fairness performance of REDRESS on: (a) ACM with SGC backbone based on Jaccard similarity, and (b) Facebook with GAE backbone based on cosine similarity in Fig. (5). Here we vary k among {2, 5, 10, 20, 50, 100}. Based on the tendencies presented, we can make the following observations (similar observations in other datasets).

Table 4: Link prediction results on BlogCatalog (Blog), Flickr and Facebook (FB) datasets. BB represents the backbone GNN model. Vanilla denotes the vanilla GNN. All values are reported in percentage. The relative improvement of each entry compared with the corresponding backbone performance is denoted in the parentheses. Best performance is marked in bold.

	BB	Model	Feature Similarity		Structural Similarity	
			Utility: AUC	Fairness: NDCG@10	Utility: AUC	Fairness: NDCG@10
Blog	GCN	Vanilla	85.87 ± 0.1 (—)	16.73 ± 0.1 (—)	85.87 ± 0.1 (—)	32.47 ± 0.5 (—)
		InFoRM	79.85 ± 0.6 (−7.01%)	15.57 ± 0.2 (−6.93%)	84.00 ± 0.1 (−2.18%)	26.18 ± 0.3 (−19.4%)
		PFR	84.25 ± 0.2 (−1.89%)	16.37 ± 0.0 (−2.15%)	83.88 ± 0.0 (−2.32%)	29.60 ± 0.4 (−8.84%)
		REDRESS (Ours)	**86.49 ± 0.8** (+0.72%)	**17.66 ± 0.2** (+5.56%)	**86.25 ± 0.3** (+0.44%)	**34.62 ± 0.7** (+6.62%)
	GAE	Vanilla	**85.72 ± 0.1** (—)	17.13 ± 0.1 (—)	85.72 ± 0.1 (—)	41.99 ± 0.4 (—)
		InFoRM	80.01 ± 0.2 (−6.66%)	16.12 ± 0.2 (−5.90%)	82.86 ± 0.0 (−3.34%)	27.29 ± 0.3 (−35.0%)
		PFR	83.83 ± 0.1 (−2.20%)	16.64 ± 0.0 (−2.86%)	83.87 ± 0.1 (−2.16%)	35.91 ± 0.4 (−14.5%)
		REDRESS (Ours)	84.67 ± 0.9 (−1.22%)	**18.19 ± 0.1** (+6.19%)	**86.36 ± 1.5** (+0.75%)	**43.51 ± 0.7** (+3.62%)
Flickr	GCN	Vanilla	**92.20 ± 0.3** (—)	13.10 ± 0.2 (—)	92.20 ± 0.3 (—)	22.35 ± 0.3 (—)
		InFoRM	91.39 ± 0.0 (−0.88%)	11.95 ± 0.1 (−8.78%)	91.73 ± 0.1 (−0.51%)	23.28 ± 0.6 (+4.16%)
		PFR	91.91 ± 0.1 (−0.31%)	12.94 ± 0.0 (−1.22%)	91.86 ± 0.2 (−0.37%)	19.80 ± 0.4 (−11.4%)
		REDRESS (Ours)	91.38 ± 0.1 (−0.89%)	**13.58 ± 0.3** (+3.66%)	**92.67 ± 0.2** (+0.51%)	**28.45 ± 0.5** (+27.3%)
	GAE	Vanilla	89.98 ± 0.1 (—)	12.77 ± 0.0 (—)	89.98 ± 0.1 (—)	23.58 ± 0.2 (—)
		InFoRM	88.76 ± 0.7 (−1.36%)	12.07 ± 0.1 (−5.48%)	**91.51 ± 0.2** (+1.70%)	15.78 ± 0.3 (−33.1%)
		PFR	**90.30 ± 0.1** (+0.36%)	12.12 ± 0.1 (−5.09%)	90.10 ± 0.1 (+1.33%)	20.46 ± 0.3 (−13.2%)
		REDRESS (Ours)	89.45 ± 0.5 (−0.59%)	**14.24 ± 0.1** (+11.5%)	89.52 ± 0.3 (−0.51%)	**29.83 ± 0.2** (+26.5%)
FB	GCN	Vanilla	95.60 ± 1.7 (—)	23.07 ± 0.2 (—)	95.60 ± 1.7 (—)	16.55 ± 1.1 (—)
		InFoRM	90.26 ± 0.1 (−5.59%)	23.23 ± 0.3 (+0.69%)	**96.66 ± 0.6** (+1.11%)	15.18 ± 0.7 (−8.28%)
		PFR	87.11 ± 1.2 (−8.88%)	21.83 ± 0.2 (−5.37%)	94.87 ± 1.9 (−0.76%)	19.53 ± 0.5 (+18.0%)
		REDRESS (Ours)	**96.49 ± 1.6** (+0.93%)	**29.60 ± 0.1** (+28.3%)	92.66 ± 0.4 (−3.08%)	**27.73 ± 1.1** (+67.5%)
	GAE	Vanilla	**98.54 ± 0.0** (—)	26.75 ± 0.1 (—)	**98.54 ± 0.0** (—)	27.03 ± 0.1 (—)
		InFoRM	90.50 ± 0.4 (−8.16%)	22.77 ± 0.2 (−14.9%)	95.03 ± 0.1 (−3.56%)	15.38 ± 0.2 (−43.1%)
		PFR	96.91 ± 0.1 (−1.65%)	23.52 ± 0.1 (−12.1%)	98.28 ± 0.0 (−0.26%)	22.89 ± 0.3 (−15.3%)
		REDRESS (Ours)	95.98 ± 1.5 (−2.60%)	**28.43 ± 0.3** (+6.28%)	94.07 ± 1.7 (−4.54%)	**33.53 ± 0.2** (+24.0%)



(a) ACM (b) Facebook

Figure 5: Study on k choices: (a) REDRESS with SGC backbone and Jaccard similarity on ACM; (b) REDRESS with GAE backbone and cosine similarity on Facebook.

- As k goes larger, REDRESS achieves larger improvement on $NDCG@10$. This proves that larger k brings better optimization effectiveness on individual fairness promotion.
- Model utility performance is barely influenced when k gets larger. This implies REDRESS achieves a proper balance between maintaining the model utility and promoting individual fairness under different choices of k in the optimization. In practice, a modest k (e.g., 20 for ACM and 10 for Facebook) achieves best balancing performance.

5 RELATED WORK

In this section, we briefly present the related works on two aspects: (1) individual fairness; (2) fairness in graphs.

Individual Fairness. Dwork et al. [8] first argue that only emphasizing group fairness regarding protected attributes can barely treat each individual user in a fair manner, and propose the definition of individual fairness: *similar individuals should be treated similarly*. In their work, Lipschitz condition is utilized as the distance constrain for instance pairs between the input and outcome of the decision-making model. Zemel et al. [41] propose to emphasize the balance between the decision-making model utility and individual fairness. Individual fairness is promoted in their work via sharing the mapping function from the model input to corresponding outcome over all individuals. Lahoti et al. [21] point out that most individual fairness works are limited within binary classification problems. They firstly extend the problem setting to multi-class, and improve the model performance on individual fairness via learning low-rank representations for individuals in a model-agnostic way. Another work from Lahoti et al. [22] specifies similar individual pairs by human experts before training, and only emphasize the individual fairness optimization over these pre-assigned pairs. Different from these mentioned works, we define individual fairness from a ranking perspective, and promote individual fairness via ranking-based optimization. To the best of our knowledge, we are the first to define and promote individual fairness from the ranking perspective.

Fairness in Graphs. Graph structured data has become ubiquitous in various high-impact areas. Nevertheless, most previous efforts achieving fairness in graphs focus on group fairness. Basically, group fairness emphasizes that all demographic groups (defined by sensitive features) should receive their fair share of interest. Among previous works, Rahman et al. [29] achieve the first-of-its-kind study to realize graph embedding learning with group fairness considerations. A modified random walk algorithm is proposed to ensure that the minority (according to sensitive features) bears the same appearing probability in the walk compared with other demographic groups. Bose et al. [3] propose to disentangle the learned embeddings from the sensitive features with an adversarial learning framework. A similar adversarial approach is also adopted by Dai et al. [6] for debiasing graph mining results. Palowitch et al. [25] promote group fairness via ensuring that the node embeddings are trained on a hyperplane orthogonal to sensitive features. Buyl et

al. [4] disentangle the node embedding from sensitive features via enforcing the prior distribution to encode sensitive information as strongly as possible. Different from group fairness, individual fairness is much less studied on graphs. Kang et al. [17] propose to reduce bias in all three stages of a graph mining pipeline (i.e., preprocessing, processing, and post-processing [4]). However, their framework is mainly for plain networks and does not allow end-to-end training. To our best knowledge, we are the first to study the individual fairness issue of GNNs based on attributed networks.

6 CONCLUSION

Due to the superior learning capability, GNNs have been widely adopted to handle graph-structured data for various decision-making scenarios. However, leaving more and more decisions and judgments to GNNs raises societal concerns as the GNNs often do not have fairness considerations. Although some recent works have aimed to improve the fairness of GNNs for certain subgroups defined by a protected attribute, the fairness notion of GNNs at a much finer granularity (i.e., individual fairness) remains under-explored. To promote individual fairness, existing studies often need to rely on the Lipschitz condition to guarantee similar individuals have similar outcomes. In this paper, we argue the conventional definition of individual fairness based on the Lipschitz condition may have some potential issues w.r.t. the subtle Lipschitz constant and the uncalibrated distance metrics. Thus, we refine the definition of individual fairness from a ranking perspective, such that for each individual, the two ranking lists that encode its similarity with other individuals in the input space and output space are consistent with each other. To achieve this goal, we develop a novel plug-and-play framework REDRESS, which encapsulates the GNN model utility optimization and ranking-based individual fairness optimization in a joint framework and enables end-to-end training. To demonstrate the effectiveness of our proposed framework REDRESS, we present empirical evaluations on different real-world graphs under two downstream tasks. The experimental results imply that REDRESS outperforms the state-of-the-art individual fairness promoting approaches without jeopardizing the prediction performance. Besides, REDRESS is not only restricted on GNNs but can be extended onto other graph mining models and tasks, which is for future works.

7 ACKNOWLEDGEMENTS

This material is, in part, supported by the National Science Foundation (NSF) under grant number 2006844 and 1939725. We would like to thank the anonymous reviewers for their constructive feedback.

REFERENCES

[1] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki. 2006. Link prediction using supervised learning. In *Workshop on Link Analysis, Counter-Terrorism and Security.*
[2] Solon Barocas, Moritz Hardt, and Arvind Narayanan. 2017. Fairness in machine learning. *NeurIPS Tutorial* 1 (2017).
[3] Avishek Bose and William Hamilton. 2019. Compositional Fairness Constraints for Graph Embeddings. In *Proceedings of the 36th ICML (ICML '19).*
[4] Maarten Buyl and Tijl De Bie. 2020. DeBayes: a Bayesian Method for Debiasing Network Embeddings. In *Proceedings of the 37th ICML (ICML '20).*
[5] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. 2009. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th CIKM (CIKM '09).*
[6] Enyan Dai and Suhang Wang. 2020. Say No to the Discrimination: Learning Fair Graph Neural Networks with Limited Sensitive Attribute Information. arXiv:2009.01454 [cs.LG]
[7] Mengnan Du, Fan Yang, Na Zou, and Xia Hu. 2019. Fairness in deep learning: A computational perspective. *arXiv preprint arXiv:1908.08843* (2019).
[8] C. Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd ITCS (ITCS '12).*
[9] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *Proceedings of the 20th WWW (WWW '10).*
[10] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *Proceedings of the 21st SIGKDD (KDD '15).*
[11] Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* (2010).
[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st NeurIPS (NIPS '17).*
[13] X. Han, R. Ding, L. Wang, and H. Huang. 2019. CreditPrint: Credit Investigation via Geographic Footprints by Deep Learning. *arXiv:1910.08734* (2019).
[14] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of opportunity in supervised learning. In *Proceedings of the 30th NeurIPS (NIPS '16).*
[15] Xiao Huang, Jundong Li, and Xia Hu. 2017. Label informed attributed network embedding. In *Proceedings of the 10th WSDM (WSDM '17).*
[16] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446.
[17] J. Kang, J. He, R. Maciejewski, and H. Tong. 2020. InFoRM: Individual Fairness on Graph Mining. In *Proceedings of the 26th SIGKDD (KDD '20).*
[18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd ICLR (ICLR '15).*
[19] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[20] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).
[21] Preethi Lahoti, Krishna P Gummadi, and Gerhard Weikum. 2019. ifair: Learning individually fair data representations for algorithmic decision making. In *Proceedings of the 35th ICDE (ICDE '19).* IEEE.
[22] Preethi Lahoti, K. P. Gummadi, and G. Weikum. 2019. Operationalizing Individual Fairness with Pairwise Fair Representations. *Proc. VLDB Endow.* (2019).
[23] Jure Leskovec and Julian J Mcauley. 2012. Learning to discover social circles in ego networks. In *Proceedings of the 26th NeurIPS (NIPS '12).*
[24] Ninareh M., Fred M., Nripsuta S., Kristina L., and Aram G. 2019. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635* (2019).
[25] John Palowitch and Bryan Perozzi. 2019. Monet: Debiasing graph embeddings via the metadata-orthogonal training unit. *arXiv preprint arXiv:1909.11793* (2019).
[26] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407* (2018).
[27] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).
[28] C Quoc and Viet Le. 2007. Learning to rank with nonsmooth cost functions. *Proceedings of the 21st NIPS (NIPS '07)* (2007).
[29] T. A Rahman, B. Surma, M. Backes, and Y. Zhang. 2019. Fairwalk: Towards Fair Graph Embedding. In *Proceedings of the 28th IJCAI (IJCAI '19).*
[30] P. Sen, Ga. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. 2008. Collective classification in network data. *AI Magazine* (2008).
[31] O. Shchur, Maximilian Mumme, Aleksandar B., and Stephan G. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).
[32] V. Shumovskaia, K. Fedyanin, I. Sukharev, D. Berestnev, and M. Panov. 2020. Linking Bank Clients using Graph Neural Networks Powered by Rich Transactional Data: Extended Abstract. In *Proceedings of the 7th DSAA (DSAA '20).*
[33] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th SIGKDD (KDD '08).*
[34] Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *Proceedings of the 15th SIGKDD (KDD '09).*
[35] Petar V., Guillem C., Arantxa C., Adriana R., Pietro L., and Yoshua B. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[36] D. Wang, J. Lin, P. Cui, Q. Jia, Z. Wang, Y. Fang, Q. Yu, J. Zhou, S. Yang, and Y. Qi. 2019. A Semi-supervised Graph Attentive Network for Financial Fraud Detection. In *Proceedings of the 19th ICDM (ICDM '19).*
[37] Yaojing Wang, Guosheng Pan, Yuan Yao, Hanghang Tong, Hongxia Yang, Feng Xu, and Jian Lu. 2020. Bringing Order to Network Embedding: A Relative Ranking based Approach. In *Proceedings of the 29th CIKM (CIKM '20).*
[38] Yanhao Wei, Pinar Yildirim, Christophe Van den Bulte, and Chrysanthos Dellarocas. 2016. Credit scoring with social network data. *Marketing Science* (2016).
[39] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th ICML (ICML '19).*
[40] Muhammad Bilal Zafar, Isabel Valera, Manuel Rodriguez, Krishna Gummadi, and Adrian Weller. 2017. From parity to preference-based notions of fairness in classification. In *Proceedings of the 31st NeurIPS (NIPS '17).*
[41] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. 2013. Learning fair representations. In *Proceedings of the 30th ICML (ICML '13).*

A APPENDIX

A.1 Reproducibility

In this section, we present the details of the implementation of REDRESS and other baselines as the supplement to Section 4.3.

Experimental Settings on Datasets. For *node classification* task, for all three datasets, we utilize the split rate for training set and validation set as 5% and 10%, respectively. The rest nodes are utilized as the test data. For *link prediction* task, we randomly mask 2.5% and 5% edges for all three datasets as the positive sampled edges in validation set and test set. Same number of negative edges are also randomly generated for validation and test set. For all experiments, datasets are randomly shuffled before training, and only training data is visible for all models. Feature dimensions of all datasets are reduced to 200 using PCA from sklearn package to reduce the time complexity of computing cosine similarity.

Implementation of REDRESS. The proposed framework REDRESS is implemented in Pytorch and the code can be found here: https://github.com/Anonymoussubmissionpurpose/REDRESS. All models including REDRESS is trained with Adam optimizer, with the learning rate being 0.01. γ is set as 1 for both downstream tasks. We inherit the hyper-parameter settings (e.g., hidden unit number) of the corresponding GNN backbone for REDRESS throughout our experiments, and only search for the optimal value of α and epoch number. Search space of α is between 1e-5 and 1e-1, and specific hyper-parameter values can be referred to our open source code. Experimental results are reported on both $NDCG@k$ and $ERR@k$. We choose $k = 10$ for quantitative experimental results presentation, but verification based on different k choices is also provided.

Implementation of Baselines. For all baselines, we vary the epoch number from 200 to 600 and the optimal performances on balancing the model utility and individual fairness promotion are reported in this paper. Here we introduce the implementation of all baselines as follows.

- **GNN Backbones.** For all GNN backbones, we utilize their released public code. It should be noted that the released source code of GAE is based on Tensorfllow. For fair comparison, we adapt GAE model to Pytorch based on the released source code. We set the layer and hidden unit number for all GCN and SGC based models (including the vanilla model baseline) as 2 and 16, respectively. The graph convolutional layer number of all GAE based models is set as 3, with two hidden unit number being 32 and 16. ReLU is assigned as the activation function between graph convolutional layers in GCN and SGC. Sigmoid function is utilized as the activation function of the inner product output of GAE.
- **PFR.** For PFR, we re-implement the feature pre-processing algorithm based on the original paper in Python. The corresponding feature pre-processing debiasing algorithm requires two weighted different similarity matrices as input in their original paper. In our experiments, we utilize the weighted sum of the oracle similarity matrix $\mathbf{S}_{\mathcal{G}}$ and adjacency matrix $\mathbf{A}$ as the corresponding pre-processing algorithm input. The weights of the two matrices are adjusted between 0 and 1 with the weights sum being 1.
- **InFoRM.** For InFoRM, we compute the individual fairness loss term defined in their original paper and add it onto the vanilla GNN loss function for minimization. We also adjust the weight of the individual fairness loss term in our experiments. To achieve comparable model utility (i.e., node classification *ACC* and link prediction *AUC*) with other models, we search the weight of the individual fairness loss term between 1e-11 and 2e-7.

Packages Required for Implementations. Key packages and their corresponding versions that we utilized for our implementations are listed as follows.

- Python == 3.8
- torch == 1.4.0
- cuda == 10.1
- numpy == 1.18.2
- tensorboard == 2.4.0
- scipy == 1.4.1
- networkx == 2.1
- scikit-learn == 0.19.2

A.2 Supplementary Experimental Results

Supplementary quantitative performance of REDRESS and other alternatives based on ERR (Expected Reciprocal Rank) are also provided in Table 5 and Table 6 for node classification and link prediction task, respectively. Similar to section 4.4, the performance of REDRESS is also compared with other baselines under different settings of oracle similarity matrix (i.e., feature similarity and structure similarity) and different GNN backbones (i.e., GCN, SGC, and GAE). It should be noted that higher *ACC* and *AUC* represents better performance on model utility, and higher *ERR*@10 indicates better performance on individual fairness. We can make the following observations from the two tables:

- From the perspective of model utility (i.e., *ACC* in node classification and *AUC* in link prediction), REDRESS provides competitive performance compared with other state-of-the-art baselines. Besides, similar to the *NDCG*@10 based experiments (Table 3 and Table 4), better utility performance from REDRESS can also be found compared with the vanilla GNN backbones. This further verifies that the individual fairness promotion term plays the role of regularization to prevent over-fitting of the backbone GNN models.
- From the perspective of ranking based individual fairness, our framework outperforms all baseline methods in all cases with different levels of improvement w.r.t. the fairness evaluation metric *ERR*@10. Considering that similar observation can also be found in *NDCG*@10 based experiments as in Table 3 and Table 4, the generalization ability of REDRESS on individual fairness promotion can be further verified.
- From the perspective of balancing the model utility and individual fairness, our framework achieves both competitive model utility performance and superior individual fairness promotion in all *ERR*@10 based cases compared with other baselines. Similar observation can also be found in *NDCG*@10 based experiments as in Table 3 and Table 4. Based on these observations, we argue that our framework generally achieves better performance on balancing the model utility and individual fairness compared with other alternatives under different ranking similarity metrics.

Table 5: Node classification results on ACM, Co-author-CS (CS) and Co-author-Phy (Phy) datasets. BB represents the backbone GNN model. Vanilla denotes the vanilla GNN. All values are reported in percentage. The relative improvement of each entry compared with the corresponding backbone performance is denoted in the parentheses. Best performance is marked in bold.

	BB	Model	Feature Similarity		Structural Similarity	
			Utility: ACC	Fairness: ERR@10	Utility: ACC	Fairness: ERR@10
ACM	GCN	Vanilla	72.49 ± 0.6 (—)	75.70 ± 0.6 (—)	**72.49** ± 0.6 (—)	37.55 ± 0.4 (—)
		InFoRM	67.65 ± 1.0 (−6.68%)	73.49 ± 0.5 (−2.92%)	65.91 ± 0.2 (−9.07%)	19.96 ± 0.6 (−46.8%)
		PFR	68.48 ± 0.6 (−5.53%)	76.28 ± 0.1 (+0.77%)	70.22 ± 0.7 (−3.13%)	36.54 ± 0.4 (−2.69%)
		REDRESS (Ours)	**73.46 ± 0.2** (+1.34%)	**82.27 ± 0.1** (+8.68%)	71.87 ± 0.4 (−0.86%)	**43.74 ± 0.0** (+16.5%)
	SGC	Vanilla	**68.40 ± 1.0** (—)	80.06 ± 0.1 (—)	**68.40 ± 1.0** (—)	45.95 ± 0.3 (—)
		InFoRM	67.96 ± 0.5 (−0.64%)	75.63 ± 0.5 (−5.53%)	66.16 ± 0.6 (−3.27%)	39.79 ± 0.1 (−13.4%)
		PFR	67.69 ± 0.4 (−1.04%)	76.80 ± 0.1 (−4.07%)	66.69 ± 0.3 (−2.50%)	46.99 ± 0.5 (+2.26%)
		REDRESS (Ours)	66.51 ± 0.3 (−2.76%)	**82.32 ± 0.3** (+2.82%)	67.10 ± 0.7 (−1.90%)	**49.02 ± 0.2** (+6.68%)
CS	GCN	Vanilla	**90.59 ± 0.3** (—)	80.41 ± 0.1 (—)	**90.59** ± 0.3 (—)	26.69 ± 1.3 (—)
		InFoRM	88.37 ± 0.9 (−2.45%)	80.63 ± 0.6 (+0.27%)	87.10 ± 0.9 (−3.85%)	29.68 ± 0.6 (+11.2%)
		PFR	87.62 ± 0.2 (−3.28%)	76.26 ± 0.1 (−5.16%)	85.66 ± 0.7 (−5.44%)	19.80 ± 1.4 (−25.8%)
		REDRESS (Ours)	90.06 ± 0.5 (−0.59%)	**83.24 ± 0.2** (+3.52%)	89.81 ± 0.2 (−0.86%)	**32.42 ± 1.6** (+21.5%)
	SGC	Vanilla	87.48 ± 0.8 (—)	90.58 ± 0.1 (—)	87.48 ± 0.8 (—)	43.28 ± 0.2 (—)
		InFoRM	87.31 ± 0.5 (−0.19%)	90.64 ± 0.1 (+0.07%)	88.21 ± 0.9 (+0.83%)	43.37 ± 0.1 (+0.21%)
		PFR	87.95 ± 0.2 (+0.54%)	79.85 ± 0.2 (−11.8%)	86.93 ± 0.1 (−0.63%)	38.83 ± 0.8 (−10.3%)
		REDRESS (Ours)	**90.48 ± 0.2** (+3.43%)	**92.03 ± 0.1** (+1.60%)	**90.39 ± 0.1** (+3.33%)	**45.81 ± 0.0** (+5.85%)
Phy	GCN	Vanilla	**94.81 ± 0.2** (—)	73.25 ± 0.3 (—)	**94.81 ± 0.2** (—)	2.58 ± 0.1 (—)
		InFoRM	88.67 ± 0.7 (−6.48%)	73.80 ± 0.6 (+0.75%)	94.68 ± 0.2 (−0.14%)	2.45 ± 0.1 (−5.04%)
		PFR	88.79 ± 0.2 (−6.35%)	73.32 ± 0.4 (+0.10%)	89.69 ± 1.0 (−5.40%)	1.67 ± 0.1 (−35.3%)
		REDRESS (Ours)	93.71 ± 0.1 (−1.16%)	**80.23 ± 0.1** (+9.53%)	93.91 ± 0.4 (−0.95%)	**3.22 ± 0.3** (+24.8%)
	SGC	Vanilla	94.45 ± 0.2 (—)	77.48 ± 0.2 (—)	94.45 ± 0.2 (—)	4.50 ± 0.1 (—)
		InFoRM	92.06 ± 0.2 (−2.53%)	75.13 ± 0.4 (−3.03%)	94.27 ± 0.1 (−0.19%)	4.44 ± 0.0 (−1.33%)
		PFR	87.39 ± 1.2 (−7.47%)	73.42 ± 0.2 (−5.24%)	89.16 ± 0.3 (−5.60%)	3.41 ± 0.2 (−24.2%)
		REDRESS (Ours)	**94.81 ± 0.2** (+0.38%)	**79.57 ± 0.2** (+2.70%)	**94.54 ± 0.1** (+0.10%)	**4.98 ± 0.1** (+10.7%)

Table 6: Link prediction results on BlogCatalog (Blog), Flickr and Facebook (FB) datasets. BB represents the backbone GNN model. Vanilla denotes the vanilla GNN. All values are reported in percentage. The relative improvement of each entry compared with the corresponding backbone performance is denoted in the parentheses. Best performance is marked in bold.

	BB	Model	Feature Similarity		Structural Similarity	
			Utility: AUC	Fairness: ERR@10	Utility: AUC	Fairness: ERR@10
Blog	GCN	Vanilla	**85.87 ± 0.1** (—)	67.95 ± 0.1 (—)	85.87 ± 0.1 (—)	38.63 ± 0.2 (—)
		InFoRM	80.14 ± 0.1 (−6.67%)	68.55 ± 0.1 (+0.88%)	83.68 ± 0.0 (−2.55%)	34.26 ± 0.9 (−11.3%)
		PFR	83.65 ± 0.0 (−2.59%)	68.04 ± 0.3 (+0.13%)	84.72 ± 0.1 (−1.34%)	37.28 ± 0.4 (−3.49%)
		REDRESS (Ours)	83.90 ± 0.2 (−2.29%)	**72.83 ± 0.2** (+7.18%)	**86.44 ± 0.0** (+0.66%)	**42.16 ± 0.1** (+9.14%)
	GAE	Vanilla	**85.72 ± 0.1** (—)	67.92 ± 0.1 (—)	85.72 ± 0.1 (—)	44.23 ± 0.2 (—)
		InFoRM	81.87 ± 0.1 (−4.49%)	68.36 ± 0.4 (+0.65%)	82.50 ± 0.1 (−3.76%)	33.98 ± 0.5 (−23.2%)
		PFR	83.49 ± 0.1 (−2.60%)	67.89 ± 0.0 (−0.04%)	84.31 ± 0.1 (−1.64%)	39.89 ± 0.2 (−9.81%)
		REDRESS (Ours)	85.30 ± 1.5 (−0.49%)	**69.62 ± 0.4** (+2.50%)	**85.77 ± 2.0** (+0.06%)	**47.44 ± 0.3** (+7.26%)
Flickr	GCN	Vanilla	92.20 ± 0.3 (—)	70.39 ± 0.1 (—)	92.20 ± 0.3 (—)	38.44 ± 0.5 (—)
		InFoRM	91.28 ± 0.0 (−1.00%)	72.17 ± 0.0 (+2.53%)	**92.24 ± 0.0** (+0.04%)	39.03 ± 0.4 (+1.53%)
		PFR	**92.43 ± 0.2** (+0.25%)	71.36 ± 0.2 (+1.38%)	92.06 ± 0.2 (−0.15%)	37.29 ± 0.7 (−2.99%)
		REDRESS (Ours)	87.89 ± 0.4 (−4.67%)	**73.90 ± 0.3** (+4.99%)	91.39 ± 0.0 (−0.88%)	**44.82 ± 0.5** (+16.6%)
	GAE	Vanilla	89.98 ± 0.1 (—)	70.34 ± 0.2 (—)	89.98 ± 0.1 (—)	36.98 ± 0.3 (—)
		InFoRM	90.56 ± 1.4 (+0.64%)	71.54 ± 0.1 (+1.71%)	**91.55 ± 0.2** (+1.74%)	35.58 ± 0.4 (−3.79%)
		PFR	90.44 ± 0.2 (+0.51%)	71.65 ± 0.2 (+1.86%)	90.09 ± 0.2 (+0.12%)	33.89 ± 0.3 (−8.36%)
		REDRESS (Ours)	**93.06 ± 0.3** (+3.42%)	**72.41 ± 0.2** (+2.94%)	87.96 ± 0.4 (−2.24%)	**44.00 ± 0.1** (+19.0%)
FB	GCN	Vanilla	95.60 ± 1.7 (—)	61.52 ± 0.5 (—)	**95.60 ± 1.7** (—)	32.18 ± 1.7 (—)
		InFoRM	90.66 ± 0.0 (−5.17%)	61.49 ± 0.2 (−0.05%)	94.65 ± 1.3 (−0.99%)	30.03 ± 1.7 (−6.68%)
		PFR	89.85 ± 2.0 (−6.01%)	62.02 ± 0.3 (+0.81%)	92.30 ± 0.5 (−3.45%)	30.62 ± 1.8 (−4.85%)
		REDRESS (Ours)	**95.99 ± 1.9** (+0.41%)	**64.08 ± 0.1** (+4.16%)	92.93 ± 0.8 (−2.79%)	**43.74 ± 1.5** (+35.9%)
	GAE	Vanilla	**98.54 ± 0.0** (—)	63.19 ± 0.1 (—)	**98.54 ± 0.0** (—)	42.17 ± 0.4 (—)
		InFoRM	92.80 ± 0.1 (−5.83%)	62.29 ± 0.0 (−1.42%)	94.75 ± 0.2 (−3.85%)	31.93 ± 0.6 (−24.3%)
		PFR	96.85 ± 0.1 (−1.72%)	61.71 ± 0.1 (−2.34%)	98.18 ± 0.1 (−0.37%)	39.04 ± 0.3 (−7.42%)
		REDRESS (Ours)	95.10 ± 0.7 (−3.49%)	**64.40 ± 0.7** (+1.91%)	92.35 ± 0.3 (−6.28%)	**44.54 ± 0.3** (+5.62%)