Minimizing the Cost of Pagerank Fairness

Spyridon Tzimas
University of Ioannina
Ioannina, Greece
s.tzimas@uoi.gr

Themistoklis Nikas
Boston University
Boston, MA, USA
tnikas@bu.edu

Evaggelia Pitoura
University of Ioannina
Ioannina, Greece
Archimedes/Athena RC
Athens, Greece
pitoura@uoi.gr

Panayiotis Tsaparas
University of Ioannina
Ioannina, Greece
Archimedes/Athena RC
Athens, Greece
tsap@uoi.gr

Abstract

Recently, there has been a surge of research activity in the field of *Algorithmic Fairness*, which aims to model and ensure the fairness of algorithms, including *network algorithms*. In this work, we focus on the celebrated Pagerank algorithm for assessing the importance of nodes in a network. Pagerank fairness was first studied by Tsioutsiouliklis et al. [28], who proposed the Locally Fair Pagerank algorithm that achieves group fairness by enforcing a fair behavior for all nodes in the graph. However, local Pagerank fairness comes at a high *cost*, since it modifies all the nodes in the graph and alters significantly the original Pagerank values, incurring a significant utility loss. We consider the problem of minimizing the cost of Pagerank fairness. Specifically, we aim to identify a set of nodes to enforce a fair behavior for achieving group fairness, while minimizing the cost, measured as either the cardinality of the set or the utility loss. We derive analytical expressions for estimating the fairness gain and the utility loss of modifying individual nodes, and we propose greedy and heuristic algorithms for selecting these nodes efficiently. Experiments on real and synthetic datasets demonstrate that our approach can achieve Pagerank fairness at a low cost.

Keywords

Pagerank, Fairness, Local Fairness, Random Walks, Fairness Cost

ACM Reference Format:
Spyridon Tzimas, Themistoklis Nikas, Evaggelia Pitoura, and Panayiotis Tsaparas. 2026. Minimizing the Cost of Pagerank Fairness. In *The Nineteenth ACM International Conference on Web Search and Data Mining (WSDM Companion '26), February 22–26, 2026, Boise, ID, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3779211.3793177

1 Introduction

We live in a world where we increasingly rely on algorithms to make decisions that affect our lives, trivial or consequential, at individual or organizational level. This increased dependence on algorithms has raised concerns about possible biases of these algorithms towards specific groups and minorities. These concerns gave birth to the research field of Algorithmic Fairness [5], which studies the fairness of algorithm output and aims to design algorithms with formal fairness guarantees. Algorithmic fairness originally focused on classification algorithms, but has recently expanded to other types of algorithms, including network algorithms [11].

Networks are important data models of complex systems of interacting entities, and network algorithms are used for a variety of tasks, such as recommendations, diffusion of information, and entity ranking. In this work, we focus on the celebrated Pagerank algorithm [8] for assessing the importance of nodes in a network. The algorithm performs a random walk on the network graph and uses the stationary distribution of the random walk to determine the importance of the nodes.

Pagerank fairness was first studied by Tsioutsiouliklis et al. [28]. They assumed that the nodes in the network are partitioned into groups and asked for a fair allocation of Pagerank values between the different groups. They proposed the *Locally Fair Pagerank* class of algorithms, which achieves fairness by making the transitions out of each node in the random walk fair. This locally fair behavior of the nodes results in overall fairness of the algorithm.

Local Pagerank Fairness is a simple and intuitive notion of fairness, but it is also particularly invasive, since it modifies the behavior of all nodes in the network, and alters significantly the original Pagerank vector. In this paper, we aim to find a set of nodes to modify so as to achieve fairness at a minimum *cost*. We consider two notions of cost. The first is the size of the set, aiming to minimize the changes in the transition matrix of the random walk. The second is the change in the Pagerank values of all nodes, aiming to minimize the *utility loss* in the resulting fair Pagerank vector.

We derive analytical expressions for estimating efficiently the gain in fairness and the loss in utility when making the transitions out of a node fair. Our formulas rely on the fact that this operation is a rank-1 perturbation of the transition matrix, and thus we can compute the change in Pagerank without recomputing the Pagerank vector. Using our formulas, we propose greedy algorithms for selecting the nodes to modify.

We experiment on real and synthetic datasets and demonstrate that our algorithms are able to achieve fairness at a low cost. We also compare with efficient heuristics and demonstrate that we can obtain a good performance with more efficient algorithms.

WSDM Companion '26, Boise, ID, USA

ACM ISBN 979-8-4007-2358-2/2026/02
https://doi.org/10.1145/3779211.3793177

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

- We define the novel problem of Minimum Cost Local Pagerank Fairness, where we ask for Pagerank fairness, while minimizing the cost of local changes.
- We derive analytical formulas for estimating the fairness gain and the utility loss change when modifying a node, and we use the formulas to design greedy algorithms and efficient heuristics.
- We perform extensive experiments on real and synthetic data that demonstrate the properties of our algorithms in different settings.

The remainder of the paper is structured as follows. In Section 2 we review the related work. In Section 3 we provide the necessary background and define the problem. In Section 4 we derive the formulas for fairness gain and utility loss when modifying a node. Section 5 presents our algorithms and Section 6 the experimental evaluation, while Section 7 concludes the paper.

2 Related Work

Algorithmic Fairness aims to model fairness and produce algorithms with fairness guarantees. There is substantial work in this area, mainly for Machine Learning algorithms [5, 12]. Recently, the work in Algorithmic Fairness has expanded to other areas, including network analysis [11, 16]. Network fairness studies the fairness of network structures and algorithms. Our work falls within this area.

In this work, we consider the fairness of the Pagerank algorithm. Our work builds upon the work in [28], where the authors introduced Pagerank fairness and provided fair Pagerank algorithms. They defined Locally Fair Pagerank, where fairness is achieved by local modifications in the transitions out of all nodes in the graph. We consider the problem of minimizing the cost of achieving fairness, where cost is measured as the number of local changes or the deviation from the original Pagerank values. Utility loss was also considered in [28], but they study it in a different context. The combinatorial problem we consider is novel and requires new techniques.

Follow-up work [27] considered the problem of achieving fairness by inserting links in the graph via recommendations. They also want to minimize the number of added links, but the problem they consider is different from ours.

The fairness of network centrality has been considered in different contexts. In [3, 18, 25], they study the different factors (group imbalance, homophily, recommendations) that may lead to degree centrality unfairness. These factors are also studied for more sophisticated link analysis algorithms such as Pagerank [26, 28], SALSA [13], and HITS [26]. We also study how homophily affects our algorithms.

Random walks have also been considered in graph embedding algorithms such as node2vec [15]. In [23], they define a fair random walk for the node2vec algorithm, which is similar to the random walk of Locally Fair Pagerank. They do not consider the problem of making selective adjustments to the nodes, which would be an interesting application of our work.

Another line of work considers fairness in ranking [6, 7, 19, 24, 29, 30] (see [22] for a survey). Here, we have a scoring function that produces a ranked list of items and we want this list to be fair. The work in [19] considers fairness in network ranking algorithms. This work does not consider local modifications of nodes, so it is orthogonal to ours. It would be interesting to consider applications of our approach to this problem.

Related to fairness is diversity, where we want to have a diverse set of nodes in the top ranks. The notion of diversity has been studied for Pagerank [21, 31]. The goal is to assign high scores to nodes that "cover" different parts of the network. There is an incremental node selection process in the algorithms that resembles our approach, but our algorithms have a different objective.

Finally, there is work in optimizing the Pagerank value of a node, or a collection of nodes, by changing the graph structure around the node, usually with the addition or removal of edges [4, 9, 10, 17]. This work is related to ours, but it does not consider the objective of fairness or modifications that enforce a fair behavior of the nodes.

3 Definitions

In this section, we present the necessary background for the Pagerank algorithm and the definition of Pagerank fairness from [28].

3.1 The Pagerank Algorithm

The Pagerank algorithm [8] pioneered the use of graph structure in assessing the importance of nodes in a graph. It was made popular through its use in the Google search engine, but it has also found several applications in other areas.

The algorithm takes a (directed) graph $G = (V, E)$ as input and produces a weight vector $\mathbf{p}$, where $p(i)$ is the weight assigned to node i. The weight vector is computed by performing a *random walk with restarts* on the graph. At each step, with probability $(1-\gamma)$ the random walk follows one of the outgoing edges of the current node uniformly at random, while with probability γ it jumps to a node chosen according to the distribution given as the jump vector $\mathbf{u}$ (usually the uniform vector). The Pagerank vector $\mathbf{p}$ is the stationary distribution of the random walk.

More formally, let P denote the transition probability matrix of the random walk when following an outgoing link. Let d_i be the out-degree of node i. Then we have $P[i, j] = 1/d_i$ if $(i, j) \in E$, and $P[i, j] = 0$ otherwise. For the stationary distribution of the random walk, it holds that

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

Solving for $\mathbf{p}^T$, we get

$$\mathbf{p}^T = \gamma \mathbf{u}^T \left(I - (1-\gamma)P\right)^{-1},$$

where I is the identity matrix. Let $Q = \gamma \left(I - (1-\gamma)P\right)^{-1}$. Then we have $\mathbf{p}^T = \mathbf{u}^T Q$. The value $p(i)$ is the Pagerank value of node i.

3.2 Pagerank Fairness

The fairness of the Pagerank algorithm was defined and studied in [28]. The work considers a group fairness definition. Given a graph $G = (V, E)$, they assume that the nodes in the graph are partitioned into groups, based on a sensitive attribute such as gender or race. Following [28], for simplicity, we assume that there are two groups of nodes, Red (R) and Blue (B), such that $R, B \subseteq V$, $R \cap B = \emptyset$ and $R \cup B = V$. We will refer to (R, B) as the group partition. We also assume that the red group is the *protected* group, for which we

want to guarantee fair treatment. Abusing the notation, let $p(R)$ denote the total Pagerank weight (probability mass) allocated to the red group. Given a parameter ϕ, the Pagerank algorithm is ϕ-fair, if $p(R) = \phi$. We will refer to $p(R)$ as the *red pagerank*. The *blue pagerank* is defined symmetrically.

To achieve fairness, Tsioutsiouliklis et al. [28], define the *Locally Fair Pagerank* class of algorithms, where they adjust the transition probability matrix of the random walk, so that all nodes in the graph transition with probability ϕ to a red node and with probability $1-\phi$ to a blue node. We can also think of each node as distributing their Pagerank weight to the two groups in a ϕ-fair manner. We say that the nodes are made to behave ϕ-fairly or simply that they are made ϕ-fair. The jump vector $\mathbf{u}$ is also made ϕ-fair as $u(i) = \phi/|R|$ if $i \in R$, and $u(i) = (1-\phi)/|B|$ if $i \in B$. They show that this local fair behavior results in global fairness of the stationary distribution.

3.3 Problem Definition

The locally fair algorithms require modifying the whole transition matrix P of the random walk so that the transitions out of the nodes are ϕ-fair (excluding only the nodes that are already ϕ-fair). Furthermore, the resulting Pagerank vector is considerably altered compared to the original. This is excessively intrusive, especially considering that for some nodes the transition probability to red nodes may already be greater than ϕ, so they do not need to be made ϕ-fair. We consider the problem of identifying the set of local changes that achieve fairness with the minimum effect.

Formally, for a subset $S \subseteq V$ of nodes, let $cost(S)$ be the cost of making the nodes in S ϕ-fair. Our goal is to find a set S of nodes such that making its nodes ϕ-fair suffices to achieve fairness and $cost(S)$ is minimized.

We consider two definitions of cost. The first considers the number of local changes that we need to achieve fairness. That is, $cost(S) = |S|$, which is the number of nodes that we make ϕ-fair. Formally, we define our problem as follows.

Problem 1 (Minimum Modification Local Fairness (MMLF)). *Given a graph $G = (V, E)$ with group partition (R, B) and a fairness parameter $\phi \in (0, 1)$, find the smallest subset $S \subseteq V$ of nodes such that after making its nodes ϕ-fair, for the resulting Pagerank vector $\mathbf{p}'$ it holds that $p'(R) \geq \phi$.*

The next definition of cost considers the change in the resulting Pagerank vector. Let $\mathbf{p}$ denote the original Pagerank vector and let $\mathbf{p}'$ denote the Pagerank vector after making a subset of nodes ϕ-fair. In [28], they define the *utility loss* of $\mathbf{p}'$ as $loss(\mathbf{p}') = \|\mathbf{p}' - \mathbf{p}\|_2^2$, that is, the sum of squares error between the two vectors. We set $cost(S) = loss(\mathbf{p}')$. Formally, we define our problem as follows.

Problem 2 (Minimum Loss Local Fairness (MLLF)). *Given a graph $G = (V, E)$ with group partition (R, B) and a fairness parameter $\phi \in (0, 1)$, find a subset $S \subseteq V$ of nodes such that after making its nodes ϕ-fair, for the resulting Pagerank vector $\mathbf{p}'$ it holds that $p'(R) \geq \phi$ and $loss(\mathbf{p}')$ is minimized.*

4 Single Node Modification

We now study the case where we modify a single node. We first describe the different ways we can make a node ϕ-fair, and then how to compute the effect of making a node ϕ-fair on the Pagerank vector and the red Pagerank value $p(R)$. We will utilize our derivations to design algorithms for the two problems.

Let us first introduce our matrix notation. For every matrix A, we use $A[i, j]$ to denote the value of its (i, j)-th cell, and we use $A[i,]$ and $A[, j]$ to denote its i-th row vector and j-th column vector respectively. Abusing the notation, we also define the following:

$$A[, J] = \sum_{j \in J} A[, j] \qquad A[i, J] = \sum_{j \in J} A[i, j]$$

4.1 Making a Node ϕ-fair

In [28], they defined different ways to make a node ϕ-fair. We will consider two approaches in our work. In what follows, we assume that we have as input a graph $G = (V, E)$, with n nodes and m edges, and a group partition (R, B). For a node x, we use d_x to denote the outgoing degree of x, and d_x^R and d_x^B to denote the outgoing degree of x to red and blue nodes respectively. We use $\rho_x = d_x^R/d_x$ to denote the fraction of the neighbors of x in the red group, and $\beta_x = d_x^B/d_x$ to denote the fraction of blue neighbors. We make ϕ-fair only nodes that are ϕ-unfair to the red group, that is, only nodes x for which $\rho_x < \phi$.

Neighborhood Locally Fair Pagerank: The first approach updates the transition matrix P, by redistributing the transition probability among the neighbors of the node, so that the transition probability to the red neighbors is ϕ. Formally, to make node x ϕ-fair, in the updated transition probability matrix P', we set the x-row as follows:

$$P'[x, y] = \begin{cases} \phi/d_x^R & \text{if } (x, y) \in E, y \in R \\ (1-\phi)/d_x^B & \text{if } (x, y) \in E, y \in B \\ 0 & \text{if } (x, y) \notin E \end{cases} \tag{1}$$

If node x has no red neighbors, then the node allocates probability ϕ uniformly at random to all nodes in the red group. Thus, we get:

$$P'[x, y] = \begin{cases} \phi/|R| & \text{if } y \in R \\ (1-\phi)/d_x & \text{if } (x, y) \in E \\ 0 & \text{if } (x, y) \notin E, y \in B \end{cases} \tag{2}$$

Residual Locally Fair Pagerank: The second approach allocates probability mass $(1-\delta_x)$ uniformly to the neighbors of node x, and δ_x uniformly to all nodes of the red group, where δ_x is such that we allocate probability mass ϕ to the red group. Let $\mathbf{u}_R$ denote the n-dimensional probability vector that allocates the probability mass uniformly among the red nodes, that is, $\mathbf{u}_R(i) = 1/|R|$ if $i \in R$, and $\mathbf{u}_R(i) = 0$ otherwise. Then we have:

$$P'[x,] = (1-\delta_x)P[x,] + \delta_x \mathbf{u}_R^T \tag{3}$$

That is, when transitioning out of node x, with probability δ_x we transition to a red node uniformly at random, and with probability $(1-\delta_x)$ we transition to a randomly selected neighbor. To ensure ϕ-fairness, it must hold that $(1-\delta_x)\rho_x + \delta_x = \phi$. Therefore:

$$\delta_x = \frac{\phi - \rho_x}{1 - \rho_x} = \frac{\phi - \rho_x}{\beta_x}$$

4.2 Computing the Effect of a Single Modification

We will now derive closed-form expressions for computing the effect on fairness and utility loss of making a single node x ϕ-fair. Our derivations are based on perturbation theory [14].

Making a node x ϕ-fair involves the modification of a single row of the transition matrix P to obtain a new transition matrix P'. We view this change as the addition of a perturbation matrix D to P, that is, $P' = P + D$. The matrix D is of a very specific form: it is zero everywhere except for its x-th row $D[x,]$. We will sometimes refer to the perturbation matrix D as the perturbation row vector $D[x,]$. The row vector $D[x,]$ depends on the type of modification we perform.

In what follows, let $\mathbf{e}_x$ denote the unit vector with 1 at x and zero everywhere else (on the x-axis of $\mathbb{R}^n$). Furthermore, for every set $X \subseteq \{1,\dots,n\}$, we define $\mathbf{e}_X = \sum_{x\in X} \mathbf{e}_x$. Note that $D = \mathbf{e}_x D[x,]$, so D is a rank-1 matrix. We can exploit perturbation theory [14] to prove the following Theorem for the updated matrix Q' after modifying node x.

Theorem 4.1. *Let $G = (V, E)$ be a graph, and let P be the transition probability matrix of the Pagerank random walk on G. After the modification of node $x \in V$, with perturbation row vector $D[x,]$, we can compute the updated matrix Q' as follows:*

$$Q' = Q + \frac{Q[,x]D[x,]Q}{\frac{\gamma}{1-\gamma} - D[x,]Q[,x]} \tag{4}$$

Proof. For the matrices Q and Q', we have $Q = \gamma(I-(1-\gamma)P)^{-1}$ and $Q' = \gamma(I-(1-\gamma)P')^{-1} = \gamma(I-(1-\gamma)P-(1-\gamma)D)^{-1}$ where $D = \mathbf{e}_x D[x,]$ by definition. The Sherman–Morrison formula [14] states that for an invertible matrix $A \in \mathbb{R}^{n\times n}$ and two vectors $u, v \in \mathbb{R}^n$, the matrix $A + uv^T$ is invertible if and only if $1 + v^T A^{-1} u \neq 0$, and it holds that

$$(A+uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1+v^TA^{-1}u}.$$

Applying this formula for $A = I-(1-\gamma)P$, $u = -(1-\gamma)\mathbf{e}_x$, $v^T = D[x,]$, and using the fact that $Q = \gamma A^{-1}$, we obtain:

$$\begin{aligned} Q' &= \gamma(A+uv^T)^{-1} \\ &= \gamma A^{-1} - \frac{\gamma A^{-1}uv^TA^{-1}}{1+v^TA^{-1}u} \\ &= Q - \frac{-(1-\gamma)Q\mathbf{e}_x D[x,]\gamma^{-1}Q}{1-(1-\gamma)D[x,]\gamma^{-1}Q\mathbf{e}_x} \\ &= Q + \frac{Q[,x]D[x,]Q}{\frac{\gamma}{1-\gamma} - D[x,]Q[,x]} \end{aligned}$$

□

Using Eq. (4) and the fact that $\mathbf{p}^T = \mathbf{u}^T Q$ and $p(x) = \mathbf{u}^T Q[,x]$, we can compute the updated Pagerank vector $\mathbf{p}^T$ after modifying node x as follows:

$$(\mathbf{p}')^T = \mathbf{p}^T + p(x)\frac{D[x,]Q}{\zeta_x}, \tag{5}$$

where $\zeta_x = \frac{\gamma}{1-\gamma} - D[x,]Q[,x]$. Using Eq. (5) and the fact that $p(R) = \mathbf{p}^T\mathbf{e}_R$, the updated red Pagerank after modifying node x is computed as follows:

$$p'(R) = p(R) + p(x)\frac{D[x,]Q[,R]}{\zeta_x}$$

We define the *fairness gain* of $\mathbf{p}'$ as $gain(\mathbf{p}') = p'(R) - p(R)$, that is, the change in red Pagerank after modifying node x. We have:

$$gain(\mathbf{p}') = p(x)\frac{D[x,]Q[,R]}{\zeta_x} \tag{6}$$

We can also compute the utility loss of $\mathbf{p}'$ with respect to vector $\mathbf{p}$, as follows:

$$loss(\mathbf{p}' \mid \mathbf{p}) = p^2(x)\frac{\|D[x,]Q\|_2^2}{\zeta_x^2} \tag{7}$$

4.3 The Perturbation Row Vector $D[x,]$

The row vector $D[x,]$ depends on the kind of modification we perform. We will now define $D[x,]$ for the different modifications we defined above. In what follows, for a node x, if N_x is the (outgoing) neighborhood of x, we use $R_x \subseteq N_x$ to denote the set of red neighbors of node x and $B_x \subseteq N_x$ to denote the set of blue neighbors of node x. Recall that $\rho_x = |R_x|/|N_x|$ is the fraction of red neighbors of x, while $\beta_x = |B_x|/|N_x|$ is the fraction of blue neighbors of x. We assume that the node we modify is ϕ-unfair, that is $\rho_x < \phi$ and $\beta_x > 1-\phi$ for the input fairness parameter ϕ.

Neighborhood Locally Fair Pagerank: In this case, for a node x that has red neighbors ($R_x \neq \emptyset$), from Eq. (1), it follows that the perturbation row vector $D[x,]$ is

$$D[x,] = \frac{1}{d_x}\left(\left(\frac{\phi}{\rho_x} - 1\right)\mathbf{e}_{R_x}^T + \left(\frac{1-\phi}{\beta_x} - 1\right)\mathbf{e}_{B_x}^T\right).$$

In simple terms, we add to the red neighbors of x ($e_{R_x}(i) = 1$) transition probability $(\phi - \rho_x)/d_x^R$, while we subtract from the blue neighbors ($e_{B_x}(i) = 1$) transition probability $(\beta_x - (1-\phi))/d_x^B = (\phi - \rho_x)/d_x^B$.

For a node x that has no red neighbors ($R_x = \emptyset$, so $N_x = B_x$), from Eq. (2), the perturbation row vector $D[x,]$ becomes

$$D[x,] = \frac{\phi}{|R|}\mathbf{e}_R^T - \frac{\phi}{d_x}\mathbf{e}_{N_x}^T.$$

Residual Locally Fair Pagerank: In this case, from Eq. (3), it follows directly that

$$D[x,] = \frac{\delta_x}{|R|}\mathbf{e}_R^T - \frac{\delta_x}{d_x}\mathbf{e}_{N_x}^T.$$

Using the appropriate $D[x,]$ we can compute the updated matrix Q', the updated vector $\mathbf{p}'$, and the quantities $gain(\mathbf{p}')$ and $loss(\mathbf{p}')$, using the formulas in Section 4.2.

4.4 Understanding the Fairness Gain

The formulas in Section 4.2 do not provide sufficient intuition on what is a good node to make ϕ-fair. To obtain such intuition, we expand the formula in Eq. (6) for the fairness gain. Performing the computations for the Neighborhood Locally Fair case, for a node

Algorithm 1 GREEDY-f Algorithm

Input: Graph $G = (V, E)$, group partition (R, B), target fairness ϕ, selection function f.
Output: The set $S \subseteq V$ of nodes to be made ϕ-fair.

Compute the set of candidate nodes $C \subseteq V$ that are ϕ-unfair.
Compute initial matrix Q and Pagerank vector $\mathbf{p}$.
$S = \emptyset$
while $p(R) < \phi$ **do**
  $x = \arg\max_{x \in C} f(x)$
  $S = S \cup \{x\}$
  $C = C \setminus \{x\}$
  Compute updated matrix Q
  Compute updated Pagerank vector $\mathbf{p}$
end while
return S

with a non-empty set of red neighbors, we obtain:

$$gain(\mathbf{p}') = p(x) \frac{\frac{\phi-\rho_x}{d_x}\left(\frac{1}{\rho_x}\sum_{z\in R_x} Q[z,R] - \frac{1}{\beta_x}\sum_{z\in B_x} Q[z,R]\right)}{\frac{\gamma}{1-\gamma} - \frac{\phi-\rho_x}{d_x}\left(\frac{1}{\rho_x}\sum_{z\in R_x} Q[z,x] - \frac{1}{\beta_x}\sum_{z\in B_x} Q[z,x]\right)} \quad (8)$$

The value $Q[i, j]$ is the personalized Pagerank that node i assigns to node j. The personalized Pagerank vector of node i is computed by performing the Pagerank random walk with jump (restart) vector $\mathbf{u} = \mathbf{e}_i$. That is, the random walk always restarts from node i. Given that $\mathbf{p}^T = \mathbf{u}^T Q$, it follows that the row $Q[i,]$ is the transposed personalized Pagerank vector of node i. Intuitively, the value $Q[i, j]$ captures how "close" node j is to node i. The value $Q[i, R]$ is the total personalized Pagerank that node i assigns to the red nodes.

Eq. (8) characterizes the nodes that contribute to the increase of $p(R)$ when made ϕ-fair. A node is a good selection if it has high Pagerank $p(x)$, so that it has a lot of weight to redistribute. The numerator suggests that the red neighbors in R_x should be close to other red nodes (high $Q[z, R]$), while blue neighbors in B_x should be far from red nodes (low $Q[z, R]$), so that the redistributed weight ends up in red nodes down the line. The denominator suggests that the red neighbors of x should have higher probability of returning to x than the blue ones, so that by redistributing the transition probability mass, we strengthen the Pagerank of node x.

5 Algorithms

We now present our algorithms for solving the MMLF and MLLF problems. Our algorithms follow the general outline of Algorithm 1. The algorithms take as input the graph G, the partition (R, B), the target parameter ϕ, and a function f that determines the selection criterion, depending on the problem we consider. They output the set S of the nodes that are made ϕ-fair.

The algorithms operate in a greedy fashion, building the solution set incrementally, each time adding the best candidate node to the set S, according to the selection function f. In what follows, let $\mathbf{p}^x$ denote the updated Pagerank vector after making node x ϕ-fair. We consider the following three selection functions that result in three different greedy algorithms:

- GREEDYGAIN: $f(x) = gain(\mathbf{p}^x)$, the increase in red Pagerank. This algorithm targets the MMLF problem and tries to find the smallest set of nodes that achieves fairness.
- GREEDYLOSS: $f(x) = -\delta loss(\mathbf{p}^x)$, where $\delta loss(\mathbf{p}^x) = loss(\mathbf{p}^x) - loss(\mathbf{p})$, the increase in utility loss. This algorithm targets the MLLF problem and tries to find the set of nodes with the minimum utility loss that achieves fairness.
- GREEDYRATIO: $f(x) = gain(\mathbf{p}^x)/\delta loss(\mathbf{p}^x)$, the ratio of fairness gain over utility loss increase. This algorithm tries to strike a balance between the goals of the MMLF and MLLF problems and find nodes that give high fairness gain with low utility loss.

For each greedy algorithm, we have two variants depending on the local fairness approach we consider (neighborhood local fairness, or residual local fairness).

5.1 Computational Complexity

A naive implementation of our algorithms would compute the updated matrix Q' after each node modification via matrix inversion, which takes $O(n^\omega)$ time[1]. We propose to compute Q' via Eq. (4) instead. Regarding its complexity, we observe the following:

- Computing $D[x,]Q$ takes $O(n^2)$ time as the product of a row vector and a square matrix in this order.
- Computing $Q[, x](D[x,]Q)$ takes $O(n^2)$ time as the product of column vector and a row vector in this order.
- Computing $D[x,]Q[, x]$ takes $O(n)$ time as the product of a row vector and a column vector in this order.

We conclude that computing Q' takes $O(n^2)$ time in total. This is an improvement over the $O(n^\omega)$ time that it takes to compute Q' via matrix inversion. We can also compute the updated Pagerank vector $\mathbf{p}'$ via Eq. (5) without increasing the complexity.

In every iteration of our algorithms, we select the next node to modify to be the one that maximizes the selection function f, so we must compute $f(x)$ for all candidate nodes x. Regarding the complexity of computing $gain(\mathbf{p}^x)$ for all x, we observe that we can compute $Q[, R]$ once in $O(n^2)$ time, and that computing $D[x,]Q[, R]$ or $D[x,]Q[, x]$ takes $O(n)$ time for each x, because it is the product of a row vector and a column vector in this order. We conclude that computing $gain(\mathbf{p}^x)$ for all x takes $O(n^2)$ time in total. For computing $\delta loss(\mathbf{p}^x)$ for all x, we observe that the bottleneck in the computation is the computation of $D[x,]Q$ for all x. These row vectors can be computed simultaneously by considering the matrix $\mathcal{D}$ whose x-th row equals $D[x,]$ and computing $\mathcal{D}Q$ in $O(n^\omega)$ time. Therefore, the total running time of our algorithms is $O(tn^\omega)$ if the selection function f incorporates $\delta loss$, and $O(tn^2)$ otherwise, where t denotes the number of iterations.

6 Experimental Evaluation

In this section, we evaluate experimentally our algorithms for the MMLF and MLLF problems. The goals of the experiments are the following:

[1]The number ω is defined as the smallest known value such that the complexity of matrix multiplication, and consequently matrix inversion, is $O(n^\omega)$. At the time of writing of this paper, this value is 2.371339 [2].

Table 1: Real dataset characteristics. r denotes the relative size of the red group, ρ_R and ρ_B denote the average percentage of homophilous (intra-group) edges adjacent to a node of the respective groups and $p(R)$ denotes the original Pagerank weight assigned in total to the red group.

Dataset	$\|V\|$	$\|E\|$	Groups (Red)	r	ρ_R	ρ_B	$p(R)$
Books	92	374	political (left)	0.47	0.96	0.96	0.4969
Blogs	1,222	16,714	political (right)	0.48	0.88	0.92	0.4711
Facebook	4,039	88,234	gender (female)	0.38	0.42	0.68	0.3685

- Evaluate the effectiveness of our proposed algorithms in achieving fairness on real and synthetic datasets.
- Understand the effect of network structure, in particular network homophily, on our difficulty in achieving fairness.
- Compare with efficient heuristics in terms of cost and time efficiency.

All real and synthetic datasets, algorithm implementations, and experimental results are publicly available online[2].

6.1 Datasets

6.1.1 Real Datasets. For our experiments, we considered the following real datasets.

Books is a network of political books, where an edge between two books indicates co-purchase[3].

Blogs is a network of political blogs, where an edge between two blogs indicates the existence of a hyperlink between them in either direction [1].

Facebook is a network of *Facebook* profiles, where an edge between two profiles indicates mutual friendship in the social network [20].

We have preprocessed these datasets to keep the largest connected component of the graph and remove nodes that are not labeled red or blue. We report the characteristics of the datasets in Table 1.

6.1.2 Synthetic Datasets. We also consider synthetic datasets constructed from a modified variant of the *Mixed Preferential Attachment* model [3]. The model assumes two groups of nodes: Red (B) and Blue (B). The ratio of the two groups is controlled by the parameter r of the model and homophily of the network is controlled by the parameters π_R and π_B, where π_C is the probability that a node in group C connects to a node in the same group.

Given parameters n, d, r, π_R and π_B and an initial undirected graph G_0 on n_0 nodes, we construct an undirected graph $G = G(n, d, r, \pi_R, \pi_B)$ as follows: Starting from G_0, we iteratively add $n - n_0$ nodes to G one at a time. A new node v is red with probability r and blue with probability $1 - r$. We then add d edges to node v. For each edge, the group of the other endpoint is the same with probability π_C and different with probability $1 - \pi_C$, where C is the group of v. We select a node from this group according to preferential attachment. For our experiments, we generate datasets for G_0 being the complete graph on $n_0 = 11$ nodes and $d = 10$.

[2]https://github.com/stzimas/pagerank-fairness
[3]https://websites.umich.edu/~mejn/netdata/

6.2 Algorithms

In our experiments we consider the three greedy algorithms we defined in Section 5, for both neighborhood local fairness and residual local fairness. For each greedy algorithm Greedy-f we also consider a corresponding efficient heuristic Sorted-f, where we compute the values of the function f only once at the beginning, then we sort the nodes x in decreasing order of the $f(x)$ values, and we select the nodes in that order. We thus have three Sorted algorithms: SortedGain, SortedLoss, and SortedRatio.

We also consider the simple heuristic of selecting at each iteration the node with the highest Pagerank value. That is, we set $f(x) = p(x)$. We will refer to this heuristic as GreedyPR. The motivation for this algorithm comes from Eq. (8), where the Pagerank value of a node x is a multiplicative factor in the fairness gain of making node x ϕ-fair.

Finally, we also consider the simple baseline that selects nodes at random. We will refer to this algorithm as Random.

6.3 Results

We now present our experimental results on real and synthetic datasets. In all of the following experiments, for simplicity, we report experiments with target fairness gain equal to 0.1, that is, for target $\phi = p(R) + 0.1$. In all experiments, we use the group with the lowest original Pagerank value as the red group.

6.3.1 Evaluation on Synthetic Datasets. We first compare our algorithms on synthetic datasets, where we vary the homophily parameters π_R and π_B. Specifically, we generate datasets with $n = 1000$ nodes, balanced group sizes ($r = 0.5$), and homophily parameters $\pi_R = \pi_B \in \{0.1, 0.2, \ldots, 0.9\}$.

The results of our experiments for all the algorithms are summarized in Figure 1, where we report the number of modifications, and the utility loss. The pink and blue panels correspond to the neighborhood locally fair and residual locally fair variants respectively. The height of the panels corresponds to the maximum number of possible modifications (the number of nodes that are candidates to be made ϕ-fair) and the corresponding utility loss. This is the cost of the vanilla Locally Fair Pagerank algorithm. All numbers are averages over 10 synthetic datasets. For the Random algorithm, we report the average results over 10 runs. We note that in all experiments, the original Pagerank is almost equally distributed between the two groups, therefore, we have target $\phi \approx 0.6$.

The first observation is that the GreedyGain is a clear winner in terms of the number of modifications required to achieve fairness. It achieves fairness with less than 65% of the maximum possible modifications, and in some cases with just 5% of the candidate nodes. However, selecting the nodes with the highest gain results in high utility loss: GreedyGain exhibits the highest utility loss among the Greedy algorithms. This is expected, since it selects the nodes with the strongest effect on the Pagerank vector.

The next observation is that, perhaps surprisingly, the GreedyLoss is not the best algorithm in terms of utility loss. GreedyLoss tries to minimize the utility loss of the modifications, and selects nodes that yield small gains in fairness. Therefore, it requires a large number of modifications (the highest among all algorithms, including random), which eventually results in increased utility loss.

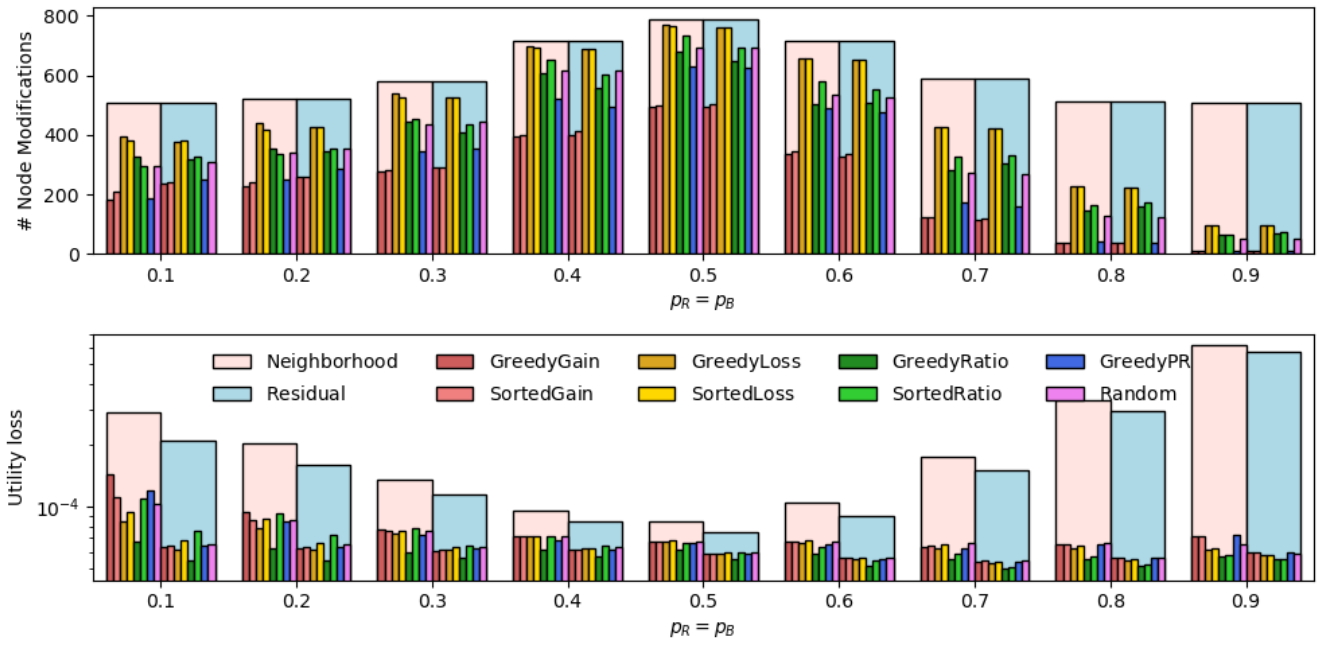

Figure 1: Number of modifications and utility loss for our algorithms for variable data homophily. Utility loss is reported in logarithmic scale. The light pink and light blue panels correspond to neighborhood locally fair and residual locally fair variants respectively. The height of these panels corresponds to the cost of the vanilla Locally Fair Pagerank algorithm.

The GreedyRatio algorithm strikes the best balance between number of modifications and utility loss. It achieves the lowest utility loss, with fewer modifications than the GreedyLoss.

Finally, we observe that the relative performance of the algorithms is consistent between the neighborhood and residual local fairness. However, all algorithms have lower utility loss in the case of residual local fairness. This is due to the fact that the residual Pagerank weight is allocated to a larger number of nodes, resulting in lower utility loss.

6.3.2 The effect of homophily. We use the same dataset to study the effect of homophily on the performance of our algorithms. Recall that the homophily parameters take the values $\pi_R = \pi_B \in \{0.1, 0.2, \ldots, 0.9\}$. The case for $\pi_R = \pi_B = 0.5$ indicates a neutral setting, where nodes have no group preference in their connections. Values greater than 0.5 result in homophilous networks, where nodes tend to connect with nodes in the same group, while values less than 0.5 result in heterophilous networks where nodes tend to connect with nodes from the opposite group. Recall that the target fairness is $\phi \approx 0.6$.

Figure 1 shows the results for the different settings. The first observation is that the neutral case ($\pi_R = \pi_B = 0.5$) is the hardest in terms of the number of modifications required to achieve fairness. In this case, almost all nodes are candidates, but we need to modify a large fraction of them to achieve fairness, even for the GreedyGain algorithm. The reason is that in this case, all nodes have balanced neighborhoods, so there is no transition imbalance we can exploit. Making a node ϕ-fair has a small effect on the Pagerank values, so we need several modifications to achieve fairness. This is also evident in the utility loss: the maximum utility loss is the lowest over all settings and we have low utility loss for all algorithms, despite the large number of modifications.

As we deviate from neutrality, the number of candidates, and the required modifications decrease. We now have nodes with large fairness deficit in their transitions (the blue nodes in the case of homophilous networks, and the red ones in the case of heterophilous networks). Making them fair has a strong effect on Pagerank. However, we have a very different behavior on the opposite ends of the spectrum. The homophilous networks ($\phi > 0.5$) are the easy case for our algorithms: we can achieve fairness with a small number of modifications, resulting also in low utility loss. On the other hand, for the heterophilous networks ($\phi < 0.5$), the number of modifications is considerable, and we have high utility loss.

To explain this, we observe that homophily tends to form a network graph consisting of two monochromatic segregated cliques. The modification of a ϕ-unfair blue node will redirect Pagerank weight from the blue clique to the red clique, increasing the red Pagerank weight significantly. On the other hand, heterophily tends to form a bipartite graph, where red nodes connect with blue nodes, and they are ϕ-unfair to their color. The modification of a ϕ-unfair red node transfers Pagerank from the blue to the red nodes, but this is not propagated downstream to the red group, since the neighbors of the red nodes are primarily blue.

Formally, looking at Eq. (8), we observe that the gain in fairness depends on the personalized red Pagerank value $Q[z, R]$ of the neighbors z of the modified node x. To have high gain, we need $Q[z, R]$ to be high for red nodes and low for blue nodes. This is

exactly the case in a homophilous network. In a heterophilous network though it is the opposite: blue nodes have relatively high personalized red Pagerank value, while red nodes relatively low. As a result, in the latter case, the modifications are less impactful. In the neutral case, the $Q[z, R]$ values are essentially the same for all neighbors, reducing again the impact of the modification.

6.3.3 Comparison with Heuristics on Synthetic Data. Figure 1 also shows the results of the SORTED algorithms, as well as those of GREEDYPR. Each SORTED algorithm is shown with a lighter color of the corresponding GREEDY algorithm.

We observe that the performance of the SORTED algorithms is close to that of the GREEDY. In terms of number of modifications, the differences are negligible, and they appear mostly for heterophilous datasets. In some cases (for SORTEDLOSS) the SORTED version performs better than the GREEDY counterpart. The differences are more noticeable in the case of utility loss, again for heterophilous datasets. In this case, the SORTEDGAIN performs better than the GREEDYGAIN. However, the differences are still small.

The only case where we have some large differences is for the RATIO algorithms. In this case, for heterophilous datasets the SORTEDRATIO algorithm has significantly higher utility loss than GREEDYRATIO, despite the comparable number of modifications. As homophily increases, SORTEDRATIO becomes inferior to GREEDYRATIO in both number of modifications and utility loss. The differences become negligible for homophilous datasets.

Finally, it is worth noting that, as we deviate from neutrality, the simple GREEDYPR heuristic is competitive with SORTEDGAIN and GREEDYGAIN. Given the simplicity of the heuristic, it is impressive that it performs competitively.

6.3.4 Evaluation on Real Datasets. We now present our experiments on the three real datasets. The red group is again the group with the lower Pagerank value $p(R)$, and the target fairness is $\phi = p(R) + 0.1$. The results are summarized in Figure 2. The results of the RANDOM algorithm are averages over 10 runs.

The first observation is that for the Books and Blogs datasets, which are balanced and extremely homophilous, we observe behavior that is compatible with the synthetic datasets for $r = 0.5$ and $\pi_R = \pi_B = 0.9$. We can achieve fairness with just a couple of modifications and low utility loss. The heuristic algorithms are also competitive, and the GREEDYPR essentially replicates the results of the GREEDYGAIN algorithm.

The Facebook dataset is dense and imbalanced, with the red group (women) being approximately 40% of the graph. Furthermore, the users in the blue group (men) are homophilous, while the users in the red group are heterophilous. The GREEDYGAIN algorithm needs to make ϕ-fair a large percentage of the nodes (around 50%) to achieve fairness, with relatively high utility loss. Its performance is well approximated though by the SORTEDGAIN algorithm. The GREEDYRATIO algorithm achieves the best utility loss, lower than any other algorithm, while performing a moderate number of modifications.

The high relative number of required modifications to achieve fairness is due to the network's roughly complementary red group heterophily (≈ 0.4) and blue group homophily (≈ 0.7). This setup implies that the nodes of both groups have roughly the same percentage of red neighbors on average, resulting in roughly the same personalized red Pagerank values $Q[z, R]$ for the nodes of both groups. In Eq. (8), these values roughly cancel each other out, yielding a significantly low fairness gain per node modification. We also observe that in this setup Residual modifications incur greater utility loss. An explanation for this occurrence lies in the capability of existence of blue hubs with a lot of red neighbors. Modifying a node to redistribute weight to all red nodes guaranties that a lot of Pagerank ends up at such a blue hub, but modifying the node to redistribute only to its red neighbors does not necessarily do so in the same magnitude or even at all.

For the Facebook dataset, in Figure 3, we also report fairness gain and utility loss values over the course of the algorithm execution. We observe that using the *gain* function results in a fast increase in fairness gain at the expense of a fast increase in utility loss, whereas using the *loss* function results in a slow increase in utility loss at the expense of a slow increase in fairness gain. This fact is also exemplified by the concavity and convexity of the respective curves. Using the *ratio* function balances between the two extremes, producing a curve between them.

6.3.5 Scalability analysis. To evaluate the scalability of our algorithms, we use synthetic datasets with variable network size. Specifically, we generate datasets with balanced group sizes ($r = 0.5$), homophily parameters $\pi_R = \pi_B = 0.7$, and n nodes where $n \in \{1000, 2000, 4000, 8000\}$. We measure the number of iterations (modifications) and the total execution time. Our experiments were performed on a machine with an *AMD Ryzen 9 5950X 16-Core Processor*. The results are summarized in Figure 4.

We observe that the growth rate of the number of node modifications is roughly the same for all algorithms. The variants using the *gain* selector function are the most efficient, followed by the variants using Pagerank, *ratio*, and finally *loss*.

With respect to running time, we observe that the Residual variants exhibit a substantially greater growth rate than their Neighborhood counterparts. This divergence is attributed to the fact that the Residual modification of a node distributes part the residual Pagerank mass to all red nodes that are not its neighbors, effectively making its respective row of the transition matrix dense, therefore adding to the time complexity of subsequent iterations.

Overall, we observe that there are four groups with respect to time complexity. The fastest group contains all Neighborhood SORTED variants. Such performance is expected, as every iteration of these variants beyond the first one is dominated by the computation of the Pagerank vector via the power method algorithm. The second group contains Neighborhood SORTEDGAIN and GREEDYPR. The third fastest group contains Neighborhood GREEDYGAIN, SORTEDRATIO and SORTEDLOSS. The final group contains Neighborhood GREEDYRATIO and GREEDYLOSS.

6.3.6 Experimental Conclusion. In conclusion, our experiments demonstrate that it is possible to achieve fairness at a reduced cost compared to the vanilla Locally Fair Pagerank algorithm, both in terms of node modifications and utility loss. The cost becomes significantly lower in the case of homophilous datasets, which is often the case in real data. Furthermore, we observe that we can effectively approximate the performance of the GREEDY algorithms with heuristics that are significantly more efficient.

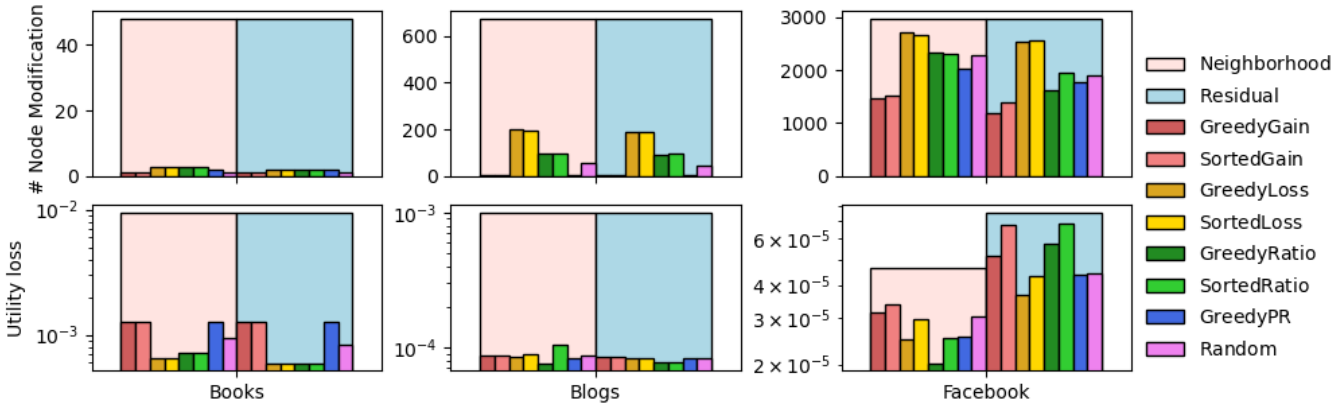

Figure 2: Results of executing our algorithm on the real datasets. For configurations selecting uniformly at random, we consider the average value of 10 executions. Utility loss is reported in logarithmic scale.

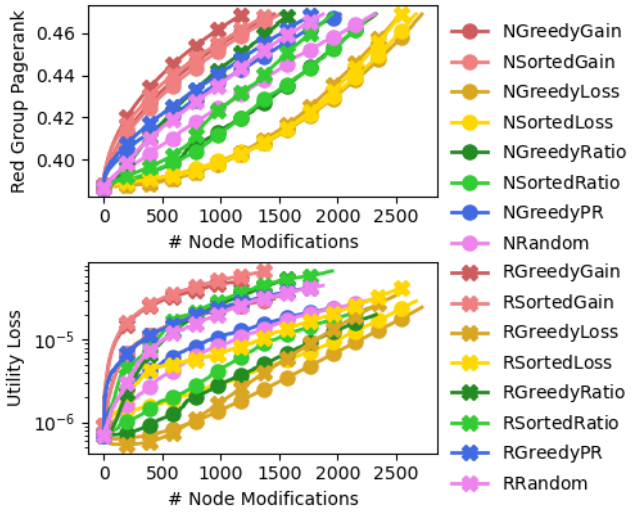

Figure 3: Per iteration performance of our algorithms on the Facebook dataset. Utility loss is reported in logarithmic scale.

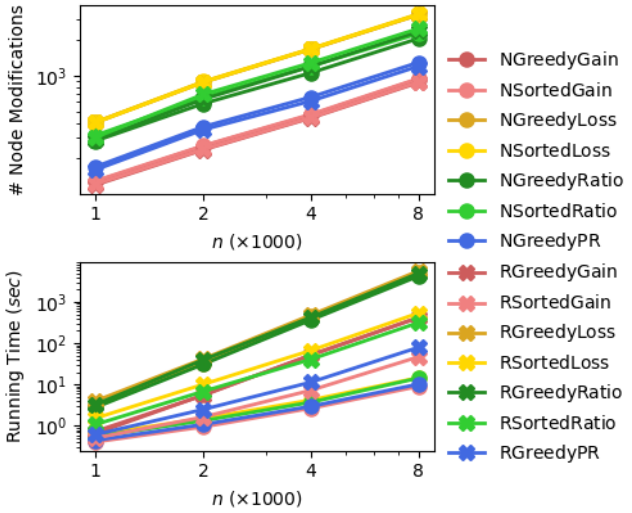

Figure 4: Number of modifications, and running time for our algorithms. All axes are in logarithmic scale.

7 Conclusions

In this paper, we considered the problem of achieving Local Pagerank fairness while minimizing the cost, either in terms of local changes or utility loss. Using analytical expressions for estimating the fairness gain and the utility loss of a local change, we proposed Greedy algorithms and efficient heuristics for selecting the nodes to modify to achieve fairness at a low cost. Our experiments on real and synthetic datasets demonstrate that it is possible to reduce the cost of Pagerank fairness, especially for homophilous networks.

For future work, we intend to further study the tradeoff between gain and loss in the selection of nodes in order to propose more efficient algorithms, and also to investigate the impact of additional network characteristics on our ability to achieve Pagerank fairness at a low cost.

Acknowledgments

This work has been supported by project MIS 5154714, and under the framework of the H.F.R.I. call "Basic Research Financing" (H.F.R.I. Project Number: 016636), both under the National Recovery and Resilience Plan "Greece 2.0" funded by the European Union - NextGenerationEU.

References

[1] L. A. Adamic and N. S. Glance. 2005. The political blogosphere and the 2004 U.S. election: divided they blog. In *LinkKDD*.
[2] Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. 2025. More Asymmetry Yields Faster Matrix Multiplication. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, Yossi Azar and Debmalya Panigrahi (Eds.). SIAM, 2005–2039. https://doi.org/10.1137/1.9781611978322.63
[3] C. Avin, B. Keller, Z. Lotker, C. Mathieu, D. Peleg, and Y. A. Pignolet. 2015. Homophily and the Glass Ceiling Effect in Social Networks. In *ITCS*. 41–50.
[4] Konstantin Avrachenkov and Nelly Litvak. 2006. The effect of new links on Google PageRank. *Stochastic Models* 22, 2 (2006), 319–331.

[5] Solon Barocas, Moritz Hardt, and Arvind Narayanan. 2023. *Fairness and Machine Learning: Limitations and Opportunities.* MIT Press.

[6] 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.

[7] A. J. Biega, K. P. Gummadi, and G. Weikum. 2018. Equity of Attention: Amortizing Individual Fairness in Rankings. In *SIGIR*. 405–414.

[8] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems* 30, 1 (1998), 107–117. https://doi.org/10.1016/S0169-7552(98)00110-X Proceedings of the Seventh International World Wide Web Conference.

[9] Balázs Csanád Csáji, Raphaël M. Jungers, and Vincent D. Blondel. 2014. PageRank optimization by edge selection. *Discret. Appl. Math.* 169 (2014), 73–87.

[10] Cristobald de Kerchove, Laure Ninove, and Paul Van Dooren. 2008. Maximizing PageRank via outlinks. *Linear Algebra Appl* 429, 5-6 (2008), 1274–1256.

[11] Yushun Dong, Jing Ma, Song Wang, Chen Chen, and Jundong Li. 2023. Fairness in Graph Mining: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 35, 10 (2023), 10583–10602.

[12] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS '12)*. 214–226.

[13] Francesco Fabbri, Maria Luisa Croci, Francesco Bonchi, and Carlos Castillo. 2022. Exposure Inequality in People Recommender Systems: The Long-Term Effects. *Proceedings of the International AAAI Conference on Web and Social Media* 16, 1 (2022), 194–204.

[14] G.H. Golub and C.F. Van Loan. 2013. *Matrix Computations.* Johns Hopkins University Press.

[15] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 855–864. https://doi.org/10.1145/2939672.2939754

[16] Jian Kang and Hanghang Tong. 2022. Algorithmic Fairness on Graphs: Methods and Trends. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4798–4799.

[17] Jian Kang, Meijia Wang, Nan Cao, Yinglong Xia, Wei Fan, and Hanghang Tong. 2018. AURORA: Auditing PageRank on Large Graphs. In *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*. 713–722.

[18] F. Karimi, M. Génois, C. Wagner, P. Singer, and M. Strohmaier. 2018. Homophily influences ranking of minorities in social networks. *Nature Scientific Reports* 8 (2018).

[19] Emmanouil Krasanakis, Symeon Papadopoulos, and Ioannis Kompatsiaris. 2021. Applying Fairness Constraints on Graph Node Ranks Under Personalization Bias. In *Complex Networks & Their Applications IX*, Rosa M. Benito, Chantal Cherifi, Hocine Cherifi, Esteban Moro, Luis Mateus Rocha, and Marta Sales-Pardo (Eds.). Springer International Publishing, Cham, 610–622.

[20] Jure Leskovec and Julian Mcauley. 2012. Learning to Discover Social Circles in Ego Networks. In *Advances in Neural Information Processing Systems*, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (Eds.), Vol. 25.

[21] Q. Mei, J. Guo, and D. R. Radev. 2010. DivRank: the interplay of prestige and diversity in information networks. In *KDD*. 1009–1018.

[22] Evaggelia Pitoura, Kostas Stefanidis, and Georgia Koutrika. 2021. Fairness-aware Methods in Rankings and Recommenders. In *2021 22nd IEEE International Conference on Mobile Data Management (MDM)*. 1–4.

[23] Tahleen Rahman, Bartlomiej Surma, Michael Backes, and Yang Zhang. 2019. Fairwalk: Towards Fair Graph Embedding. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. 3289–3295.

[24] A. Singh and T. Joachims. 2018. Fairness of Exposure in Rankings. In *KDD*.

[25] 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.

[26] Ana-Andreea Stoica, Nelly Litvak, and Augustin Chaintreau. 2024. Fairness Rising from the Ranks: HITS and PageRank on Homophilic Networks. In *Proceedings of the ACM Web Conference 2024*. 2594–2602.

[27] Sotiris Tsioutsiouliklis, Evaggelia Pitoura, Konstantinos Semertzidis, and Panayiotis Tsaparas. 2022. Link Recommendations for PageRank Fairness. In *Proceedings of the ACM Web Conference 2022*. 3541–3551.

[28] Sotiris Tsioutsiouliklis, Evaggelia Pitoura, Panayiotis Tsaparas, Ilias Kleftakis, and Nikos Mamoulis. 2021. Fairness-Aware PageRank. In *Proceedings of the Web Conference 2021*. 3815–3826.

[29] K. Yang and J. Stoyanovich. 2017. Measuring Fairness in Ranked Outputs. In *SSDBM*.

[30] 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*.

[31] 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.