Fairness-Aware Link Analysis

Sotiris Tsioutsiouliklis  
stsiouts@cse.uoi.gr  
Computer Science & Engineering  
University of Ioannina

Evaggelia Pitoura  
pitoura@cse.uoi.gr  
Computer Science & Engineering  
University of Ioannina

Panayiotis Tsaparas  
tsap@cse.uoi.gr  
Computer Science & Engineering  
University of Ioannina

Ilias Kleftakis  
ikleftakis@cse.uoi.gr  
Computer Science & Engineering  
University of Ioannina

Nikos Mamoulis  
nikos@cse.uoi.gr  
Computer Science & Engineering  
University of Ioannina

ABSTRACT

Algorithmic fairness has attracted significant attention in the past years. Surprisingly, there is little work on fairness in networks. In this work, we consider fairness for link analysis algorithms and in particular for the celebrated PageRank algorithm. We provide definitions for fairness, and propose two approaches for achieving fairness. The first modifies the jump vector of the PageRank algorithm to enforce fairness, and the second imposes a fair behavior per node. We also consider the problem of achieving fairness while minimizing the utility loss with respect to the original algorithm. We present experiments with real and synthetic graphs that examine the fairness of PageRank and demonstrate qualitatively and quantitatively the properties of our algorithms.

1 INTRODUCTION

Today, algorithmic systems driven by large amounts of data are increasingly being used in all aspects of life. Often, such systems are being used to assist, or even replace human decision making. This increased dependence on algorithms has given rise to the field of algorithmic fairness, where the goal is to ensure that algorithms do not exhibit biases towards specific individuals, or groups of users (see e.g., [11] for a survey). We also live in a connected world where networks, be it, social, communication, interaction, or cooperation networks, play a central role. However, surprisingly, fairness in networks has received less attention.

Link analysis algorithms, such as Pagerank [6], HITS [15], or SALSA [16], take a graph as input and use the structure of the graph to determine the relative importance of its nodes. The output of the algorithms is a numerical weight for each node that reflects its importance. The weights are used to produce an ordering of the nodes and as input features in a variety of machine learning algorithms including classification [7], and search result ranking [6]. Previous research has only considered algorithms that weight nodes according to their degree, and found biases that arise as a network evolves [2, 24].

In this work, we focus on the Pagerank algorithm [6]. Pagerank performs a random walk on the input graph, and ranks the nodes according to the stationary probability of this walk. At every step, the random walk restarts with probability $\gamma$. The restart node is selected according to the "jump" distribution vector $v$. Since its introduction in the Google search engine, Pagerank has been the cornerstone algorithm in several applications (see, e.g., [12]).

As in previous research, we view fairness as lack of discrimination against a protected group defined by the value of a sensitive attribute, such as, gender, or race [11]. We operationalize this view by saying that a link analysis algorithm is $\phi$-fair, if the fraction of the total weight allocated to the members of the protected group is $\phi$. The value of $\phi$ is a parameter that can be used to implement different fairness policies. In the simplest case, $\phi$ is set equal to the fraction of the protected nodes in the graph, asking that these nodes have a share in the weights proportional to their share in the population. We also consider targeted fairness, where we focus on a specific subset of nodes to which we want to allocate weight in a fair manner.

We revisit Pagerank through the lens of our fairness definition, and we consider the problem of defining Pagerank variants that are fair. We also define the utility loss of a fair algorithm as the difference between its output and the output of the Pagerank algorithm, and we consider the problem of achieving fairness while minimizing utility.

We consider two approaches for achieving fairness. Our first approach, the fairness-sensitive Pagerank algorithm, exploits the jump vector $v$. There has been a lot of work on modifying the jump vector to obtain variants of pagerank biased towards a specific set of nodes, for example, in personalized pagerank all jump probability is assigned to a single node, while in topic-sensitive pagerank the probability is assigned to nodes of a specific topic [13]. In this paper, we take the novel approach of using the jump vector to achieve $\phi$-fairness. We determine the conditions under which this is feasible and formulate the problem of finding the jump vector that achieves $\phi$-fairness while minimizing utility loss from the original pagerank as a convex optimization problem.

Our second approach takes a microscopic view by looking at the behavior of each individual node in the graph. Implicitly, a link analysis algorithm assumes that links in the graph correspond to endorsements between the nodes. Therefore, we can view each node, as an agent that endorses (or votes for) the nodes that it links to. The link analysis algorithm defines a process that takes these individual actions of the nodes and transforms them into a global weighting of the nodes. To this end, we introduce, the locally fair PageRank algorithms, where each individual node acts fairly by distributing its own pagerank to the protected and non-protected groups according to the fairness ratio $\phi$. Local fairness defines a dynamic process that can be viewed as a fair random walk, where at each step of the random walk (not only at convergence), the probability of being at a node of the protected group is $\phi$. 


In our first locally fair PageRank algorithm, termed the neighborhood locally fair pagerank algorithm, each node distributes its Pagerank fairly among its immediate neighbors, allocating a fraction \( \phi \) to the neighbors in the protected group, and \( 1 - \phi \) to the neighbors in the non-protected group. Or, in random walk terms, at each node the probability of transitioning to a neighbor in the protected group is \( \phi \) and the probability of transitioning to a non-protected neighbor is \( 1 - \phi \). The residual-based locally fair pagerank algorithms generalizes this idea. Consider a node \( i \) that has less neighbors in the protected group than \( \phi \). The node distributes an equal portion of its pagerank to each of its neighbors and a residual portion \( \delta(i) \) to members in the protected group but not necessarily in its neighborhood. Or, in random walk terms, at each node \( i \), the probability of transitioning to a neighbor is \( 1 - \delta(i) \) and the probability of transitioning to a node in the protected group is \( \delta(i) \).

The residual is allocated based on a residual redistribution policy, which allows us to control the fairness policy. In this paper, we use the residual redistribution policy to minimize the utility loss.

Finally, we present a post-processing approach that given the output of a link analysis algorithm, it redistributes the weights so as to attain fairness. This gives us a lower bound on the utility loss. We study the fairness of the original pagerank in both real and synthetic networks. We also evaluate quantitatively and qualitatively the output of our fairness-sensitive algorithms. The weights produced by the neighborhood locally fair Pagerank tend to promote protected nodes lying on the boundaries of the two groups especially in homophilic networks, while the fairness-sensitive Pagerank tends to jump to protected nodes especially when the requested \( \phi \) is large.

In summary, in this paper we make the following contributions:

- We initiate a study of fairness in link analysis. To the best of our knowledge we are the first to consider fairness in this problem.
- We propose the fairness-sensitive Pagerank algorithm that modifies the jump vector so as to attain fairness and the locally fair Pagerank algorithms that guarantee that individually each node behaves in a fair manner.
- We formulate optimization problems for finding the algorithms that minimize the utility loss and estimate a lower bound for the optimal utility loss by post-processing the output of Pagerank.
- We perform experiments on several datasets. Our experiments demonstrate qualitatively and quantitatively the properties of the fair Pagerank algorithms.

2 DEFINITIONS

A link analysis algorithm can be seen as a function \( A : G^n \rightarrow \mathbb{R}^n \) from the set \( G^n \) of all graphs of size \( n \) to the real vectors of size \( n \). The function takes as input a graph \( G = (V, E) \) (directed, or undirected) of size \( n \), and produces a vector \( w \) of size \( n \), which assigns a weight \( w_v \) to each node \( v \) in the graph. This weight defines the importance of the node in the graph \( G \), and it depends on the graph structure. The best known link analysis algorithm is PageRank, which we consider in this paper.

Given a graph \( G = (V, E) \), we assume that there exists a subset of nodes that define a protected group. This group may be defined based on a protected attribute of the nodes in the graph, such as race or gender. In this paper, we consider two types of nodes, the groups \( R \) and \( B \) of red and blue nodes, and we assume that \( R \) is the protected group. We denote with \( r = \frac{|R|}{n} \) and \( b = \frac{|B|}{n} \) the fraction of nodes that belong to the red and blue group respectively.

We will say that a link analysis algorithm is fair, if it assigns weights to each group according to a specified ratio \( \phi \). Ratio \( \phi \) may be specified so as to implement specific affirmative action policies, or other fairness enhancing interventions. For example, \( \phi \) may be set in accordance to the 80 percent rule advocated by the US Equal Employment Opportunity Commission (EEOC), or some other formulation of disparate impact [10].

**Definition 2.1 (Fair link analysis).** A link analysis algorithm \( A : G^n \rightarrow \mathbb{R}^n \) is \( \phi \)-fair on graph \( G \), if for the output \( w = A(G) \), it holds that:

\[
\frac{\sum_{v \in R} w_v}{\sum_{v \in V} w_v} = \phi \quad \text{where} \quad R \subset V \text{ is the protected set of nodes.}
\]

For instance by setting \( \phi = r \), we ask for a fair link analysis algorithm that assigns weights proportionally to the sizes of the two groups. In this case, fairness is analogous to demographic parity, i.e., the requirement that the demographics of those receiving a positive outcome are identical to the demographics of the population as a whole [8]. It is easy to show (see Appendix) that in this case the weights produced by the fair link analysis are such that the average weight of the red nodes is the same with the average weight of the blue nodes.

We define the following problem:

**Problem 1.** Given a value \( \phi \), a graph \( G \), and a link analysis algorithm \( A \), design a link analysis algorithm \( A_F \) that is \( \phi \)-fair on graph \( G \).

Note that the fair variant \( A_F \) will necessarily change the original weights of algorithm \( A \), incurring some loss in utility. We quantify the utility loss using the sum of squares loss function \( L(A, A_F) = \|A(G) - A_F(G)\|^2 \). We then consider the problem of designing a fair algorithm that minimizes utility loss.

**Problem 2.** Given a value \( \phi \), a graph \( G \), and a link analysis algorithm \( A \), design a link analysis algorithm \( A_F \) that is \( \phi \)-fair on graph \( G \), such that the utility loss \( L(A(G), A_F(G)) \) is minimized.

Finally, we consider an extension of the fairness definition that asks for a fair distribution of weights among a specific set of nodes \( S \) that is given as input. We assume that the set \( S \) is selected such that it contains nodes from both groups \( R \) and \( B \).

**Definition 2.2 (Targeted Fair link analysis).** A link analysis algorithm \( A : G^n \rightarrow \mathbb{R}^n \) is targeted \( \phi \)-fair on graph \( G = (V, E) \) for a set of nodes \( S \subset V \), if for the output \( w = A(G) \), it holds that:

\[
\frac{\sum_{v \in S \cap R} w_v}{\sum_{v \in V} w_v} = \phi \quad \text{where} \quad R \subset V \text{ is the protected set of nodes.}
\]

In this paper, we consider the PageRank link analysis algorithm.

**The PageRank Algorithm.** The PageRank algorithm is the best-known link analysis algorithm, popularized by its application in the Google search engine. The scoring vector of the algorithm is the stationary distribution of a random walk on the graph \( G \). We will use \( p \) to denote this probability vector (which is the same as the scoring vector \( w \)).
The PageRank random walk is parameterized by the value $\gamma$ which is the probability that the random walk will restart. Typically, the jump probability is set to $\gamma = 0.15$. The node from which the random walk restarts is selected according to the jump vector $\mathbf{v}$, which defines a distribution over the nodes in the graph. Typically, the jump vector is set to the uniform vector $\mathbf{u}$.

Let $\mathbf{P}$ denote the normalized adjacency matrix of graph $G$. The matrix $\mathbf{P}$ defines the transition probability $P(i,j)$ between two nodes $i$ and $j$. Assuming that there are no sink nodes, we have that

$$
\mathbf{p}^T = (1-\gamma)\mathbf{p}^T \mathbf{P} + \gamma \mathbf{v}^T \tag{1}
$$

In the case where there are sink nodes in the graph, we assume that the random walk performs a jump to a node chosen uniformly at random [12]. That is, the corresponding zero-rows in the matrix $\mathbf{P}$ are replaced by the uniform vector $\mathbf{u}$.

## 3 FAIRNESS SENSITIVE PAGERANK

Our first algorithm achieves fairness by keeping the transition matrix fixed and changing the jump vector $\mathbf{v}$ so as to meet the fairness criterion.

### 3.1 The Algorithm

First, we note that that pagerank vector $\mathbf{p}$ can be written as linear function of the jump vector $\mathbf{v}$. Solving Equation (1) for $\mathbf{p}$, using column vector notation, we have that $\mathbf{p} = \mathbf{Qv}$, where

$$
\mathbf{Q} = \gamma \left( [I - (1-\gamma) \mathbf{P}]^{-1} \right)^T
$$

Let $\mathbf{p}_R$ denote the pagerank mass that is allocated to the nodes of the protected category. We have that

$$
\mathbf{p}_R = \left( \sum_{i \in R} \mathbf{Q}_i \right) \mathbf{v} = \mathbf{Q}_R^T \mathbf{v}
$$

where $\mathbf{Q}_R^T$ is the $i$-th row of matrix $\mathbf{Q}$, and $\mathbf{Q}_R^T$ is the vector that is the sum of the rows in the set $R$. In order for the algorithm to be fair, we need $\mathbf{p}_R \geq \phi$. Our goal is to find a vector $\mathbf{v}$ such that $\mathbf{Q}_R^T \mathbf{v} \geq \phi$.

Does such a vector always exist? We prove the following:

**Lemma 3.1.** Given the vector $\mathbf{Q}_R^T$, there exists a vector $\mathbf{v}$ such that $\mathbf{Q}_R^T \mathbf{v} \geq \phi$, if and only if, there exists entries $i, j$ in $\mathbf{Q}_R^T$, where $\mathbf{Q}_R^T(i) \leq \phi$ and $\mathbf{Q}_R^T(j) \geq \phi$.

**Proof.** We have that $\mathbf{p}_R = \sum_{j=1}^N \mathbf{Q}_R^T(j) v_j$, that is, $\mathbf{p}_R$ is the weighted average of the values $\mathbf{Q}_R^T(j)$, with weights $v_j$, where $0 \leq v_j \leq 1$. Since $\mathbf{Q}_R^T \mathbf{v} = \phi$, there must exist at least one entry $i$ with $\mathbf{Q}_R^T(i) \leq \phi$, and one entry $j$ $\mathbf{Q}_R^T(j) \geq \phi$. Conversely, if there exists two such entries $i, j$, then we can find values $v_i$ and $v_j$, such that $v_i \mathbf{Q}_R^T(i) + v_j \mathbf{Q}_R^T(j) = \phi$ and $v_i + v_j = 1$. \[\square\]

### 3.2 Optimizing Utility

An implication of Lemma 3.1 is that, in most cases, there are multiple jump vectors that give a fair pagerank vector. We are interested in the solution that minimizes the utility loss.

We first consider the case were we want fairness over all nodes. To solve this problem we exploit the fact that the utility loss function $L(\mathbf{p}_R, \mathbf{p}_u) = \|\mathbf{p}_R - \mathbf{p}_u\|^2$ is convex, and that we can express the fairness requirement as a linear function. We can then define the following convex optimization problem.

$$
\begin{align*}
\text{minimize} & \quad \|\mathbf{Qx} - \mathbf{p}_u\|^2 \\
\text{subject to} & \quad \mathbf{Q}_R^T \mathbf{x} = \phi \\
& \quad \sum_{i=1}^n x_i = 1 \\
& \quad 0 \leq x_i \leq 1, \quad i = 1, \ldots, n
\end{align*}
$$

This problem can be solved using standard convex optimization solvers.

### 3.3 Targeted Fairness Algorithm

We will now formulate a similar convex optimization problem for the targeted fairness problem. Let $\mathbf{Q}_S^T = \sum_{i \in S} \mathbf{Q}_i^T$ be the sum of rows of $\mathbf{Q}$ for the nodes in $S$, and $\mathbf{Q}_R^T(N) = \sum_{i \in N \setminus R} \mathbf{Q}_i^T$ be the sum of rows of $\mathbf{Q}$ for the $R$ nodes in $S$. We define a convex optimization problem that is exactly the same as in Section 3.2, except for the fact that we replace the constraint $\mathbf{Q}_R^T \mathbf{x} = \phi$ with the constraint $\mathbf{Q}_S^T \mathbf{x} = \phi \mathbf{Q}_S^T \mathbf{x}$.

We can model specific cases by adding additional constraints. For example, let $T_k(\mathbf{w})$ denote the $k$ nodes with the largest weights in vector $\mathbf{w}$, and let $S = T_k(\mathbf{p}_u)$, that is, the top-$k$ nodes of the original Pagerank algorithm. We want fair redistribution of Pagerank among the nodes in $S$, but we also want these nodes to remain in the top-$k$ position in the fair pagerank, that is, $T_k(\mathbf{p}) = T_k(\mathbf{p}_u)$. This requirement can be achieved by adding the constraint:

$$
\mathbf{Q}_S^T \mathbf{x} \geq \mathbf{Q}_S^T \mathbf{x}, \quad i \in T_k(\mathbf{p}), \ j \notin T_k(\mathbf{p}).
$$

### 4 LOCALLY FAIR PAGERANK

In the locally fair PageRank algorithms, each individual node acts fairly, that is, each node distributes its own pagerank to red and blue nodes following the $\phi$ ratio.

#### 4.1 The Algorithms

#### 4.1.1 The neighborhood locally fair PageRank (LFPR$_N$) algorithm.

We first consider a node that treats its neighbors fairly, that is, by adhering to the $\phi$ ratio. Specifically, we define the neighborhood locally fair pagerank (LFPR$_N$) $\mathbf{p}_N$ as follows. Each node $i$ splits the $\phi \mathbf{p}_N(i)$ portion of its pagerank value among its red out-neighbors and the remaining $(1 - \phi) \mathbf{p}_N(i)$ portion of itspagerank evenly among its blue out-neighbors. Similarly, we use a modified jump vector $\mathbf{v}_N$ with $\mathbf{v}_N[i] = \phi / |\mathbf{R}||, \quad i \in \mathbf{R}$, and $\mathbf{v}_N[i] = \frac{1-\phi}{|\mathbf{R}||}, \quad i \in \mathbf{B}$.

Let $outg(i)$ and $outd(i)$ be the number of edges directed from node $i$ to red nodes and blue nodes respectively. We define $\mathbf{PG}$ as the normalized adjacency matrix that includes links to red nodes, or random jumps to red nodes if such links do not exist:

$$
\mathbf{P}_G(i,j) = \begin{cases} 
\frac{1}{outg(i)}, & \text{if } i \in \mathbf{R}, \ outg(i) \neq 0, \text{ and } (i,j) \in \mathbf{E} \\
\frac{1}{|\mathbf{R}|}, \ & \text{if } i \in \mathbf{R}, \ \text{and } outg(i) = 0 \\
0, & \text{otherwise}
\end{cases}
$$
With the residual algorithm, each of the pagerank to red and blue nodes.

1. and with probability \( 3^5 \) arrives at terms of the random walker interpretation, a random walker that neighbors gets 1 from its pagerank, resulting in red nodes getting 1 of their color and assigns to each of them the same portion of its pagerank. When the ratio of blue and red neighbors is different of their color and assigns to each of them the same portion of its pagerank. When the ratio of blue and red neighbors is different than \( \phi \), to be fair, node \( i \) distributes any remaining portion of its pagerank to nodes in the appropriate group. We call the remaining portion residual and denote it by \( \delta(i) \). How \( \delta(i) \) is distributed to the appropriate group is determined by a residual policy.

Intuitively, this corresponds to a fair random walker that upon arriving at a node \( i \), with probability \( 1 - \delta(i) \) follows one of \( i \)'s out-links and with probability \( \delta(i) \) jumps to one or more nodes in the underrepresented group.

We now describe the algorithm formally. We divide the nodes in \( V \) into two sets, \( L_R \) and \( L_B \), based on the fraction of their red and blue neighbors. Set \( L_R \) includes all nodes \( i \) such that \( (1 - \phi) \text{out}_{R}(i) < \phi \text{out}_{R}(i) \), that is, the nodes for which the ratio of red nodes in their neighborhood is smaller than the required \( \phi \) ratio. These are the nodes having a residual that needs to be distributed to red nodes. Analogously, \( L_B \) includes all nodes \( i \) such that \( (1 - \phi) \text{out}_{R}(i) \geq \phi \text{out}_{R}(i) \).

Let us first consider a node \( i \) in \( L_R \). Each neighbor of \( i \) gets the same portion of \( i \)'s pagerank, let \( \rho_R(i) \) be this portion. To attain the \( \phi \) ratio, the residual \( \delta_R(i) \) of \( i \)'s pagerank goes to the red nodes. Portions \( \rho_R(i) \) and \( \delta_R(i) \) must be such that:

\[
(1 - \phi) \rho_R(i) + \delta_R(i) = \phi \text{out}_{R}(i) \rho_R(i) + \delta_R(i) = 1
\]

From Equations (2) and (3), we get \( \rho_B(i) = \frac{1 - \phi}{\text{out}_{R}(i)} \) and the residual is \( \delta_R(i) = \phi - \frac{(1 - \phi) \text{out}_{R}(i)}{\text{out}_{R}(i)} \).

Analogously, for a node \( i \) in \( L_B \), we get \( \rho_B(i) = \frac{\phi}{\text{out}_{R}(i)} \) and a residual \( \delta_B(i) = (1 - \phi) - \frac{\phi \text{out}_{R}(i)}{\text{out}_{R}(i)} \) that goes to the blue nodes.

For example, consider a node \( i \) with 5 red neighbors, 1 red and 4 blue, and let \( \phi = 0.5 \). In the original Pagerank, each of the 5 neighbors gets 1/5 of \( i \)'s Pagerank, resulting in red nodes getting 1/5 and blue nodes 4/5, which is an unfair behavior for node \( i \).

With the residual algorithm, each of \( i \)'s neighbors gets 1/8 of \( i \)'s Pagerank, resulting in red neighbors getting 1/8 and blue neighbors 4/8 and the residual 3/8 goes to nodes in the red group so as to attain the \( \phi \) ratio and make \( i \) fair. Which of the nodes in the red group will get the residual is determined by the residual policy. In terms of the random walker interpretation, a random walker that arrives at \( i \), with probability 5/8 chooses one (any) of \( i \)'s out-links and with probability 3/8 jumps to nodes in the red group.

The transition matrix \( \mathbf{P}_L \) is defined as

\[
\mathbf{P}_L(i, j) = \begin{cases} 
\frac{1 - \phi}{\text{out}_{R}(i)} & \text{if } (i, j) \in E \text{ and } i \in L_R \\
\phi & \text{if } (i, j) \in E \text{ and } i \in L_B \\
0 & \text{otherwise}
\end{cases}
\]

Let \( \delta_R \) be the vector carrying the red residual, that is, \( \delta_R(i) = \phi - \frac{(1 - \phi) \text{out}_{R}(i)}{\text{out}_{R}(i)} \), if \( i \in L_R \) and 0 otherwise. Similarly, let \( \delta_B \) be the vector carrying the blue residual, that is, \( \delta_B(i) = \phi - \frac{\phi \text{out}_{R}(i)}{\text{out}_{R}(i)} \), if \( i \notin L_B \) and 0 otherwise. We have a total red residual \( \Delta_R = \mathbf{P}^T_L \delta_R \) and a total blue residual \( \Delta_B = \mathbf{P}^T_B \delta_B \), where \( \mathbf{P}_L \) is the locally fair pagerank vector.

To express the residual policy, we use two matrices, matrices \( X \) and \( Y \), that capture the policy for distributing the residual to red and blue nodes respectively. Specifically, \( X[i, j] \) denotes the portion of the \( \delta_R(i) \) of node \( i \) in \( L_R \) that goes to node \( j \) in \( R \) and \( Y[i, j] \) the portion of the \( \delta_B(i) \) of node \( i \) in \( L_B \) that goes to node \( j \) in \( B \).

The locally-fair pagerank vector \( \mathbf{p}_L \) is defined as:

\[
\mathbf{p}_L = (1 - \gamma) \mathbf{P}^*_L \mathbf{P}_L + \gamma \mathbf{v}_N^T
\]

Note that the \( \text{LFP}_{PR} \) algorithm is a special case, where the residual of a node is distributed only among its neighbors (details in the Appendix).

We consider simple residual policies where all nodes follow the same policy in distributing their residual. In this case, the residual policy is expressed through two (column) vectors \( x \) and \( y \), with \( x[i] \) being the portion of \( \Delta_R \) going to red node \( i \), and \( y[i] \) the portions of \( \Delta_B \) going to blue node \( i \), in which case, we have:

\[
\mathbf{p}_L = (1 - \gamma) \mathbf{P}^*_L \mathbf{P}_L + \gamma \mathbf{x}^T \mathbf{r} + \gamma \mathbf{y}^T \mathbf{v}_N^T
\]

Then, two intuitive policies of distributing the residual are: (1) uniformly, which gives us the uniform locally fair PageRank algorithm, \( \text{LFPR}_U \), and (2) proportionally based on the original pagerank weight \( \mathbf{p}_R(i) \) of node \( i \) which gives us the proportionally locally fair PageRank algorithm, \( \text{LFPR}_P \).

For the \( \text{LFP}_{PR} \) algorithm, we define the vectors \( \mathbf{x}_R \) as \( \mathbf{x}_R[i] = \frac{1}{|R|} \) if \( i \in R \) and 0 otherwise, and \( \mathbf{y}_U \) as \( \mathbf{y}_U[i] = \frac{1}{|B|} \) if \( i \in B \) and 0 otherwise. For the \( \text{LFP}_{PR} \) algorithm, we define the vectors \( \mathbf{x}_R \) and \( \mathbf{y}_U \) as \( \mathbf{x}_R[i] = \frac{p(i)}{\sum_{i \in R} p(i)} \) if \( i \in R \) and 0 otherwise, and \( \mathbf{y}_U \) as \( \mathbf{y}_U[i] = \frac{1}{|B|} \) if \( i \in B \) and 0 otherwise.

4.1.3 Fairness of the locally fair PageRank algorithms. In the locally fair pagerank algorithms, each node in the graph treats the red and blue nodes fairly by respecting the \( \phi \) ratio. However, each node acts independently of the other nodes in the network. It is interesting to see how this microscopic view of fairness relates to our macroscopic view of link fairness.

We prove the following theorem.

**Theorem 4.1.** The locally fair pagerank algorithms are fair.

**Proof.** We must show that \( \sum_{i \in R} \mathbf{p}_R(i) = \phi \). Since each node in the graph gives a portion \( \phi \) of its pagerank to red nodes, we have

\[
\sum_{i \in R} \mathbf{p}_R(i) = \sum_{i \in V} \phi \mathbf{p}_R(i)
\]

which proves the theorem.
4.2 Optimizing Utility

We now consider how to optimally distribute the residual so as to minimize the utility loss of the fair Pagerank. We denote this algorithm as LPRF. We can now write the vector $p_L$ as a function of the vectors $x$ and $y$ as follows:

$$p_L(x, y) = y v^T \left[ I - (1 - y)(p_L + \delta_R x^T + \delta_R y^T) \right]^{-1}$$

We can now define the optimization problem of finding the vectors $x$ and $y$ that minimize the loss function $L(p_L, p_u) = \|p_L(x, y) - p_u\|^2$, subject to the constraint that the vectors $x$ and $y$ define a distribution over the nodes in $R$ and $B$ respectively. We solve this optimization problem using gradient descent. We enforce the distribution constraints by adding a penalty term $\lambda \left( (\sum_{i=1}^n x_i - 1)^2 + (\sum_{i=1}^n y_i - 1)^2 \right)$. We enforce the positivity constraints through proper bracketing at the line-search step.

Note that we can also formulate a convex optimization problem asking for the jump vector that minimizes utility loss, as in Section 3.2. In this case, since the transition matrix is fair, we just need to constrain the jump vector to obey the $\phi$ ratio.

4.3 Targeted Fair Local Algorithms

We now consider how to apply the local algorithms to the targeted fairness case. Let $S_R$ and $S_B$ be the red and blue nodes in the set $S$ respectively, and let $I_S$ be the set of in-neighbors of $S$. The idea is that the nodes in $I_S$ should distribute their PageRank to $S_R$ and $S_B$ fairly, such that the ratio of the portion that goes to nodes in $S_R$ and the portion that goes to nodes in $S_B$ is equal to $\frac{\phi}{1 - \phi}$. We can implement the same redistribution policies as in the case of the neighborhood local and the residual-based local fair algorithms.

We also need the (global) jump vector $\delta$ to obey the $\phi$ ratio for the nodes in $S$. We can achieve this by redistributing the probability $|S|/n$ of the jump vector according to the $\phi$ ratio. Note that there is a variety of policies one could implement, depending on a specific objective. For example if we want to increase the weight of the nodes in $S$, we can make the jump vector allocate all probability to the nodes in $S$.

5 A POST PROCESSING APPROACH

We now consider a post processing approach in which we assume that we are given a weight vector $w = A(G)$ of a link analysis algorithm $A$ on graph $G$. The goal is to produce a new weight vector $f$ such that: (1) $f$ is fair, and (2) the utility loss $L(w, f) = \|w - f\|^2$ is minimized. The post-processing algorithm is agnostic to the fact that the weight vector $w$ is the result of a link analysis algorithm, much less of the specific link analysis algorithm (e.g., Pagerank). Therefore, the vector $f$ that minimizes the loss $L(w, f)$ may not be attainable by any Pagerank algorithm.

5.1 The Post Processing Algorithm

Given the weight vector $w$, let $w_R$ denote the weight vector for the nodes in $R$, and $w_B$ the weight vector for the nodes in $B$. We also use $W_R$ to denote the total weight allocated to $R$, and $W_B$ to denote the total weight allocated to $B$. We assume that $w$ has non-negative entries, and it is normalized so that its entries sum to 1. Without loss of generality assume that $W_R < \phi$. Let $\Delta = \phi - W_R$. To make the vector fair we need to distribute weight $\Delta$ to the nodes in $R$, and remove weight $\Delta$ from the nodes in $B$. It is easy to show that in order to minimize the loss, the optimal redistribution will remove weight $\Delta / |B|$ from all nodes in $B$ and add $\Delta / |R|$ from all nodes in $B$. This follows from the fact that among all distribution vectors the one with the smallest length is the uniform one. Therefore, we obtain the following lower-bound for the loss:

$$\text{Loss}_{LB} = \frac{\Delta^2}{|R|} + \frac{\Delta^2}{|B|}$$

Note that this lower bound does not guarantee that the new vector $w$ has non-negative entries, thus it is not a valid weight vector. We now describe an optimal redistribution algorithm that ensures that when removing weight no entry becomes negative, while using the principle that whenever removing weight, the optimal way is to remove uniformly from all nodes. The pseudocode for the algorithm is shown in Algorithm 1.

Algorithm 1 Optimal Redistribution Algorithm

Input: Excess weight $\Delta$, nodes $B$, weights $w_B$
Output: Optimal weight vector $f_B$

1: procedure REDISTRIBUTE($\Delta, B, w_B$)
2: $B_{NZ} \leftarrow \{ x \in B : w_x > 0 \}$
3: $\delta = \Delta / |B_{NZ}|$
4: $\beta = \min_{x \in B_{NZ}} w_x$
5: if $\beta \geq \delta$ then
6: $w_x = w_x - \delta$ for all $x \in B_{NZ}$
7: return $w_B$
8: else
9: $w_x = w_x - \beta$ for all $x \in B_{NZ}$
10: $\Delta = \Delta - |B_{NZ}| \beta$
11: return REDISTRIBUTE($\Delta, B, w_B$)
12: end if
13: end procedure

The algorithm takes as input the value of excess weight $\Delta$ that needs to be removed, the set of nodes $B$ from which we want to remove the weight, and the current weights $w_B$ of these nodes. First, it finds the subset of nodes $B_{NZ}$ in $B$ that have non-zero weight. If the minimum weight $\beta$ among these nodes is at least $\Delta / |B_{NZ}|$, then we can remove the weight uniformly without making the weights negative. The algorithm updates the weights and returns. Otherwise, we can remove at most $\beta$. The algorithm removes $\beta$ from all nodes in $B_{NZ}$ and makes a recursive call with the remaining excess weight $\Delta - |B_{NZ}| \beta$. Note that anytime we want to remove weight from a set of nodes, we remove it uniformly from all nodes, which guarantees optimality. The algorithm returns the updated weight vector $f_B$ for the nodes in $B$. We can now compute the optimal loss as

$$\text{Loss}_O = \frac{\Delta^2}{|R|} + ||f_B - w_B||^2$$

5.2 Targeted Fairness

Computing algorithmically the optimal redistribution is harder in the targeted fairness case, since there are many different options in how we can redistribute weight. We can move weight between
Figure 1: Fairness of the PageRank algorithm with the size of the protected group for varying homophily.

The nodes in \( S \), or bring in weight from outside of \( S \), or move weight out of \( S \), or a combination of those. In Appendix 8.2 we compute analytically a lower bound for the loss, which provides some intuition on how the weight is moved in different cases.

Finding the optimal redistribution vector can be formulated as a convex optimization problem:

\[
\begin{align*}
\text{minimize} & \quad \|f - p\|^2 \\
\text{subject to} & \quad \sum_{i \in S \cup \bar{R}} f_i = \phi \\
& \quad \sum_{i=1}^n f_i = 1 \\
& \quad 0 \leq f_i \leq 1, \quad i = 1, \ldots, n
\end{align*}
\]

We use the solution of the optimization problem to compare the optimal redistribution with that achieved by the modified Pagerank algorithms.

6 EXPERIMENTAL EVALUATION

In this section, we evaluate experimentally the different fair Pagerank algorithms and provide quantitative and qualitative results. We have used various real data sets. We focus on the following four, while results for additional datasets can be found in the Appendix.

- **Twitter**: A political retweet graph from [22].
- **DBLP**: An author collaboration network constructed from DBLP including a subset of data mining and database conferences.
- **Books**: A network of books about US politics where edges between books represented co-purchasing.
- **Blogs**: A directed network of hyperlinks between weblogs on US politics [1].

The characteristics of the real datasets, and the protected groups, are shown in Table 1. To infer the gender in the DBLP, we used the python gender_guesser package. We also report homophily which was shown to affect degree distributions among groups [2]. We measure it as the number of mixed edges, i.e., edges between nodes belonging to different groups, divided by \( 2r(r - 1) \), i.e., the expected number of such edges. Values significantly smaller than 1 indicate that the network exhibits homophily [9].

Synthetic networks are generated using a variation of the biased preferential attachment model introduced in [2]. The graph evolves with time as follows. Let \( G_t = (V_t, E_t) \) and \( d_t(v) \) denote the graph and the degree of node \( v \) at time \( t \), respectively. The process starts with an arbitrary initial connected graph \( G_0 \), with \( n_0 \) red and \( n_0(1 - r) \) blue nodes. At each time step \( t + 1, t > 0 \), a new node \( v \) enters the graph. The color of \( v \) is red with probability \( r \) and blue with probability \( 1 - r \). Node \( v \) chooses to connect with an existing node \( u \) with probability \( \frac{d_t(u)}{\sum_{w \in G_t} d_t(w)} \). If the color of the chosen node \( u \) is the same with the color of the new node \( v \), then an edge between them is inserted with probability \( h \); otherwise an edge is inserted with probability \( 1 - h \). If no edge is inserted, the process of selecting a neighbor for node \( v \) is repeated until an edge is created.

Parameter \( h \) controls the level of homophily in the network, where \( h = 0 \) corresponds to homophily, \( h = 0.5 \) to the random case and \( h = 1 \) to heterophily. We also consider asymmetry in homophily. In this case, the above procedure is followed by a node \( v \) only when \( v \) belongs to the red group. A node \( v \) in the blue group connects with the selected node \( u \) without testing \( u \)’s color.

**Fairness in the original Pagerank algorithm.** We use the synthetic datasets to study the behavior of PageRank for different levels of homophily and relative sizes of the two groups. For this set of experiments, we set \( \phi = r \). As shown in Figure 1, for the symmetric case, when the groups exhibit homophily (\( h = 0.2 \) and \( h = 0.4 \)), PageRank is unfair towards the minority group. On the contrary, when the groups exhibit heterophily (\( h = 0.6 \) and \( h = 0.8 \)), then PageRank is unfair towards the majority class. For the asymmetric case, i.e., when the blue group shows no homophily, being homophilic helps the red group independently of its size, while being heterophilic hurts the red group independently of its size.

For the real dataset, we report the fraction of the total weight allocated to each of the two groups in Table 1. In some cases (BLOGS, TWITTER), the fraction of the weight assigned to the protected group is significantly smaller than \( r \). In all cases, by setting \( \phi \) to the desired level of fairness, we can redistribute weights so that we get the desired \( \phi \)-fairness. We report quantitative and qualitative results for \( \phi = 0.5 \) in the next section.

To get a better insight about the distribution of the weights between the two groups, we also run personalized Pagerank algorithms starting from each node \( i \) and calculated for each node \( i \) the fraction of the weight allocated to the blue and red nodes (ignoring the Pagerank allocated to the node itself). In all graphs, most of the starting nodes allocate the majority of their personalized pagerank weights to nodes in their group, resulting in highly unfair weights. We report the histogram of the fraction of the weights allocated to blue and red node for personalized pageranks starting from each of the blue nodes in Figure 2. Correlation with homophily can be observed, with the most homophilic networks, i.e., books and twitter, showing the largest unfairness. Our locally fair Pagerank algorithms can be used to attain \( \phi \)-fairness for personalized Pagerank as well.

**The Fair PageRank Algorithms.** We run our fair Pagerank algorithm for various values of \( \phi \). In Figure 3, we report results for \( \phi = r \) and \( \phi = 0.5 \) for the fairness sensitive Pagerank (SFPR), while in
and for the targeted Pagerank algorithms can be found in the Appendix.

Table 2: Utility loss with respect to optimal utility (\(\text{LFPR}_{\text{OPTIMAL}}\), for \(\phi = 0.5\))

| Dataset   | LFPR_N | LFPR_U | LFPR_P | LFPR_O | SFFR |
|-----------|--------|--------|--------|--------|------|
| TWITTER   | 6.576  | 6.683  | 4.218  | 2.583  | 2.699|
| DBLP2     | 1.356  | 1.232  | 1.516  | 1.418  | 2.6  |
| BLOGS     | 5.05   | 5.08   | 3.163  | 3.09   | 1.73 |
| BOOKS     | 9.53   | 4.94   | 1.576  | 1.41   | 1    |

Figure 4, results for \(\phi = 0.5\) for the various locally fair Pagerank algorithms (i.e. neighbor (LFPR_N), uniform (LFPR_U), proportional (LFPR_P) and with optimized residual (LFPR_O)). Results for \(\phi = r\) and for the targeted Pagerank algorithms can be found in the Appendix.

Table 2 reports the utility loss for each of fair pagerank algorithms relative to the optimal utility loss as estimated by Algorithm 1. For the non-optimized algorithms, as expected taking into account the original pagerank values, the LFPR_P algorithm results in the smallest utility loss. The LFPR_N algorithm incurs the highest utility loss. The utility loss decreases significantly when considering the optimized algorithms. It is interesting that for different datasets different variants perform better. This suggests that the different algorithms provide different levers for adjusting fairness. Depending on the dataset one approach may be more applicable for preserving utility than another.

Qualitative Comparison. To provide some insight on the weights produced by the various algorithm, we visualize their output for \(\phi = 0.5\) in Figures 5 and 6. In the visualizations, red nodes are colored red, and blue nodes are colored blue. Their size depends on the value of the quantity we visualize.

For the Twitter and the books datasets, where the fraction of the weight of the protected group is close to \(\phi\), the fairness sensitive pagerank is very similar to the original one. For the blogs and especially for the DBLP2 datasets, where the fraction of the weight of the protected (red) group is much smaller, the fairness sensitive pagerank promotes red nodes. We also visualize the jump vector for the fairness sensitive pagerank. We observe that for the Twitter and the books dataset, where the algorithm is already “almost” fair, the jump vector assigns rather uniform weights, as the original Pagerank. For the other two datasets, it gives large values to a number of red nodes. This suggests an interesting line for future work: considering these nodes in link recommendation algorithms, since it seems that these nodes play a role in fairness.

The neighborhood locally fair pagerank algorithm produces different weights from the original Pagerank for all four datasets. In all cases, it promotes nodes connecting the two opposite groups, i.e., nodes that are minorities in their neighborhoods. This is more evident in the most homophilic networks, that is, in Twitter and books. Such nodes are also known as weak links and play an important role. They can also be useful in the context of recommendations, since research shows that it is more likely for such nodes to be accepted from the other side [17].

7 RELATED WORK

Algorithmic fairness. Recently, there has been increasing interest in algorithmic fairness, especially in the context of machine learning. Fairness is regarded as the lack of discrimination on the basis of some protective attribute. Various definition of fairness having proposed especially for classification [8, 11, 18, 20]. We use a group-fairness definition, based on parity. Approaches to handling fairness can be classified as pre-processing, that modify the input data, in-processing, that modify the algorithm and post-processing ones, that modify the output. We are mostly interested in in-processing techniques.

There is also prior work on fairness in ranking [3, 4, 27, 28]. All of these works consider ranking as an ordered list of items, and use different rules for defining and enforcing fairness that consider different prefixes of the ranking [27, 28], pair-wise orderings [3], or exposure and presentation bias [4, 23].

Our goal in this paper is not to propose a new definition of ranking fairness, but rather to initiate a study of fairness in link analysis. A distinguishing aspect of our approach is that we take into account the actual Pagerank weights of the nodes, not just their ranking. Furthermore, our focus in this paper is to design in-processing algorithms that incorporate fairness in the inner working of the Pagerank algorithm. We present a post-processing approach as a means to estimate a lower bound on the utility loss. None of the previous approaches considers ranking in networks, so the proposed approaches are novel.

Fairness in networks. There has been some recent work on network fairness in the context of graph embeddings [5, 21]. The work in [21] extends the node2vec graph embedding method by modifying the random walks used in node2vec with fair walks, where nodes are partitioned into groups and each group is given the same probability of being selected when a node makes a transition.

There are also previous studies on the effect of homophily, preferential attachment and differences in group sizes. It was shown that the combination of these three factors leads to uneven degree distributions between groups [2]. Recent work shows that this phenomenon is exaggerated by many link recommendation algorithms [24]. Evidence of inequality between degree distribution of minorities and majorities was also found in many real networks [14]. Our work extends this line of research by looking at Pagerank values instead of degrees.
Finally, there is previous work on diversity in network ranking. In this line of research, the goal is to find important nodes that also maximally cover the nodes in the network [19, 30]. Our problem is fundamentally different, since we look for rankings that follow a parity constraint.

8 CONCLUSIONS
In this paper we initiate a study of fairness for link analysis algorithms. We give general definitions of fairness, and we focus on fair algorithms for the Pagerank algorithm. We considered two approaches, one that modifies the jump vector, and one that imposes a fair behavior per node. We also consider the problem of attaining fairness while minimizing the utility loss of Pagerank. Our experiments demonstrate the behavior of our different algorithms.

REFERENCES
[1] L. A. Adamic and N. S. Glance. 2005. The political blogsphere and the 2004 U.S. election: divided they blog. In LinkKDD.
[2] C. Avin, B. Keller, Z. Lotker, C. Mathieu, D. Peleg, and Y. A. Pinolet. 2015. Homophily and the Glass Ceiling Effect in Social Networks. In ITCS 41–50.
[3] A. Beutel, J. Chen, T. Doshi, H. Qian, L. Wei, Y. Wu, L. Heldt, Z. Zhao, L. Hong, E. H. Chi, and C. Goodrow. 2019. Fairness in Recommendation Ranking through Pairwise Comparisons. In KDD. 2212–2220.
[4] A. J. Biega, K. P. Gummadi, and G. Weikum. 2018. Equity of Attention: Amortizing Individual Fairness in Rankings. In SIGIR. 405–414.
[5] A. Joey Bose and W. Hamilton. 2019. Compositional Fairness Constraints for Graph Embeddings. In ICML. 715–724.
[6] S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems 30, 1 (1998), 107 – 117.
[7] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri. 2007. Know Your Neighbors: Web Spam Detection Using the Web Topology. In SIGIR.
[8] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. S. Zemel. 2012. Fairness through awareness. In Innovations in Theoretical Computer Science. 214–226.
[9] D. A. Easley and J. M. Kleinberg. 2010. Networks, Crowds, and Markets. Cambridge University Press.
[10] M. Feldman, S. A. Friedler, J. M. M. Feldman, S. Scheidegger, and S. Venkatasubrahmanian. 2015. Certifying and Removing Disparate Impact. In KDD. 259–268.
Figure 5: Visualization for $\phi = 0.5$

Figure 6: Visualization for $\phi = 0.5$

[1] S.e A. Friedler, C. Scheidegger, S. Venkatasubramanian, S. Choudhary, E. P. Hamilton, and D. Roth. 2019. A comparative study of fairness-enhancing interventions in machine learning. In FAT’17 329–338.
[12] D. F. Gleich. 2015. PageRank Beyond the Web. SIAM Rev. 57, 3 (2015), 321–363.
[13] T. H. Haveliwala. 2002. Topic-Sensitive PageRank. In WWW.
[14] F. Karimi, M. Gnois, C. Wagner, P. Singer, and M. Strohmaier. 2018. Homophily influences ranking of minorities in social networks. Nature Scientific Reports 8 (2018).

[15] J. M. Kleinberg. 1999. Authoritative Sources in a Hyperlinked Environment. J. ACM 46, 5 (1999), 604–632.
[16] R. Lempel and S. Moran. 2001. SALSA: The Stochastic Approach for Link-Structure Analysis. ACM Trans. Inf. Syst. 19, 2 (2001), 131–160.
[17] Q. V. Liao and W. Fu. 2014. Can you hear me now?: mitigating the echo chamber effect by source position indicators. In CSCW. 184–196.
[18] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. 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
Q. Mei, J. Guo, and D. R. Radev. 2010. DivRank: the interplay of prestige and diversity in information networks. In KDD. 1009–1018.

A. Narayanan. 2018. Fairness Definitions and Their Politics (tutorial). In FAT.

T. A. Rahman, B. Surma, M. Backes, and Y. Zhang. 2019. Fairwalk: Towards Fair Graph Embedding. In IJCAI. 3289–3295.

Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In AAAI. http://networkrepository.com

A. Singh and T. Joachims. 2018. Fairness of Exposure in Rankings. In KDD.

A. Stoica, C. J. Riederer, and A. Chaintreau. 2018. Algorithmic Glass Ceiling in Social Networks: The effects of social recommendations on network diversity. In WebConf. 923–932.

L. Takac and Michal Zbovsk. 2012. Data Analysis in Public Social Networks. In International Scientific Conference and International Workshop Present Day Trends of Innovations, Lomza, Poland.

J. Tang, J. Zhang, L. Yao, J. Li. L. Zhang, and Z. Su. 2008. ArnetMiner: extraction and mining of academic social networks. In KDD. 990–998.

K. Yang and J. Stoyanovich. 2017. Measuring Fairness in Ranked Outputs. In SSDM.

M. Zehlike, F. Bonchi, C. Castillo, S. Hajian, M. Megahed, and R. Baeza-Yates. 2017. FA'IR: A Fair Top-k Ranking Algorithm. In CIKM.

Y. Zhang, J. Tang, Z. Yang, J. Pei, and P. S. Yu. 2015. COSNET: Connecting Heterogeneous Social Networks with Local and Global Consistency. In KDD. 1485–1494.

X. Zhu, A. B. Goldberg, J. Van Gael, and D. Andrzejewski. 2007. Improving Diversity in Ranking using Absorbing Random Walks. In HLT-NAACL. 97–104.
APPENDIX

8.1 Proofs
When \( \phi = r \), the average weight of red nodes is equal with the average weight of the blue nodes, i.e., \( \frac{\sum_{i \in B} w_i}{|B|} = \frac{\sum_{i \in R} w_i}{|R|} \).

**Proof.** It holds:
\[
\frac{1}{r} \frac{\sum_{i \in B} w_i - \sum_{i \in R} w_i}{|N|} = \frac{N}{|R|} \frac{\sum_{i \in B} w_i - \sum_{i \in R} w_i - \sum_{i \in B} w_i - \sum_{i \in R} w_i}{|N|}.
\]

The LFPFR\(_N\) algorithm is a special case of the residual pagerank algorithm.

**Proof.** For the LFPFR\(_N\) algorithm, we define the matrices \( X \) and \( Y \) that determine the residual policy as:
\[
X_N[i,j] = \begin{cases} \frac{1}{\text{out}(j)} & \text{if } i \in R, j \in L_R, \text{ and } (j,i) \in E \\ 0 & \text{otherwise} \end{cases}
\]
\[
Y_N[i,j] = \begin{cases} \frac{1}{\text{out}(j)} & \text{if } i \in B, j \in L_B, \text{ and } (j,i) \in E \\ 0 & \text{otherwise} \end{cases}
\]

From the transition matrix, each node \( i \in L_R \) gives a portion \( \frac{1-\phi}{\text{out}(i)} \) of each of its pagerank to its neighbors. The blue neighbors do not get any residual pagerank, thus they get an \( 1-\phi \) portion as in the LFPFR\(_N\) algorithm. Each of the red neighbors gets an additional \( \frac{1}{\text{out}(i)} \beta(i) = \frac{1}{\text{out}(i)} (\phi - (1-\phi)\text{out}(i)) \), which sums to \( \phi \). Thus, the red nodes get an \( \phi \) portion as in the LFPFR\(_N\) algorithm.

8.2 A lower bound for the optimal weight redistribution for targeted fairness

Given the weight vector \( w \) and the set \( S \), we divide the full set of nodes in three categories: the set \( B_S \) of blue nodes in \( S \), the set \( R_S \) of red nodes in \( S \), and the rest of nodes \( O \) not in \( S \). In order for \( f \) to be fair, it must be that it moves weight between these categories. Furthermore, this movement is always in one direction, e.g., all nodes in \( R_S \) will increase their weight. It is clearly suboptimal to increase the weight of some nodes in \( R_S \) and decrease the weight of others. We define the variables \( x_B = \sum_{i \in B_S} (f_i - w_i) \), \( x_R = \sum_{i \in R_S} (f_i - w_i) \), and \( x_O = \sum_{i \in O} (f_i - w_i) \) to be the total change in weight for the nodes in \( B_S, R_S \), and \( O \) respectively. Note that these values may be positive, indicating an increase in weight for the respective category, or negative, indicating a decrease in weight for the respective category. It holds:

\[
x_B + x_R + x_O = 0 \tag{4}
\]

Let \( f_R \) and \( f_B \) the weight allocated to nodes in \( R_S \) and \( R_B \) respectively, and \( \rho \) and \( \beta \) their desired values according to \( \phi \). Also, let \( w_R \) and \( w_B \) be the weight of the nodes in \( B_S \) and \( R_S \) respectively. Since the vector \( f \) is fair for the nodes in \( S \) it holds that:

\[
\frac{w_R + x_R}{w_B + x_B} = \frac{\rho}{\beta} \tag{5}
\]

Using Equations 4 and 5, we can express \( x_R \) and \( x_R \) as a function of \( x_O \):

\[
x_R = \rho w_B - \beta w_R - \rho x_O \tag{6}
\]
\[
x_B = \beta w_R - \rho w_B - \rho x_O \tag{7}
\]

Now, let \( N_B, N_R, \) and \( N_O \) denote the number of nodes in categories \( B_S, R_S \), and \( O \) respectively. To minimize loss, and since we allow \( f \) to have negative entries, the change in weight must be distributed equally in each category. Thus, the total loss is:

\[
\text{Loss}(f, w) = \frac{x_R^2}{N_R} + \frac{x_B^2}{N_B} + \frac{x_O^2}{N_O} \tag{8}
\]

We substitute Equations 7 and 8 in Equation 8, we take the derivative with respect to \( x_O \), and we set it zero. Solving for \( x_O \), we get:

\[
x_O = \frac{N_B(\beta w_R - \rho w_B)(\beta N_R - \rho N_B)}{\rho N_B(\rho N_O + N_R) + \beta N_R(\beta N_O + N_R)} \tag{9}
\]

Substituting \( x_O \) in Equations 7 and 8, we obtain:

\[
x_R = \frac{(\rho w_R - \beta w_B)N_B(\beta N_O + N_R)}{\rho N_B(\rho N_O + N_R) + \beta N_R(\beta N_O + N_R)} \tag{10}
\]
\[
x_B = \frac{(\rho w_R - \beta w_B)N_B(\rho N_O + N_R)}{\rho N_B(\rho N_O + N_R) + \beta N_R(\beta N_O + N_R)} \tag{11}
\]

There are some interesting observations in these equations. First, a factor that appears in all equations is \( \beta w_R - \rho w_B \), which tells us how unfair the original weights are. For example, if \( \beta w_R - \rho w_B < 0 \), then we are unfair towards category \( R \). In this case the nodes in category \( R \) will always receive weight \( w_R > 0 \). The origin of the weight depends on the ratio \( N_R/N_B \) of the nodes in \( S \). If \( \beta N_R - \rho N_B < 0 \), then we have proportionally more nodes of \( B \) in \( S \) with an excess of weight. In this case we remove weight only from the nodes in \( B \), and we distribute it to the nodes in \( R \) and \( O \) as defined by Equations 11 and 9. If \( \beta N_R - \rho N_B < 0 \), then we have proportionally less nodes of \( B \) in \( S \), but they have proportionally more weight. In this case we remove weight from both the nodes in \( B \), and \( O \), as defined by Equations 11 and 9, and we distribute it to the nodes in \( R \). If \( \beta N_R - \rho N_B = 0 \), then we take weight only from the nodes in \( B \) and give only to the nodes in \( R \).

Having computed the values for \( x_R, x_B \), and \( x_O \), we can now compute the loss using Equation 8. Note that this is a lower bound to the optimal loss for our problem, since it does not guarantee that the resulting vector \( f \) has non-negative entries.

8.3 Reproducibility

Code and datasets will be available at github in the following link: https://anonymous.4open.science/r/13016d8f-3516-497e-9788-d0bb06150b51/

8.4 Additional datasets and experiments

In Table 3, we present statistics for additional datasets.

- **POKEC [25]:** This is a Slovak social network. Nodes correspond to users, and links to friendships. Friendship relations are directed.
- **DBLP1:** An author collaboration network constructed by the Arnetminer academic search system [26] using publication data from dblp. Two authors are connected if they have co-authored an article.
Table 3: Real dataset characteristics. $r$, $b$ relative size of protected and unprotected group, respectively; $PR$, $PB$ pagerank assigned to the red and blue group respectively.

| Dataset | #nodes  | #edges  | Protected attribute | $r$  | $b$  | homophily | $PR$ | $PB$ |
|---------|---------|---------|---------------------|------|------|-----------|------|------|
| POKEC   | 1,632,803 | 30,622,564 | gender (women)      | 0.51 | 0.49 | 1.11      | 0.54 | 0.46 |
| DBLP1   | 423,469  | 2,462,422  | gender (women)      | 0.19 | 0.81 | 0.83      | 0.13 | 0.87 |
| LINKEDIN| 3,209,448 | 13,016,453 | gender (women)      | 0.37 | 0.63 | 0.72      | 0.37 | 0.63 |
| PHYSICS | 30,359   | 347,235   | year (after 1997)   | 0.66 | 0.34 | 0.76      | 0.39 | 0.61 |

• LINKEDIN [29]: Nodes correspond to LinkedIn profiles. Two profiles are linked if they were co-viewed by the same user.

• PHYSICS: This is the Arxiv HEP-PH (high energy physics phenomenology) citation graph from the SNAP dataset³. Nodes correspond to papers and there is an edge from a paper to another, if the first paper cites the second one.

Again, there are cases where the fraction of the weight assigned to the protected group is even smaller than $r$.

In Figure 7, we report results for the locally fair pagerank algorithms for $\phi = r$, while in Figure 8, we report results for the targeted locally fair pagerank algorithms for $\phi = 0.5$. For comparison, we also report the optimal redistribution using the post-processing algorithm.

³http://snap.stanford.edu/data