Prior Signal Editing for Graph Filter Posterior Fairness Constraints

Emmanouil Krasanakis Symeon Papadopoulos Ioannis Kompatsiaris

MANIOSPAS@ITI.GR PAPADOP@ITI.GR IKOM@ITI.GR

Information Technologies Institute Centre for Research and Technology—Hellas 57001 Thermi, Thessaloniki, Greece

Andreas Symeonidis

ASYMEON@ENG.AUTH.GR

Department of Electrical and Computer Engineering Aristotle University of Thessaloniki 54124, Thessaloniki, Greece

Editor: This is an author preprint

Abstract

Graph filters are an emerging paradigm that systematizes information propagation in graphs as transformation of prior node values, called graph signals, to posterior scores. In this work, we study the problem of mitigating disparate impact, i.e. posterior score differences between a protected set of sensitive nodes and the rest, while minimally editing scores to preserve recommendation quality. To this end, we develop a scheme that respects propagation mechanisms by editing graph signal priors according to their posteriors and node sensitivity, where a small number of editing parameters can be tuned to constrain or eliminate disparate impact. We also theoretically explain that coarse prior editing can locally optimize posteriors objectives thanks to graph filter robustness. We experiment on a diverse collection of 12 graphs with varying number of nodes, where our approach performs equally well or better than previous ones in minimizing disparate impact and preserving posterior AUC under fairness constraints.

Keywords: graph signal processing, node ranking, algorithmic fairness, disparate impact, optimization

1. Introduction

Relational data can be organized into graphs, where entities are represented as nodes and linked through edges corresponding to real-world relations between them. Due to the pervasiveness of complex graphs across disciplines such as social networking, epidemiology, genomic analysis and software engineering, various schemes have been proposed to mine relational information. These range from the unsupervised paradigms of node clustering (Schaeffer, 2007; Kulis et al., 2009) and exposing underlying structures through edge sparsification (Spielman and Teng, 2004) to semi-supervised inference of posterior attribute scores based on known node prior values (Kipf and Welling, 2016; Chen et al., 2020). A mechanism favored by many approaches is propagating latent or predictive attribute information through graphs via recursive aggregation among node neighbors. For example, information

propagation has been used in unsupervised extraction of tightly knit node clusters with few outgoing edges (Andersen et al., 2006; Wu et al., 2012), node ranking algorithms that recommend nodes based on their structurally proximity to a set of query ones (Tong et al., 2006; Kloster and Gleich, 2014) and recent graph neural network advances that decouple latent attribute extraction with their propagation to neighbors (Klicpera et al., 2018; Dong et al., 2020; Huang et al., 2020).

A systematic way to study information propagation is through graph signal processing (Gavili and Zhang, 2017; Ortega et al., 2018; Sandryhaila and Moura, 2013). This domain extends discrete signal processing to higher dimensions, where signals comprise prior values spread not across points in time but across graph nodes. In analogy to time filters, graph filters produce posterior node scores through a weighted aggregation of propagating priors different hops away, where propagation follows either spectral or stochastic rules.

Fairness concerns arise when the outputs of data mining systems, such as graph filters, are correlated to sensitive attributes, such as gender or ethnicity (Chouldechova, 2017; Kleinberg et al., 2018). Previous research has studied bias mitigation in the sense that sensitive and non-sensitive groups of data samples behave similarly under evaluation measures of choice (Chouldechova, 2017; Krasanakis et al., 2018; Zafar et al., 2019; Ntoutsi et al., 2020). In this work, we tackle the fairness objective of achieving (approximate) statistical parity between sensitive and non-sensitive positive predictions—a concept known as disparate impact elimination (Biddle, 2006; Calders and Verwer, 2010; Kamiran and Calders, 2012; Feldman et al., 2015) and often assessed through a measure called *pRule*. In particular, we explore the problem of imposing fairness objectives on graph filter posteriors, such as maximimizing pRule or making it reach a predetermined level, while maintaining the ability to form accurate predictions over different thresholding criteria, as measured by recommender system measures (Shani and Gunawardana, 2011; Wang et al., 2013; Isinkaye et al., 2015), such as AUC. For instance, this can help ensure that a protected set of nodes (the ones considered sensitive) are also frequently but not erroneously recommended.

This work extends our previous paper on making graph filter posteriors fair (Krasanakis et al., 2020a). There, we proposed that satisfying fairness-aware objectives with minimal impact to posterior score quality can be achieved by appropriately editing graph signal priors. This way, new posteriors can be guided to be fairer while respecting information propagation through the graph's structure, which is responsible for predictive quality. In our previous paper, we first tackled this objective by editing priors with schemes of few parameters, which depend on whether nodes are sensitive and the differences between priors and posteriors. Our base assumption was that parameters can be tuned to yield priors proportionate to their estimated contribution to fair yet similar to original posteriors.

In this work we improve various aspects of our previous research. First, we provide a novel mathematical framework to express how tightly prior editing mechanisms should approximate the gradients of posterior objectives and use it to explain why coarse prior editing models can locally optimize fairness-aware objectives. Second, we propose an alternative to our approach that uses an error-based instead of perturbation-based surrogate model, a more robust training objective that is not impacted by high posterior score outliers and an explicit prior retention term. These changes induce higher average AUC when mitigating

^{1.} Discrete signal processing can be modeled by graph signal processing if time is considered a line graph where points in time are nodes and consecutive ones are linked.

disparate impact, though the two alternatives outperform each other on different experiments. Finally, we assess the efficacy of our work by experimenting on a significantly larger corpus of 12 instead of 4 multidisciplinary real-world graphs combined with 8 instead of 2 base graph filters. To facilitate an informed discussion of our results, we also perform a rigorous instead of empirical post-hoc analysis of summary statistics.

2. Background

In this section we provide the theoretical background necessary to understand our work. We start with a common community-based interpretation of node scores and their practical usefulness in Subsection 2.1. Then, in Subsection 2.2, we present graph signal processing concepts used to study a wide range of methods for obtaining node scores given prior information of node attributes. We finally discuss algorithmic fairness under the prism of graph mining and overview the limited research done to merge these disciplines in Subsection 2.3. The operations and symbols used in this work are summarized in Table 2.

Notation	Interpretation
\mathcal{I}	Identity matrix with appropriate dimensions
0	Column vector of appropriate rows and zero elements
1	Column vector of appropriate rows and one elements
r[v]	Element corresponding to node v of graph signal r
$\mathcal{L}(r)$	Loss function for graph filter posteriors r
$\nabla \mathcal{L}(r)$	Gradient vector of loss $\mathcal{L}(r)$ with elements $\nabla \mathcal{L}(r)[v] = \frac{\partial \mathcal{L}(r)}{\partial r[v]}$
x	Absolute value for numbers, number of elements for sets
x	L2 norm of vector x computed as $\sqrt{\sum_{v} x[v]^2}$
$ x _1$	L1 norm of vector x computed as $\sum_{v=1}^{\infty} x[v] $
$ x _{\infty}$	Maximum value of x computed as $\max_{v} x[v]$
λ_1	Smallest eigenvalue of a positive definite matrix
$\lambda_{ m max}$	Largest eigenvalue of a positive definite matrix
H(W)	Graph filter on normalization W of the adjacency matrix
$A \setminus B$	Set difference, that is the elements of A not found in B
a^Tb	Dot product of column vectors a, b as matrix multiplication
$\mathbb{R}^{ \mathcal{V} }$	Space of column vectors comprising all graph nodes
$diag([\lambda_i]_i)$	A diagonal matrix $diag([\lambda_i]_i)[i,j] = {\lambda_i \text{ if } i = j, 0 \text{ otherwise}}$
A[u,v]	Element of matrix A at row u and column v
A^T	Transposition of matrix A for which $A^{T}[u.v] = A[v, u]$
A^{-1}	Inverse of invertible matrix A
$\{x \mid cond(x)\}$	Elements x satisfying a condition $cond$
P(e)	Probability of event e occurring
P(e cond)	Probability of event e given that condition $cond$ is satisfied
${\mathcal S}$	Set of sensitive nodes
\mathcal{S}'	Complement of S , that is the set of non-sensitive nodes

Table 1: Mathematical notation. Graph-related quantities refer to a common studied graph.

2.1 Node Scores and Community Structure

Nodes of real-world graphs can often be organized into communities of either ground truth structural characteristics (Fortunato and Hric, 2016; Leskovec et al., 2010; Xie et al., 2013;

Papadopoulos et al., 2012) or shared node attributes (Hric et al., 2014, 2016; Peel et al., 2017). A common task in graph analysis is to score or rank all nodes based on their relevance to such communities. This is particularly important for large graphs, where community boundaries can be vague (Leskovec et al., 2009; Lancichinetti et al., 2009). Furthermore, node scores can be combined with other characteristics, such as their past values when discovering nodes of emerging importance in time-evolving graphs, in which case they should be of high quality across the whole graph. Many algorithms that discover communities with only a few known members also rely on transforming and thresholding node scores (Andersen et al., 2006; Whang et al., 2016).

A measure frequently used to quantify the quality of attribute recommendations, such as node scores, is AUC (Hanley and McNeil, 1982), which compares operating characteristic trade-offs at different decision thresholds. If we consider ground truth node scores $q_{test}[v] = \{1 \text{ if node } v \text{ is a community member}, 0 \text{ otherwise}\}$ (these form a type of graph signal defined in the next subsection), and posterior scores r[v], the True Positive Rate (TPR) and False Positive Rate (FPR) operating characteristics for decision thresholds θ can be respectively defined as:

$$TPR(\theta) = P(q_{test}[v] = 1 \mid r[v] \ge \theta)$$
$$FPR(\theta) = P(q_{test}[v] = 0 \mid r[v] \ge \theta)$$

where P(a|b) denotes the probability of a conditioned on b. Then, AUC is defined as the cumulative effect induced to the TPR when new decision thresholds are used to change the FPR and quantified per the following formula:

$$AUC = \int_{-\infty}^{\infty} TPR(\theta)FPR'(\theta) d\theta \tag{1}$$

AUC values closer to 100% indicate that community members achieve higher scores compared to non-community members, whereas 50% AUC corresponds to randomly assigned node scores.

2.2 Graph Signal Processing

2.2.1 Graph signals

Graph signal processing (Ortega et al., 2018) is a domain that extends traditional signal processing to graph-structured data. To do this, it starts by defining graph signals $q: \mathcal{V} \to \mathbb{R}$ as maps that assign real values q[v] to graph nodes $v \in \mathcal{V}$.² Graph signals can be represented as column vectors $q' \in \mathbb{R}^{|\mathcal{V}|}$ with elements $q'[i] = q[\mathcal{V}[i]]$, where $|\cdot|$ is the number of set elements and $\mathcal{V}[i]$ is the *i*-th node of the graph after assuming an arbitrary fixed order. For ease of notation, in this work we use graph signals and their vector representations interchangeably by replacing nodes with their ordinality index—in other words, we assume the isomorphism $\mathcal{V}[i] = i$.

Graph signals are often constructed from sets of query nodes that share an attribute of interest, in which case their elements are assigned binary values depending on whether respective nodes are queries $q[v] = \{1 \text{ if } v \text{ has the attribute}, 0 \text{ otherwise}\}$. For example, the

^{2.} Signals with multidimensional node values can be expressed as ordered collections of real-valued signals.

attribute of interest could capture whether nodes belong to the same structural community as query ones. In general, we consider graph signals that are constrained to non-negative values $q \in [0,\infty)^{|\mathcal{V}|}$, where q[v] are proportional to the probabilities that nodes v exhibit the attribute of interest. In this case, graph signal elements can be understood as node rank scores.

2.2.2 Graph Signal Propagation

A pivotal operation in graph signal processing is the one-hop propagation of node values to their graph neighbors, where incoming values are aggregated on each node. Expressing this operation for unweighted graphs with edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ requires the definition of adjacency matrices A, whose elements correspond to the binary existence of respective edges, i.e. $A[i,j] = \{1 \text{ if } (\mathcal{V}[i],\mathcal{V}[j]) \in \mathcal{E}, 0 \text{ otherwise}\}$. A normalization operation is typically employed to transform adjacency matrices into new ones W with the same dimensions, but which model some additional assumptions about the propagation mechanism (see below for details). Then, single-hop propagation of node values stored in graph signals q to neighbors yields new graph signals q_{next} with elements $q_{next}[u] = \sum_{v \in \mathcal{V}} W[u,v]q[v]$. For the sake of brevity, this operation is usually expressed using linear algebra as $q_{next} = Wq$.

Two popular types of adjacency matrix normalization are column-wise and symmetric. The first sets up one-hop propagation as a stochastic process (Tong et al., 2006) that is equivalent to randomly walking the graph and selecting the next node to move to from a uniformly random selection between neighbors. Formally, this is expressed as $W_{col} = AD^{-1}$, where $D = diag(\left[\sum_{v} A[u,v]\right]_{u})$ is the diagonal matrix of node degrees. Columns of the column-normalized adjacency matrix sum to 1. This way, if graph signals priors model probability distributions over nodes, i.e. their values sum to 1 and are non-negative, posteriors also model probability distributions.

On the other hand, symmetric normalization arises from a signal processing perspective, where the eigenvalues of the normalized adjacency matrix are treated as the graph's spectrum (Chung and Graham, 1997; Spielman, 2012). In this case—and if the graph is undirected in that the existence of edges (u,v) also implies the existence of edges (v,u)—a symmetric normalization is needed to guarantee that eigenvalues are real numbers. To achieves this, the normalization $W_{symm} = D^{-1/2}AD^{-1/2}$ is predominantly selected, on merit that it has bounded eigenvalues and $D^{1/2}(I-W_{symm})D^{1/2}$ is a Laplacian operator that implements the equivalent of discrete derivation over graph edges.

Graph signal processing research often targets undirected graphs and symmetric normalization, since these enable computational tractability and closed form convergence bounds of resulting tools.³ In this work, we also favor this practice, because it also allows the graph equivalent of signal filtering we later adopt to maintain spectral characteristics needed by our analysis. The key property we take advantage of is that, as long as the graph is connected, the normalized adjacency matrix W is invertible and has the same number of real-valued eigenvalues as the number of graph nodes $|\mathcal{V}|$ residing in the range [-1,1]. If we annotate these eigenvalues as $\{\lambda_i \in [-1,1] | i=1,\ldots,|\mathcal{V}|\}$, the symmetric normalized

^{3.} Theoretical groundwork has been established to consider spectral equivalent for undirected graphs and, ultimately, asymmetric adjacency matrix normalizations (Chung, 2005; Yoshida, 2019). Unfortunately, these involve the extraction of Peron vectors in polynomial yet non-linear times (Björner and Lovász, 1992) that do not scale well to graphs with hundreds of thousands or millions of nodes and edges.

adjacency matrix's Jordan decomposition takes the form:

$$W = U^{-1} diag([\lambda_i]_i)U$$

where U is an orthonormal matrix with columns the corresponding eigenvectors. Therefore, scalar multiplication, power and addition operations on the adjacency matrices also transform eigenvalues the same way. For example, it holds that $W^n = U^{-1} diag([\lambda_i^n]_i)U$.

2.2.3 Graph filters

The one-hop propagation of graph signals is a type of shift operator in the multidimentional space modeled by the graph, in that it propagates values based on a notion of relational proximity. Based on this observation, graph signal processing uses it analogously to the time shift z^{-1} operator and defines the notion of graph filters as weighted aggregations of multi-hop propagations. In particular, since $W^n q$ expresses the propagation $n = 0, 1, 2, \ldots$ hops away of graph signals q, a weighted aggregation of these hops means that the outcome of graph filters can be expressed as:

$$r = H(W)q$$

$$H(W) = \sum_{n=0}^{\infty} h_n W^n$$
(2)

where H(W) is graph filter characterized by real-valued weights $\{h_n \in \mathbb{R} | n = 0, 1, 2, ...\}$ indicating the importance placed on propagation of graph signals n hops away. In this work, we understand the resulting graph signal r to capture posterior node scores that are the result of passing prior node values of the original signal q through the filter.

In this work, we consider graph filters that are positive definite matrices, that is whose eigenvalues are all positive. For symmetric normalized graph adjacency matrices with eigenvalues $\{\lambda_i \in [-1,1] | i=1,\ldots,|\mathcal{V}|\}$, it is easy to check whether graph filters defined per Equation 2 are positive definite, as they assume eigenvalues $\{H(\lambda_i) = \sum_{n=0}^{\infty} h_n \lambda_i^n \mid i=1,\ldots,|\mathcal{V}|\}$ and hence we can check whether the graph filter's corresponding polynomial assumes only positive values:

$$H(\lambda) > 0 \,\forall \lambda \in [-1, 1] \tag{3}$$

For example, filters arising from decreasing importance of propagating more hops away $h_n > h_{n+1} \,\forall n = 0, 1, \ldots$ are positive definite. Two well-known graph filters that are positive definite for symmetric adjacency matrix normalizations are personalized pagerank (Andersen et al., 2007a; Bahmani et al., 2010) and heat kernels (Kloster and Gleich, 2014). These respectively arise from power degradation of hop weights $h_n = (1-a)a^n$ and the exponential kernel $h_n = e^{-t}t^n/n!$ for empirically chosen parameters $a \in [0,1]$ and $t \in \{1,2,3,\ldots\}$.

2.2.4 Sweep ratio

The sweep procedure (Andersen et al., 2006, 2007b) is a well-known mechanism that takes advantage of graph filters to identify tightly-knit congregations of nodes that are also well-separated from the rest of the graph—a concept known as low subgraph conductance (Chalupa, 2017). When attributes modeled by graph signal priors are closely correlated

to the formation of structural communities, it enhances the recommendation quality of posteriors.

In detail, the sweep procedure assumes that a base personalized graph node scoring algorithm R with strong locality (Wu et al., 2012), such as personalized pagerank and heat kernels, outputs graph signal posteriors R(q) for graph signal priors q that comprise structurally close query nodes. The posteriors are said to be personalized on the query nodes and are compared to their non-personalized counterparts R(1), where 1 is a vector of ones, through the following division:

$$r_{sweep}[v] = \frac{R(q)[v]}{R(\mathbf{1})[v]} \tag{4}$$

In this work, we follow the terminology of our previous research (Krasanakis et al., 2020a) and refer to the post-processing of Equation 4 as the *sweep ratio*. The sweep procedure orders all nodes based on this ratio and splits the order in two partitions so that conductance is minimized for the respective graph cut. This practice statistically yields well-separated partitions for a variety of node ranking algorithms (Andersen et al., 2006, 2007b; Chalupa, 2017). Hence, when nodes are scored based on their relevance to structural communities, the sweep ratio can be deployed to improve their quality.

2.3 Fairness of Graph Filter Posteriors

In this subsection we introduce the well-known concept of disparate impact in the domain of algorithmic fairness, which we aim to either fully or partially mitigate. We also overview previous works that can be used to bring fairness to graph mining and graph filter posteriors.

2.3.1 DISPARATE IMPACT ELIMINATION

Algorithmic fairness is broadly understood as parity between sensitive and non-sensitive samples over a chosen statistical property. Three popular fairness-aware objectives (Choulde-chova, 2017; Krasanakis et al., 2018; Zafar et al., 2019; Ntoutsi et al., 2020) are disparate treatment elimination, disparate impact elimination and disparate mistreatment elimination. These correspond to not using the sensitive attribute in predictions, preserving statistical parity between the fraction of sensitive and non-sensitive positive labels and achieving identical predictive performance on the two groups under a measure of choice.

In this work, we focus on mitigating disparate impact unfairness (Chouldechova, 2017; Biddle, 2006; Calders and Verwer, 2010; Kamiran and Calders, 2012; Feldman et al., 2015). An established measure that quantifies this objective is the pRule (Biddle, 2006); denoting as $R[v] \in \{0,1\}$ the binary outputs of a system R for samples v, S the set of sensitive samples and S' the set of non-sensitive ones, that is the complement of S:

$$pRule = \frac{\min\{q_S, q_{S'}\}}{\max\{q_S, q_{S'}\}} \in [0, 1]$$

$$q_S = P(R[v] = 1 | p \in S)$$

$$q_{S'} = P(R[v] = 1 | p \notin S)$$
(5)

The higher the pRule, the fairer a system is. There is precedence (Biddle, 2006) for considering 80% pRule or higher fair, and we also adopt this constraint in our experiments later.

Calders-Verwer disparity $|p_S - q_{S'}|$ (Calders and Verwer, 2010) is also a well-known disparate impact assessment measure. However, although it is optimized at the same point as the pRule, it biases fairness assessment against high fractions of positive predictions. For example, it considers the fractions of positive labels $(p_S, q_{S'}) = (0.8, 0.6)$ less fair than $(p_S, q_{S'}) = (0.4, 0.3)$. We shy away from this understanding because a stochastic interpretation of posterior nodes scores could be scaled by a constant yet unknown factor. On the other hand, the pRule would quantify both numerical examples as $\frac{0.6}{0.8} = \frac{0.3}{0.4} = 75\%$ fair.

2.3.2 Posterior score fairness

In domains related to the outcome of graph mining algorithms, fairness has been defined for the order of recommended items (Beutel et al., 2019; Biega et al., 2018; Yang and Stoyanovich, 2017; Zehlike et al., 2017) as equity in the ranking positions between sensitive and non-sensitive items. However, these notions of fairness are not applicable to the more granular understanding provided by node scores.

Another definition of graph mining fairness has been introduced for node embeddings (Bose and Hamilton, 2019; Rahman et al., 2019) under the guise of fair random walks—the stochastic process modeled by personalized pagerank when the adjacency matrix is normalized by columns. Yet the fairness of these walks is only implicitly asserted through embedding fairness. Furthermore, they require a certain number of sensitive nodes to make sure that at least one is available to walk to at every step.

Fairness has also been recently explored for the outcome of graph neural networks (Dai and Wang, 2020), which can be trained to produce fair recommendations, even under partial knowledge of sensitive attributes. Still, advances on graph neural network theory (Dong et al., 2020) suggest that they can be perceived as decoupled multiplayer perceptron and graph filter components, in which case the question persists on how to make the outcome of graph filters fair, given potentially unfair priors. For graph neural networks that follow a predict-then-propagate paradigm (Klicpera et al., 2018), this could be achieved by the aforementioned practice of adding fairness objectives to loss functions responsible for the construction of graph signal priors from data. However, the model deriving graph signal priors could be too costly to retrain or not available. In such cases, the problem of inducing fairness reverts to the graph signal processing viewpoint of this work.

A recent work by Tsioutsiouliklis et al. (2020) has initiated a discourse on posterior score fairness. Although focused on algorithms scoring the structural importance of nodes without any kind of personalization pertaining to specific graph signal priors, it first recognized the need for optimizing a trade-off between fairness and preserving posterior score quality. Furthermore, it provided a first definition of node score fairness, called ϕ -fairness. Under a stochastic interpretation of node scores, where they are proportional to the probability of nodes assuming positive labels, ϕ -fairness is equivalent to full disparate impact elimination when $\phi = \frac{|S|}{|S| + |S'|}$.

In our previous work (Krasanakis et al., 2020a), we introduced a generalization of pRule to posterior scores, which we also adopt in this work. In particular, we start from the

above-mentioned stochastic interpretation of posteriors and calculate the expected positive number of sensitive and non-sensitive node labels obtained by sampling mechanism that uniformly assigns positive labels with probabilities proportional to posterior scores. This defines the quantities:

$$p_S = P(R[v] = 1 | p \in S) = \frac{1}{|S|} \sum_{v \in S} \frac{r[v]}{\|r\|_{\infty}}$$
$$q_{S'} = P(R[v] = 1 | p \notin S) = \frac{1}{|S'|} \sum_{v \notin S} \frac{r[v]}{\|r\|_{\infty}}$$

where the maximum value of posteriors $||r||_{\infty} = \max_{u} r[u]$ is used for normalization and R is a stochastic process with probability $P(R[v] = 1) = \frac{r[v]}{||r[u]||_{\infty}}$. Plugging these in the pRule cancels out the normalization of dividing both the nominator and denominator with the same value and yields the following formula for calculating a stochastic interpretation of the pRule given posterior node scores r:

$$pRule(r) = \frac{\min\{|S'|\sum_{v \in S} r[v], |S|\sum_{v \notin S} r[v]\}}{\max\{|S'|\sum_{v \in S} r[v], |S|\sum_{v \notin S} r[v]\}}$$
(6)

By convention, we consider pRule = 0 when all node scores are zero.

3. Optimizing Posterior Scores

Optimizing graph filter posteriors to better satisfy objectives by adjusting their values deteriorates the quality gained by passing priors through graph filters. Ultimately, this leads to losing the robustness of propagating information through node relations by introducing one additional degree of freedom for each node. To prevent loss of quality, we propose that, instead of posteriors, graph signal priors can be edited based on their initial posteriors so as to find new posteriors better satisfying set objectives. This scheme is demonstrated in Figure 1, where initial priors q and their respective posteriors r are used to construct new edited priors q_{est} . The editing process is ideally controlled by few parameters, which can be tuned to let posteriors r_{est} of edited priors optimize the objective.

In Subsection 3.1 we demonstrate fairness-aware objectives that encourage disparate impact mitigation and discourage large posterior changes. Then, in Subsection 3.2 we explore mechanisms of editing priors to be proportional to the probability of respective posterior scores approaching ideal ones optimizing the objective. This probability is estimated with a surrogate model of differences between original priors and posteriors. Given that biases against sensitive nodes could also affect the ability to estimate ideal posteriors, different model parameters are reserved for nodes with sensitive attributes. We also propose an alternative to our previous approach, that uses absolute errors instead of differences between priors and posteriors and introduces an additional parameter to explicitly control original prior retention. The involvement of sensitive attributes in both prior editing and objective calculation is demonstrated in the updated scheme of Figure 2.

Finally, in Subsection 3.3 we present a new theoretical justification of why prior editing approaches are able to (locally) optimize posterior objectives using few parameters. To do this, we take advantage of graph filter robustness against prior perturbations to show in

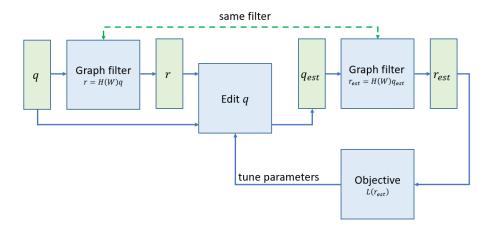


Figure 1: Starting from graph priors q and estimating edited priors q_{est} that help find graph filter posteriors r_{est} locally optimizing an objective $\mathcal{L}(r_{est})$.

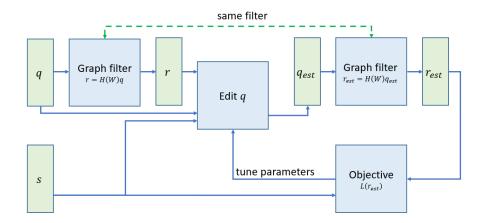


Figure 2: Starting from graph priors q and estimating edited priors q_{est} that help find graph filter posteriors r_{est} locally optimizing an fairness-aware objective $\mathcal{L}(r_{est})$. A graph signal s holding sensitive attribute values is used for prior editing and calculating the objective.

Theorem 2 that non-tight (that is, bound by the eigenvalue ratio of minimum over maximum eigenvalues multiplied by scalar depending on the post-processing) approximations of prior optimization slopes suffice to lead posteriors to local optimality. Under the assumption that prior editing gradients exhibits enough degrees of freedom to express objective gradients with the same loose tightness as before, we then show in Theorem 3 that there exist (not necessarily observed) parameter optimization trajectories that arive at edited priors with locally optimal posteriors.

3.1 Fairness-Aware Objectives

Tuning prior editing mechanisms requires fairness-aware objectives that trade-off disparate impact mitigation and original posterior score preservation. The first of these two components is quantified by the pRule, which can also be constrained so that values over a threshold sup_{prule} do not induce further gains but lower ones are penalized with large weights w_{prule} . On the other hand, retaining original posterior scores can be assessed through a distance measure between the original posteriors r and the estimated ones r_{est} . In our previous work, we penalized the mean absolute difference between the max-normalized version of original and new posteriors:

minimize
$$\mathcal{L}(r_{est})$$

$$\mathcal{L}(r_{est}) = \frac{1}{|V|} \left\| \frac{r_{est}}{\|r_{est}\|_{\infty}} - \frac{r}{\|r\|_{\infty}} \right\|_{1} - w_{prule} \cdot \min\{pRule(r_{est}), sup_{prule}\}$$
(7)

A shortcoming of this objective is that it is heavily influenced by the maximum posterior scores used for normalization. Furthermore, if a few high posteriors were disproportionately large (for example, orders of magnitude larger) compared to the rest, their comparisons would dominate the assessment. As an extreme example, if a clique of nodes with prior values 1 were disconnected from the rest of the graph, personalized pagerank would yield posteriors 1 for these and hence would not normalize other node scores at all. Similar arguments hold if priors favor the discovery of well-separated communities from the rest of the graph, as graph filter algorithms often do. For instance, the sweep ratio explicitly aims to amplify this phenomenon.

Dividing posteriors with summary statistics, such as their L1 norm, can not prevent high posterior scores from assuming disproportionately larger values and dominate comparisons. To address this persisting issue, in this work we move away from the mean absolute difference comparison and instead use the KL-divergence of estimated posteriors distributions given the distribution of original high quality posteriors. A clear advantage of this measure is that it compares distributions as a whole (it has thus been used to estimate differences between different Gaussian distributions for metric learning (Wang et al., 2017)) and is hence less influenced by outliers or few disproportionately large posteriors. We convert posteriors to distributions by normalizing them by division with their L1 norm, in which case the KL divergence measures the entropy lost by moving from the original posterior distribution to the new one:

$$KL(r_{est} \mid r) = -\frac{r_{est}^T}{\|r_{est}\|_1} \ln \frac{r/\|r\|_1}{r_{est}/\|r_{est}\|_1}$$
 (8)

where the vector division is performed element-by-element and by convention $0 \ln x = 0 \forall x \in [0, \infty) \cup \{\infty\}$. Replacing the mean absolute error in the previous objective with KL-divergence, we end up with the following more robust variation of the objective:

minimize
$$\mathcal{L}(r_{est})$$

$$\mathcal{L}(r_{est}) = KL(r_{est} \mid r) - w_{prule} \cdot \min\{pRule(r_{est}), sup_{prule}\}$$
(9)

3.2 Fairness-Aware Prior Editing

We now explore in more detail the advances of our previous work (Krasanakis et al., 2020a), where we ported to graphs a pre-processing scheme that weights training samples to make well-calibrated black box classifiers fair (Krasanakis et al., 2018). This scheme proposes that unfairness is correlated to misclassification error (the difference between binary classification labels and calibration probabilities) and is influenced by whether samples are members of a sensitive group. Since strongly misclassified samples could exhibit different degrees of bias from correctly classified ones, it skews calibrated probabilities to make them fair by a transformation of misclassification error. Different skewing parameters are determined for sensitive and non-sensitive samples.

These considerations introduce a type of balancing between original predictive abilities and sources of unfairness, depending on the parameters of misclassification error transformation and the ones controlling whether calibration probabilities should increase and decrease. Unfortunately, the same principles can not be ported as-are to graph signal processing, because weighting zero node priors through multiplication does not affect posteriors and there is no posterior validation set on which to tune permutation parameters. We address these issues by respectively performing non-linear edits of graph signal priors instead of scaling them and using priors as a rough one-class validation set of known positive examples.

3.2.1 Conditional error estimation

Under the above assumptions, we refer to a stochastic interpretation of graph signal posteriors similar to the one used to set up the stochastic generalization of pRule; posteriors are snapped to 1 with probability proportional to their value and to 0 otherwise. We also consider edited graph signal priors q_{est} that help estimate posteriors $r_{est} = H(W)q_{est}$ of similar fairness to some unobserved ideal ones r_{fair} . Then, given that graph signal priors suggest a ground truth categorization of nodes based on predictive attributes, we condition the probability of fairness-inducing node posteriors achieving their ideal values on whether they match their priors, that is on whether high posterior values correspond to high prior values. This is formally expressed for graph nodes v as:

$$P(r_{fair}[v] = r_{est}[v]) = P(r_{fair}[v] = r_{est}[v] | q[v] = r_{est}[v])P(q[v] = r_{est}[v]) + P(r_{fair}[v] = r_{est}[v] | q[v] \neq r_{est}[v])P(q[v] \neq r_{est}[v])$$

Since graph filters tend to be strongly local in the sense that their posteriors preserve at least some portion of their priors (a fraction of their value equal to or greater than $\frac{h_0}{\sum_{n=0}^{\infty} h_n}$ comes from their priors if non-negative parameters are used to define filters per Equation 2), we further argue that fairness of posteriors pertains to fairness of respective priors. Hence, probabilities of estimated node posteriors approaching fair ones given tight approximations

original priors are correlated with the probabilities of priors being fair $P(q_{fair}[v] = q_{est}[v])$. The same reasoning applies for estimated posteriors not approximating well the original priors.

An additional observation is that, when one of the conditional probabilities of correctly discovering fairness-aware posteriors increases (compared to the rest of nodes), the others should decrease and conversely, as they are conditioned on complementary events. Therefore, they should not simultaneously overestimate or underestimate original ones. Formally, this property can be expressed as:

$$\left(\frac{1}{K_p}P(r_{fair}[v] = r_{est}[v] \mid q[v] = r_{est}[v]) - P(r_{fair}[v] = r_{est}[v])\right)
\cdot \left(\frac{1}{K_p}P(r_{fair}[v] = r_{est}[v] \mid q[v] \neq r_{est}[v]) - P(r_{fair}[v] = r_{est}[v])\right) < 0$$

for some scaling parameters $K_p, K_n > 0$ that let probabilities sum to 1. We now develop a surrogate model satisfying this property. This model depends on differences between posteriors and priors and whether nodes are sensitive:

$$P(r_{fair}[v] = r_{est}|q[v] = r_{est}[v]) \approx K_p P(q_{fair}[v] = q_{est}[v]) e^{-b[v](r[v]/||r||_{\infty} - q[v])} P(r_{fair}[v] = r_{est}|q[v] \neq r_{est}[v]) \approx K_n P(q_{fair}[v] = q_{est}[v]) e^{b[v](r[v]/||r||_{\infty} - q[v])}$$

where b is a vector of real values such that $b[v] = \{b_S \text{ if } v \in S, b_{S'} \text{ otherwise}\}.$

3.2.2 Prior editing mechanism

The selection of sensitive or non-sensitive nodes to contribute to original priors can be viewed as Bernoulli trials with probabilities α_S and $\alpha_{S'}$ (Krasanakis et al., 2018), which may differ depending on bias involved in prior selection. For example, there is often a lower probability of reporting sensitive nodes to construct priors (Cassel and Bindman, 2019). We organize prior selection probabilities into a graph signal α with values $\alpha[v] = P(q[v] = r_{est}[v]) = {\alpha_S \text{ if } v \in S, \alpha_{S'} \text{ otherwise}} \in [0, 1].$

Therefore, given the surrogate model of conditional errors, we make a fair prior estimation q_{est} of ideal priors based on the self-consistency criterion that, when it approaches fairness-inducing personalization, estimated fair ranks should also approach the ideal fair ones:

$$q_{est}[v] = P(r_{fair}[v] = r_{est}[v] \mid q_{fair}[v] = q_{est}[v]) \approx \frac{P(r_{fair}[v] = r_{est}[v])}{P(q_{fair}[v] = q_{est}[v])}$$

$$= \alpha[v] K_p e^{-b[v](r[v]/||r||_{\infty} - p[v])} + (1 - \alpha[v]) K_n e^{b[v](r[v]/||r||_{\infty} - p[v])}$$

$$\propto a[v] e^{-b[v](r/||r||_{\infty} - p[v])} + (1 - a[v]) e^{b[v](r[v]/||r||_{\infty} - p[v])}$$
(10)

for graph signal $a[v] = \{a_S \text{ if } v \in S, a_{S'} \text{ otherwise}\} \in [0, 1]$ that relates to prior selection probabilities and permutation scaling factors per $a[v] \propto \frac{\alpha[v]K_p}{\alpha[v]K_p+(1-\alpha[v])K_n}$. Determining a requires only two independent parameters $a_S, a_{S'}$.

3.2.3 Error-based prior editing

In the above process we started with a surrogate model of conditional fair posterior probability estimation that uses the differences between original priors and their normalized

posteriors as inputs. Although this approach was met with success in our previous work, we also point out that it implicitly involves information on whether nodes assume positive values in the original binary priors instead of only the deviation between posteriors and priors. In detail, the differences $r[v]/||r||_{\infty} - p[v]$ are positive for zero priors p[v] = 0 but negative for positive priors p[v] = 1. For example, when b[v] > 0 high posteriors are penalized for the conditional probability of priors approaching estimated posteriors when in reality high posteriors need to be encouraged for positive priors. Similarly, when b[v] > 0 it encourages small posteriors for the conditional probability of priors not approaching estimated posteriors when in reality small posteriors need to be encouraged for zero priors.

To introduce an explainable interpretation of parameters b[v], we thus propose the alternative of adopting the absolute value of differences by using errors $|r[v]/||r||_{\infty} - p[v]|$ as inputs to the surrogate model. Following the same analytical process as above, this leads us to the following error-based prior editing mechanism as a potential competitor:

$$q_{est}[v] = a[v]e^{-b[v]\left|r/\|r\|_{\infty} - p[v]\right|} + (1 - a[v])e^{b[v]\left|r[v]/\|r\|_{\infty} - p[v]\right|}$$

The ability of error-based perturbations to mitigate disparate impact while enjoying high predictive quality has also been justified and experimentally corroborated in previous research (Krasanakis et al., 2018). As a final step, and to provide more granular control of the retention of original posteriors required by objective functions, we introduce an explicit trade-off between original and edited fairness-aware priors by adding a term of the latter weighted by a new parameter $a_0 \in [0, 1]$:

$$q_{est}[v] = a_0 q[v] + a[v]e^{-b[v]|r/||r||_{\infty} - p[v]|} + (1 - a[v])e^{b[v]|r|||r||_{\infty} - p[v]|}$$
(11)

3.3 Local Optimality of Prior Editing

In this subsection we investigate theoretical properties of positive definite graph filters that allow coarse approximations of optimal prior editing schemes to also reach local optimality with regards to posterior objectives.

3.3.1 Graph filters with post-processing

Real-world graph filters often end up with a post-processed version of posteriors. For example, when personalized pagerank is implemented as an iterative application of the power method scheme r = aWr + (1 - a)q, potential feedback loops that continuously increase posteriors for some asymmetric adjacency matrix normalizations are often avoided by performing L1 normalization, which divides node posteriors with their sum. This ends up not affecting the ratio of importance scores placed on propagating prior graph signals different number of hops away, but produces a scaled version of the filter for which node score posteriors sum to 1.

More complex post-processing mechanisms, such as the sweep ratio, may induce different transformations per node that can not be modeled by graph filters. Taking these concerns into account, we hereby introduce a notation with which to formalize post-processing; we consider a $post-processing\ vector$ with which posteriors are multiplied element-by-element. Formally, this lets us write post-processed posteriors r of passing graph signal priors q

through a graph filter H(W) as:

$$r = diag(p)H(W)q \tag{12}$$

We stress that the exact post-processing transformation could change depending on both the graph filter and graph signal priors. However, we can decouple this dependency by thinking of the finally selected post-processing as one particular selection out of many possible ones. In this work we consider two types of postprocessing: a) L1 output normalization and b) the sweep ratio. These are respectively modeled as multiplication with the inverse of the original posterior's L1 norm $q_{L1}[v] = \frac{1}{\|H(W)q\|_1}$ and the inverse of non-personalized posteriors $q_{sweep}[v] = \frac{1}{(H(W)\mathbf{1})[v]}$.

3.3.2 Coarse approximation of gradients

To formalize the concept of approximately tracking the optimization slope of posterior objectives with small enough error, in Definition 1 we introduce λ -optimizers of objectives as multivariate multivalue functions that approximate the (negative) gradients of objectives with relative error bound λ . Smaller values of this strictness parameter indicate tighter approximation of optimization slopes, whereas smaller values indicate looser tracking. To disambiguate the possible directions of slopes, our analysis considers loss functions of nonnegative values to be minimized.

Definition 1 A continuous function $f: \mathcal{R} \to \mathbb{R}^{|\mathcal{V}|}$ will be called a λ -optimizer of a loss function $\mathcal{L}(r)$ over graph signal domain $\mathcal{R} \subseteq \mathbb{R}^{|\mathcal{V}|}$ only if:

$$||f(r) + \nabla \mathcal{L}(r)|| < \lambda ||\nabla \mathcal{L}(r)||$$
 for all $\nabla \mathcal{L}(r) \neq \mathbf{0}$

.

We now analyse the robustness of positive definite graph filters with post-processing in terms of how tight optimizers of posterior losses should be for the graph filter's propagation mechanism to "absorb" the error. To this end, in Theorem 2 we find a maximum tightness parameter sufficient to lead to local optimality of posteriors with respect to the loss. The required tightness depends on the graph filter's maximum and minimum eigenvalues and the post-processing vector's maximum and minimum values and requires the graph filter to be symmetric positive definite.

Theorem 2 Let H(W) be a positive definite graph filter and p a postprocessing vector. If f(r) is a $\frac{\lambda_1 \min_v p[v]}{\lambda_{\max} \max_v p[v]}$ -optimizer of a differentiable loss $\mathcal{L}(r)$ over graph signal domain $\mathcal{R} \subseteq \mathbb{R}^{|\mathcal{V}|}$, where $\lambda_1, \lambda_{\max} > 0$ are the smallest positive and largest eigenvalues of H(W) respectively, updating graph signals per the rule:

$$\begin{aligned} \frac{\partial q}{\partial t} &= f(r) \\ r &= diag(p)H(W)q \end{aligned} \tag{13}$$

asymptotically leads to the loss to local optimality if posterior updates are closed in the domain, i.e. $r \in \mathcal{R} \Rightarrow diag(p)H(W) \int_0^t f(r)dt \in \mathcal{R}$.

Proof For non-negative posteriors r = diag(p)H(W)q, and non-zero loss gradients, the Cauchy-Shwartz inequality in the bilinear space $\langle x,y\rangle = x^T H(W)y$ determined by the positive definite graph filter H(W) yields:

$$\begin{split} &\frac{d\mathcal{L}(r)}{dt} \\ &= (\nabla \mathcal{L}(r))^T diag(p)H(W) \frac{dq}{dt} \\ &= (\nabla \mathcal{L}(r))^T diag(p)H(W)f(r) \\ &= (\nabla \mathcal{L}(r))^T diag(p)H(W) \big(- \nabla \mathcal{L}(r) + (f(r) + \nabla \mathcal{L}(r)) \big) \\ &\leq -\min_v p[v] (\nabla \mathcal{L}(r))^T H(W) \nabla \mathcal{L}(r) + (\nabla \mathcal{L}(r))^T diag(p)H(W)(f(r) + \nabla \mathcal{L}(r)) \\ &\leq -\lambda_1 \min_v p[v] \|\nabla \mathcal{L}(r)\|^2 + \lambda_{\max} \|f(r) + \nabla \mathcal{L}(r)\| \|diag(p) \nabla \mathcal{L}(r)\| \\ &< -\min_v p[v] \lambda_1 \|\nabla \mathcal{L}(r)\|^2 + \lambda_{\max} \frac{\lambda_1 \min_v p[v]}{\lambda_{\max} \max_v p[v]} \max_v p[v] \|\nabla \mathcal{L}(r)\| \|\nabla \mathcal{L}(r)\| \\ &= 0 \end{split}$$

Therefore, the loss asymptotically converges to a locally optimal point.

3.3.3 Known robustness limits

Following a similar approach as to check whether graph filters are positive definite in Equation 3, we can also bound the eigenvalue ratio by knowing only the type of filter but not the graph or priors. In particular, for graph filters H(W) where W are symmetric adjacency matrices of undirected graphs:

$$\frac{\lambda_1}{\lambda_{\max}} \ge \frac{\min_{\lambda \in [-1,1]} H(\lambda)}{\max_{\lambda \in [-1,1]} H(\lambda)} \tag{14}$$

Since personalized pagerank can be expressed in closed form as the filter $(1-a)(I-aW)^{-1}$ and heat kernels as $e^{-t(I-W)}$, where a and t are their parameters, their respective eigenvalue ratios for $\lambda \in [-1,1]$ are at most $\frac{1-a}{1+a}$ and e^{-2t} . For larger values of a and t, which penalize less the spread of graph signal priors farther away, stricter optimizers are required to keep track of the gradient's negative slope. When normalization is the only post-processing employed, all elements of the personalization vector are the same and bounds for sufficient optimizer strictness coincide with the aforementioned eigenvalue ratios. In practice, these bounds are often lax compared to the small average node score posteriors $\frac{1}{|\mathcal{V}|}$ arising from L1 normalization in graphs with many (such as thousands or millions of) nodes. For example, for personalized pagerank with a=0.85 it suffices to select 0.081-optimizers to edit priors. Even for wider prior diffusion with a=0.99 it suffices to select 0.005-optimizers.

3.3.4 Prior edits as optimizers

Based on the above analysis, we finally explain why, if the surrogate model of bias estimation matches real-world behavior, the proposed personalization error-based editing mechanism

of Equation 11 can tweak posteriors to locally optimize the set fairness-aware objectives. To do this, in Theorem 3 we translate the required tightness of optimizers to a corresponding tightness of projecting loss gradients to vector spaces of prior editing model parameter gradients. When this requirement is met, there exist prior editing model parameters that lead posteriors to locally optimize the objective. Since the same quantity as in Theorem 2 is used to decide adequate tightness, the analysis of the previous subsection indicates that even coarse surrogate models can be met with success.

Intuitively, in Theorem 3 the matrix E computes the difference between the unit matrix and multiplying the matrix of parameter gradients $D_F(\theta)$ with its right pseudo-inverse; these differences are further constrained to matter only for nodes with high gradient values. This means that, as long as the objective's gradient has a clear direction to move towards to and this can be captured by the surrogate prior editing model $F(\theta)$, a parameter trajectory path exists to arrive at locally optimal prior edits. For example, if prior editing had the same number of parameters as the number of nodes and its parameter gradients were linearly independent, $D_F(\theta)$ would be a square invertible matrix, yielding $D_F(\theta)(D_F^T(\theta)D_F(\theta))^{-1}D_F^T(\theta) = \mathcal{I} \Leftrightarrow E = \mathbf{0}$ and the theorem's precondition inequality would always hold. Whereas as the number of parameters decreases, it becomes more important for $F(\theta)$ to be able to induce degrees of freedom in the same directions as its induced loss's gradients, which our theoretical analysis aims to approximate with the self-consistency criterion that prior edits should induce fairness-aware posteriors.

At this point we stress that this section's theoretical analysis indicates that prior editing models with few parameters *could* yield locally optimal priors, since it suffices to form only loose approximations of desired properties. On the other hand, due to the mathematical intractability of plugging in true prior editing gradients in the computation of the pseudo-inverse, the real-world efficacy of proposed prior editing schemes needs to be experimentally corroborated, as we do in the next section. As a final remark, we stress that being able to explicitly retain priors for some parameters, as the new proposed model of Equation 11 (but not the previous model of Equation 10) does, is a necessary condition for our analysis to hold true.

Theorem 3 Let us consider graph signal priors q_0 , positive definite graph filter H(W) with largest and smallest eigenvalues λ_{\max} , λ_1 , post-processing vector p, differentiable loss function $\mathcal{L}(r)$ in domain \mathcal{R} and a differentiable graph signal editing function $F(\theta)$ with parameters $\theta \in \mathbb{R}^K$ for which there exist parameters θ_0 satisfying $F(\theta_0) = q_0$ and $pH(W)F(\theta) \in \mathcal{R}$. Let us consider the $|\mathcal{V}| \times K$ table function $D_F(\theta)$ with rows:

$$D_F[v] = \nabla_{\theta}(F(\theta)[u])$$

where ∇_{θ} indicates parameter gradient vectors. If for any parameters θ it holds that:

$$||E\nabla \mathcal{L}(r)|| < \frac{\lambda_1 \min_v p[v]}{\lambda_{\max} \max_v p[v]} ||\nabla \mathcal{L}(r)|| \quad \text{for } \nabla \mathcal{L}(r) \neq \mathbf{0}$$

$$E = \mathcal{I} - D_F(\theta) (D_F^T(\theta) D_F(\theta))^{-1} D_F^T(\theta)$$

$$r = diag(p) H(W) F(\theta)$$

then there exist parameters θ_{∞} that make respective posteriors $r_{\infty} = diag(p)H(W)F(\theta_{\infty})$ be locally optimal.

Proof Let us consider a differentiable trajectory for parameters $\theta(t)$ for times $t \in [0, \infty)$ that starts from $\theta(0) = \theta_0$ and (asymptotically) arrives at $\theta_{\infty} = \lim_{t \to \infty} \theta(t)$. For this trajectory, it holds that $\frac{dF(\theta(t))}{dt} = D_F(\theta) \frac{d\theta}{dt}$. Then, let us consider the posteriors $r(t) = diag(p)H(W)F(\theta(t))$ arising from the graph signal function $F(\theta(t))$ at times t, as well the least square problem of minimizing the projection of the loss's gradient to the row space of $D_F(\theta(t))$:

minimize
$$\|\nabla \mathcal{L}(r(t)) - D_F(\theta(t))x(t)\|$$

The closed form solution to this problem can be found by:

$$x(t) = (D_F^T(\theta)D_F(\theta))^{-1}D_F^T(\theta)\nabla\mathcal{L}(r(t))$$

Thus, as long as $q(t) = F(\theta(t))$ (Proposition I), the theorem's precondition for posteriors r(t) = diag(p)H(W)q(t) and $\nabla \mathcal{L}(r(r)) \neq \mathbf{0}$ can be written as:

$$\|\nabla \mathcal{L}(r(t)) - D_F(\theta(t))x(t)\| < \frac{\lambda_1 \min_v p[v]}{\lambda_{\max} \max_v p[v]} \|\nabla \mathcal{L}(r(t))\|$$

Hence, if we set x(t) as parameter derivatives:

$$\frac{d\theta(t)}{dt} = x(t) \Rightarrow \|\nabla \mathcal{L}(r(t)) - \frac{dF(\theta(t))}{dt}\| < \frac{\lambda_1 \min_v p[v]}{\lambda_{\max} \max_v p[v]} \|\nabla \mathcal{L}(r(t))\|$$

which means that, on the selected parameter trajectory, $\frac{F(\theta(t))}{dt}$ is a $\frac{\lambda_{\max} \max_v p[v]}{\lambda_1 \min_v p[v]}$ -optimizer of the loss function. As, such, from Theorem 2 it discovers locally optimal posteriors.

As a final step, we now investigate the priors signal editing converges at. To do this, we can see that the update rule leads to the selection of priors q(t) at times t for which:

$$\frac{dq(t)}{dt} = \frac{dF(\theta(t))}{dt}$$

$$\Rightarrow \lim_{t \to \infty} q(t) = q(0) + \int_{t=0}^{\infty} \frac{F(\theta(t))}{dt} dt = F(\theta(t_{\infty})) - F(\theta(0)) = F(\theta(t_{\infty}))$$

We can similarly show that (Proposition I) holds true. Hence, there exists an optimization path of prior editing parameters (not necessarily the same as the one followed by the optimization algorithm used in practice) that arrives at the edited priors $F(\theta(t_{\infty}))$ for some parameters $\theta(t_{\infty})$ that let posteriors exhibit local optimality.

4. Experiment Setup

In this section we describe the experiment settings (graphs and base graph filters), competing approaches and evaluation methodology used to assess whether prior editing can make graph filter posteriors fair.

4.1 Graphs

Our experiments are conducted on 12 real-world graphs from the domains of social networking, scientific collaboration and software engineering. The graphs are retrieved from publicly available sources, namely the SNAP repository of large networks (Leskovec and Krevl, 2014), the network repository (Rossi and Ahmed, 2015), the LINQS repository (lin), the AMiner repository of citation data sets (Tang et al., 2008) and the software dependency graph data set (Musco, 2016). They are selected on merit of seeing widespread use in their respective domain research and comprising multiple node attributes, which we use as predictive and sensitive information.

In all graphs, nodes are organized into potentially overlapping communities based on their attributes. Motivated by the frequent use of graph filters for recommendation, we consider membership to each community as a different binary attribute. When not stated otherwise, and to facilitate experiments with a small portion of seed nodes, we select the first community with more than 100 nodes to experiment on. If there are no explicit sensitive node attributes, we designate members of the second community found during the previous search as sensitive.

The sensitive attributes of social networking graphs could give rise to discriminative behavior against persons, such as recommending other attributes with disproportionately lower scores. Disparate impact mitigation safeguards against this kind of disparity. This is also useful in other types of graphs, where biases are less impactful from a humanitarian perspective, but can still impact the inclusiveness of recommendation to end-users. For example, in scientific collaboration graphs, it could be important for older publishing venues to compete fairly with newer ones by not biasing recommendation scores against them, for example due to fewer handled publications. Or it could be important to make searches of scientific publications respect a lesser-known area of research, such as less known types of diabetes in the PubMed graph below. Finally, in software engineering graphs it is often useful to discover entities (such as libraries, source code artifacts) related to query ones while protecting those of a subsystem from being avoided, for example because it plays a pivotal role in a software project's architecture and potential risks of impacting it should be well-understood.

Below we detail the type of data captured by each graph. Quantifiable characteristics, such as the number of graph nodes, edges, positive labels, sensitive labels and pRule of positive labels given sensitive ones are summarized in Table 2. For most graphs, the pRule of positive labels 0, which indicates that either all positive labels are sensitive or all of them are not sensitive (flipping which nodes are considered sensitive retains data set pRule and experiment results).

ACM (Tang et al., 2008) A co-authorship graph whose nodes are authors forming edges based on whether they have co-authored a publication. We consider communities corresponding to different publication venues (such as journals or conferences) the authors have published in. To construct this data set, we processed the authorship and publication data of the 2017 version of the ACM citation data set extracted by AMiner.⁴

Amazon (Leskovec et al., 2007) A graph comprising frequent Amazon product copurchases, as well as their category. The graph was parsed from frequent co-purchase metadata hosted in the SNAP repository.⁵

^{4.} ACM-Citation-network V9 from https://aminer.org/citation

^{5.} https://snap.stanford.edu/data/amazon-meta.html

Ant (Musco, 2016) A method call graph for the Apache Ant dependency builder tool, where nodes are source code methods and edges indicate that one of the methods called the other. We retrieve this graph from the feature file corresponding to the project's release 1.9.2 in the software dependency graph data set and remove dangling nodes with only one edge. We consider methods of the same class, as identified through their signatures, to belong to the same community. Hence, predicting communities corresponds to predicting the organization of methods into classes.

Citeseer (Getoor, 2005) A citation graph where scientific publications are nodes and edges correspond to citations between them. Nodes are categorized into communities based. We use the version of the data set provided by the LINQS repository. Since our aim is to experiment on attribute propagation mechanisms, we do not experiment with available publication text tokens that could be used by more complex schemes (such as predict-then-propagate graph neural networks (Klicpera et al., 2018) that estimate graph signals through multilayer perceptrons before propagating them).

DBLP (Tang et al., 2008) A co-authorship graph whose nodes are authors forming edges based on whether they have co-authored a publication. We consider communities corresponding to different publication venues the authors have published in. To construct this data set, we processed the authorship and publication data of the 2011 version of the DBLP citation data set extracted by AMiner⁶.

Facebook0 (Leskovec and Mcauley, 2012) A Facebook graph of user friendships starting from given user and record social relations between them and their friends, including relations between friends. Ten such graphs are available in the source material, out of which we experiment on the first one. We use the version of the data set hosted by SNAP and select the anonymized binary 'gender' attribute as sensitive and the first anonymized binary 'education' attribute as the prediction label.

Facebook686 (Leskovec and Mcauley, 2012) A graph obtained through the same process as Facebook0 and choosing a different graph out of those available in the source material (the ego network of user with identifier 686).

Log4J (Musco, 2016) A method call graph for the Java logging project Log4J, where nodes are source code methods and edges indicate that one of the methods called the other. We retrieve this graph from the feature file corresponding to the project's release 2.0b9 in the software dependency graph data set and apply the same preprocessing and community extraction steps as we did for Ant.

Maven (Benelallam et al., 2019) A dependency graph of Java libraries hosted in Maven central.⁷ Nodes correspond to libraries and edges to dependencies between them, i.e. that one of the edge's nodes depends on the other. We organize libraries into communities based on their parent projects (for example, the libraries org.seleniumhq.selenium:selenium-selenium-selenium-support:3.0.1 are both consider part of the org.seleniumhq.selenium project) and remove dangling nodes with only one edge.

^{6.} DBLP-Citation-network V4 from https://aminer.org/citation

^{7.} https://zenodo.org/record/1489120#.YCnNumgzaUk

data set	Nodes	\mathbf{Edges}	\mathbf{pRule}	Positive	Sensitive
ACM	505,016	1,137,011	51%	52,222	1,098
Amazon	554,789	$1,\!545,\!228$	0	$280,\!507$	64,915
Ant	4,920	9,393	0	783	336
Citeseer	3,327	4,676	0	596	668
DBLP	$978,\!488$	3,491,030	10%	1,039	193
Facebook0	333	2519	91%	225	120
Facebook686	170	3,312	91%	94	78
Log4J	2,121	3,600	0	115	272
Maven	76,137	425,339	0	185	488
Pubmed	19,717	44,327	0	4,103	7,739
Squirrel	6,791	10,880	0	232	372
Twitter	18,470	48,365	0	11,355	7,115

Table 2: Characteristics of data sets involved in experiments.

Pubmed (Namata et al., 2012) A citation data set of PubMed publications pertaining to diabetes research, where nodes correspond to papers and edges to citations between them. Nodes form communities based on the type of diabetes they research. We use the version of the data set provided by the LINQS repository.

Squirrel (Musco, 2016) A method call graph for the Squirrel email server written in Java, where nodes are source code methods and edges indicate that one of the methods called the other. We retrieve this graph from the feature file corresponding to the project's release 0.34 in the software dependency graph data set and apply the same preprocessing and community extraction steps as we did for Ant and Log4J.

Twitter (Rossi and Ahmed, 2015) A Twitter graph of political retweets, where nodes are social media users and edges indicate which users have retweeted posts of others. This data set comprises only one anonymized attribute of binary political opinions (left or right). We consider this attribute as sensitive and its complement as prediction labels.

4.2 Base Graph Filters

For our experiments we consider the two popular types of graph filters we outline in Subsection 2.2: personalized pagerank and heat kernels. Commonly used parameters for those filters that encourage few-hop propagation are a=0.85 and t=3. However, previous findings suggest that propagating graph signal priors more hops away leads to higher posterior AUC for communities with many nodes (Krasanakis et al., 2020b). We thus experiment with propagation parameters a=0.99 and t=5 that induce this behavior too. Since the sweep procedure is also a well-known method in improving posterior score quality (for example, it often increases AUC) we also experiment both with and without it. In total, depending on the choice of filter type, propagation parameter and post-processing, we experiment across $2 \cdot 2 \cdot 2 = 8$ base graph filters, which we outline in Table 3.

In all cases, we treat graphs as undirected ones and employ symmetric normalization. This way, graph filters become positive definite and hence support our theoretical results. Posterior scores are computed to numerical precision of 10^{-9} using the implementations of chosen graph filters provided by the $pyqrank^8$ graph ranking library.

^{8.} https://pypi.org/project/pygrank/

Annotation	Type of Filter	Parameter	Sweep Ratio Post-processing
PPR.85	personalized pagerank	a = 0.85	
PPR.99	personalized pagerank	a = 0.99	
HK3	heat kernels	k = 3	
HK7	heat kernels	k = 7	
PPR.85S	personalized pagerank	a = 0.85	\checkmark
PPR.99S	personalized pagerank	a = 0.99	\checkmark
HK3S	heat kernels	k = 3	\checkmark
HK7S	heat kernels	k = 7	\checkmark

Table 3: Base graph filters used in experiments.

4.3 Compared Approaches

In this subsection we outline promising existing and new approaches that improve the fairness of base graph filters. These span multiple fairness-aware objectives, such as maximizing fairness, meeting fairness constraints and trying to preserve the posterior quality measured by AUC, which we detail in Table 4. Their implementation has been integrated in the *pygrank* library.

None The base graph filter.

Mult A simple post-processing baseline that multiplies posterior scores across the sensitive and non-sensitive groups with a different constant each, so that disparate impact is fully mitigated. If r are the base graph filter's posteriors and s a graph signal holding the sensitive attribute, this method post-processes posteriors per the rules:

$$r_{Mult}[v] = \left(\frac{\phi s[v]}{\sum_{u \in S} s[u]r[u]} + \frac{(1-\phi)(1-s[v])}{\sum_{u \notin S} s[u]r[u]}\right)r[v]$$

where $\phi = \frac{|S|}{|S| + |S'|}$ is the fraction of graph nodes that are sensitive and $s[u] = \{1 \text{ if } u \in S, 0 \text{ otherwise} \}$. It holds that $\sum_{v \in S} r_{Mult}[v] = \sum_{v \notin S} r_{Mult}[v]$.

FairWalk A random walk strategy previously used for fair node embeddings. We implement this as an asymmetric preprocessing applied to the graph's adjacency matrix that retains nodes degree but makes signals spread equally between sensitive and non-sensitive neighbors. This approach aims to predominantly maintain the propagation mechanism and attempts to make posteriors fair only if this is convenient by encountering nodes with both sensitive and non-sensitive neighbors.

LFPRO Near-optimal redistribution of ranks causing disparate impact. Contrary to our assumption that aims to influence posteriors by editing priors, this approach directly offsets the disparate impact of posteriors by moving excess node scores between the sensitive and the non-sensitive nodes to improve the pRule as much as possible while maintaining nonnegative scores. To avoid numerical underflows that erroneously prevent this approach from exact convergence in the graphs with many nodes, we repeat the gradual movement of scores up to numerical tolerance 10^{-12} .

FairPers Our previously proposed fair personalization model described by Equation 10. Its tradeoff parameters assume values $a_S, a_{S'} \in [0, 1]$, whereas we limit exponentials to

		Preserve	Maximimize	Fairness
Method		Posteriors	Fairness	Constraints
None				
Mult	(baseline)		\checkmark	
Fairwalk	(Rahman et al., 2019)	✓	Partially	
LFPRO	(Tsioutsiouliklis et al., 2020)	Partially	\checkmark	
FairPers	(Krasanakis et al., 2020a)	Partially	\checkmark	✓
FairPers-C	(Krasanakis et al., 2020a)	✓		✓
FairEdit	(this work)	Partially	\checkmark	✓
FairEdit-C	(this work)	✓		✓

Table 4: Explicit objectives of fairness-aware methods.

large enough range $b_S, b_{S'} \in [-10, 10]$ that, if needed, lets some posteriors dominate the editing mechanism for large exponents of vanish for small exponents. This variation aims to maximize fairness while partially retaining posterior quality and is hence trained towards optimizing Equation 7 for $w_{prule} = 1$ and $sup_{prule} = 1$. Parameters are tuned with an algorithm we developed as part of the pygrank library for non-derivative optimization, which is detailed in Appendix A.

FairPers-C A variation of FairPers that aims to impose strong fairness constraints with $w_{prule} = 10$ of optimizing the pRule up to value $sup_{prule} = 80\%$.

FairEdit The personalization editing mechanism of Equation 11 proposed in this work. Its parameters assume values $\{\theta_K \in [-1,1] | K=0,1\}$ and $a_{pers} \in [0,1]$ and this variation trains them to maximize fairness while partially retaining posterior quality by optimizing Equation 9 for $w_{pRule} = 1$ and $sup_{pRule} = 1$. Parameters are tuned with the same optimization algorithm as FairPers.

FairEdit-C A variation of FairEdit that aims to impose strong fairness constraints with $w_{prule} = 10$ of optimizing the pRule up to value $sup_{prule} = 80\%$.

4.4 Evaluation Methodology

This subsection details the methodology we follow to assess fairness-aware approaches on combinations of graphs and base graph filters, as well as the summary statistics used to perform a high level comparison between approaches.

4.4.1 Train-test splits

To rigorously evaluate fairness-aware approaches, we separate graph nodes into training and test sets, by uniformly sampling the former without repetition to comprise one of the fractions $\{10\%, 20\%, 30\%\}$ of all graph nodes. Positive labels are on average also split alongside the same fraction to construct graph signal priors (i.e. by assigning 1 to their respective elements). For example, for the Maven data set a 20% train-test split uses only $20\% \cdot 185 = 37$ of positive labels to construct graph signal priors. For each graph, we experiment with all three split ratios and average evaluation measures between them. To make sure that comparisons between different approaches are not affected by sampling variance, we use seeded sampling to select training nodes.

4.4.2 Measures

Our experiments are focused on two types of fairness-aware objectives: maximizing the pRule of posterior scores and achieving high pRule values while preserving high posterior score quality. Both consider a recommender system setting, where we start from graph signal priors and graph filters are used to obtain node score posteriors. These posteriors then provide a granular understanding on how much nodes pertain to an attribute shared by nodes with non-zero priors. To assess how well approaches achieve high posterior quality and fairness we respectively calculate the AUC and pRule of their posteriors over the evaluation subset of nodes.

An overview of information flow during evaluation is presented in Figure 3. It is worth noting that, although only test nodes are used to calculate measures, the sensitive-aware approaches explored in this work require knowledge of sensitive attributes for all nodes, which include the test ones. Nonetheless, evaluation remains robust in the sense that training data are not overfitted as long as fairness-aware approaches are not provided with information about *which* nodes are used for testing. For example, approaches such as *Mult* induce perfect fairness when considering all nodes, but this fairness needs to generalize to subsets of graph nodes, such as test ones.

4.4.3 Approach comparison

Our experiments output one value for each measure per combination of graph, data set, fairness-aware approach and graph filter, for a total of $12 \cdot 8 \cdot 8 = 768$ combinations. To

^{9.} In our previous work, we performed robust fairness evaluation by keeping the minimum pRule between considering all vs considering only test nodes. However, in practice, we found that the pRule of test nodes dominates this quantity.

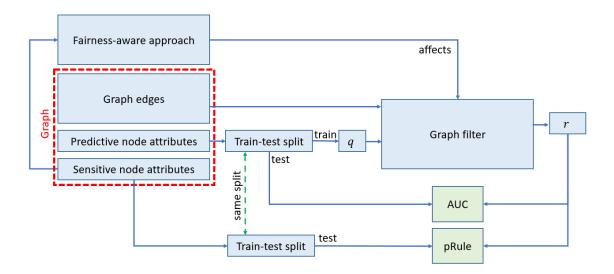


Figure 3: The process of assessing the AUC and pRule of a fairness-aware approach on a graph filter for a given graph and train-test split ratio.

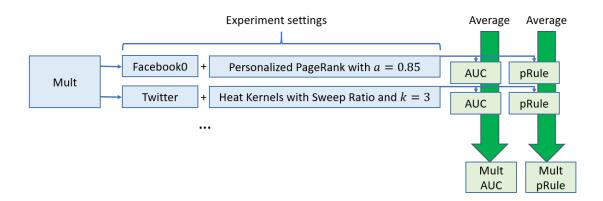


Figure 4: Averaging the outcome of Mult AUC and pRule values across experiment settings.

help gather insights from these results, we opt to delegate them to Appendix B and instead extract high-level summary statistics that make it easy to compare different approaches.

To this end, we extract three summary statistics across all graphs and graph filters, depending on the type of post-processing (no post-processing or the sweep ratio). These statistics are: a) the average AUC of approaches and their statistical significant differences, b) the average pRule of approaches and their statistical significant differences and c) the percentage of experiments in which pRule achieves the constraint of 80% or more. As an example, in Figure 4 we demonstrate the process of obtaining the average AUC and pRule values for the Mult approach, where the same procedure is followed for all others too.

We assert which approaches differ significantly from the rest by following a well-known procedure for multi-approach comparison (Demšar, 2006; Garcia and Herrera, 2008; Derrac et al., 2011). In detail, we first employ a Friedman test with p-value <0.001 to assert that at least one approach differs from the others with statistical significance. Provided that the Friedman test rejects the null hypothesis that all approaches produce similar outcomes, we then use a Nemenyi post-hoc test to compare individual approaches. The latter ranks approaches for each measure (assigning rank 1 to the best approach, 2 to the second best and so on), reports the average rank across data sets and base ranking algorithms and outputs a critical difference with p-value at most 0.05, where average rank differences greater than it imply statistical significance at that p-value level. ¹⁰

5. Experiment Results

Following the evaluation methodology detailed in the previous section, we compare the fairness-aware approaches proposed in this work (FairEdit, FairEdic-C) to the ones of our previous paper (FairPers, FairPers-C), an intuitive baseline (Mult) and two similar methods employed in the literature (FairWalk, LFPRO). This comparison involves running experi-

^{10.} The Nemenyi test is weaker than most other post-hoc tests, such as the Holm test, in that it does not necessarily discover all of statistically significant differences. But we prefer it on merit that it reports a clear order between approaches and a universal criterion to determine statistically significant differences. The p-value for this test is not selected to be too tight to offset its weakness.

	N	o Post-pro	ocessing	Sweep Ratio					
	AUC	\mathbf{pRule}	$ m pRule \geq 80\%$	AUC	\mathbf{pRule}	$\mathrm{pRule} \geq 80\%$			
None	.76 (3.0)	.52 (6.5)	.21	.77 (3.2)	.51 (6.8)	.23			
Mult	.73 (4.3)	.68(5.7)	.33	.73(5.0)	.68(5.8)	.35			
LFPRO	.67 (5.4)	.83(4.2)	.75	.67 (6.0)	.82(4.7)	.69			
FairWalk	.75 (3.3)	.47(6.8)	.17	.75 (3.9)	.54(6.3)	.31			
FairPers	.66 (5.8)	.93(2.2)	.92	.66 (5.3)	.95(2.2)	.96			
FairPers-C	.72 (4.6)	.86(4.3)	.79	.72 (4.7)	.88 (4.1)	.92			
FairEdit	.69 (4.6)	.92(2.2)	.88	.70 (4.6)	.94(2.0)	.92			
FairEdit-C	.72 (4.1)	.86(4.2)	.90	.76 (3.3)	.88 (4.2)	.94			

Table 5: Average AUC, pRule and fraction of experiments achieving 80% pRule for fairness-aware approaches (higher are better). Average Nemenyi ranks for measures in parenthesis (smaller are better), where rank differences greater than 1.6 are statistically significant. Different results are presented depending on the post-processeing of the base graph filter.

ments for all combinations of graph filters and data sets and exploring the resulting AUC and pRule values presented in Appendix B.

Table 5 forms a high-level summary of experiment results. In particular, it reports the average of evaluation measures across all experiments for both types of post-processing. It also presents the Nemenyi ranks for multiway comparison between approaches. We remind that lower ranks indicate approaches performing better for the particular measure, where rank differences exceeding a critical difference indicate statistically significant improvements. For our both sets of comparisons, the critical difference indicating statistical significance is calculated as approximately 1.6. For example, base graph filters (the approach dubbed None) with no post-processing are assigned 3.0 AUC rank, which indicates that they outperform LFPRO in this regard with statistical significance, as the latter's corresponding rank is 5.4 > 3.0 + 1.6. All numbers rounded to their two most important digits.

5.1 Fairness maximization

Compared to base graph filters, the approaches LFPRO, FairPers and FairEdit reduce the average AUC by at least 7% with statistical significance, where LFPRO suffers from the greatest posterior quality reduction. Posterior quality loss for all three methods arises from their attempt to maximize fairness while only partially preserving posteriors.

Among these approaches, FairEdit exhibits at least 3% greater average AUC compared to the second-best FairPers at the cost of 1% pRule. Although these differences are not statistically significant, they nonetheless identify FairEdit as the preferred method to eliminate disparate impact in real-world applications. Looking at detailed experiment results, FairPers and FairEdit outperform each other in different experiments, which suggests that there is added value in better controlling prior retention in future research, which we find in Appendix C to contribute the most in this improvement.

In terms of disparate impact elimination, LFPRO yields significantly smaller pRule by 9% compared to the two prior editing approaches, which corroborates our assumption that

respecting the graph's structure by editing priors can better satisfy posterior objectives than directly editing those through uniformly skewing mechanisms.

5.2 Achieving fairness constraints

The Mult, FairPers-C and FairEdit-C approaches in large part preserve posterior score quality, with their AUC seeing a reduction by at most by 5% compared to base graph filters, while improving their pRule by large margins. This achievement can be attributed to prior editing approaches placing significance to pRule improvements only up to the target level, whereas Mult affects posterior node score comparisons only between sensitive nodes and the rest of the graph, which in most graphs at introduces only a small fraction of erroneously high node posterior scores compared to the number of nodes with non-positive labels.

Overall, prior editing approaches achieve over 80% pRule both on average and a remarkably large portion of experiments, which hints at their robust generalization capabilities. By contrast, Mult yields significantly lower pRule on average with statistical significance regardless of the and does not meet the fairness criterion in most experiments. In fact, the constrained approaches exhibit similar -and on average higher- pRule than even LFPRO, while at the same time enjoying sigificantly higher AUC values.

5.3 Sweep ratio improvements

We previously mentioned that the sweep ratio is often employed to improve posterior quality for graph filters. In line with this consensus, the average AUC of approaches when this kind of post-processing is used is at least equal to or greater than the one with no post-processing. Although this improvement is often marginal (\sim 1%), in the case of FairEdit-C the sweep ratio improves AUC by 4%. More importantly, this mechanism induces 2% pRule improvement for all personalization editing approaches, which suggests that there is likely merit in employing it in new settings. Intuitively, improvements thanks to the sweep ratio can be attributed to increasing the posteriors of low-scored nodes so that it becomes easier to move a portion of those to sensitive ones without affecting pairwise node score comparisons.

5.4 Experiment outliers

Of the fairness-aware approaches, FairWalk performs similarly or worse than base graph filters. However, result details reveal that, in some experiments, it manages to improve the pRule. Hence, we argue that this method's failure lies more in sensitive nodes not being randomly mixed in all graphs we experiment on but forming structurally close communities; this prevents the fair random walk formulation from visiting sensitive and non-sensitive neighbors with equal probabilities, as for most nodes only one of those kinds of neighbors is available.

Lastly, we overview some noticeable outliers in experiment outcomes. First, Mult and LFPRO sometimes outperform our approaches in preserving AUC (such as the near-100% AUC of the Maven data set). Our understanding of these phenomena lies in these two approaches potentially affecting only the sensitive nodes without respecting propagation mechanisms, which lets them find success in specific cases. Another case that warrants at-

tention is the inability of our approach to meet 80% pRule for the DBLP graph when base graph filters involve personalized pagerank. We attribute this finding to further approximation terms being needed given that graph's eigenvalue ratio, but further investigation is needed.

6. Conclusions and Future Work

In this work we explored the concept of editing graph signal priors to locally optimize fairness-aware graph filter posterior objectives. To this end, we proposed an editing mechanisms with few parameters and showed that, given its parameter gradients' lose approximation of fairness-aware objective optimization slopes, it reaches local optima without overfitting posterior node scores. We then experimented on a large number of real-world graphs with various graph filters and signal priors, where we used our approach to produce high-quality posterior scores in terms of AUC that are fair (enough). Our findings suggest that, compared to other fairness-aware methods, prior editing is able to better reduce disparate impact or meet reduction constraints while in large part preserving the predictive quality of posteriors.

Promising applications of this work could involve bringing fairness to decoupled graph neural network architectures, where feature extraction and propagation with graph filters are performed separately. Our theoretical framework on optimizing posteriors is also general enough to warrant application on other types of objectives, such as maximizing smooth relaxations of AUC. Finally, we are interested in further exploring the assumption that prior editing in its current form exhibits enough degrees of freedom and aim to provide a bound on its necessary number of terms in future work.

Acknowledgments

This work was partially funded by the European Commission under contract numbers H2020-860630 NoBIAS and H2020-825585 HELIOS.

Appendix A. Parameter Tuning

In this appendix we detail the algorithm we developed as part of the *pygrank* library to optimize the parameters of graph signal prior editing mechanisms (FairPers, FairEdit and their constrained variations) in a given hypercube without derivating posterior objectives.

Our algorithm involves cycling through a list of parameters θ and optimizing a loss function $\ell(\theta)$ by finding the best permutation around each one. For the prior editing approach of this work, the parameters are $\theta = \{a_S, a_{S'}, b_S, b_{S'}\}$ and evaluating the loss function $\ell(\theta) = \mathcal{L}(H(W)q_{est})$ once for the respective prior estimation q_{est} involves running a graph filter; for the graphs with millions of nodes, this requires approximately 1-2 sec on the 2.2 Ghz 4-core CPU with hyperthreading used in experiments, after a preprocessing operation obtains a common sparse version of the adjacency matrix (sparse matrices require $O(|\mathcal{E}|)$ amortized time in big-o notation to perform one-hop propagations by multiplying graph signals). Though this time is not prohibitively large, to run the large number of experiments in this work we design the optimization algorithm with the goal of performing only a small number of parameter loss function evaluations. Intuitively, it is equivalent to moving the center of the selected rectangle chosen for each parameter based on subsequent selections of other parameters. So, at the time where the parameter's permutation breadth shrinks to the size of its intended rectangle, potential combinations with permutations of other parameters have been considered too.

The implemented algorithm is a variation of divided rectangles (DIRECT) (Finkel and Kelley, 2006) that, instead of keeping many candidate rectangles to divide, keeps only one, though of larger width than the partition. This practice corresponds to the shrinking radius technique proposed for non-convex block coordinate optimization (Lyu, 2020), although the two are not mathematically equivalent due to the finite sum of rectangle widths that limits the optimization within the hypercube of searched parameters,

In detail, we start from the center of parameter square bounds and cycle through parameters i. For each of those parameters, we consider the maximal range $\Delta\theta[i]$ in which

```
Input: parameter loss \ell(\theta), parameter square bound vectors \theta_{\max}, \theta_{\min} \in \mathbb{R}^K, tolerance \epsilon, line partitions part, range contraction T

Output: near-optimal vector of K parameters \theta \leftarrow (\theta_{\max} + \theta_{\min})/2

\Delta \theta \leftarrow \theta_{\max} - \theta_{\min}

err \leftarrow [\infty] \times K

i \leftarrow 0

while \max_i err[i] > \epsilon do

u_i \leftarrow \text{unit vector with } 1 at element i, 0 elsewhere \Theta \leftarrow \{\theta + u_i \cdot \Delta \theta[i] \cdot (p/part - 1) \mid p = 0, 1, \dots, 2 \, part\}

\theta \leftarrow \arg\min_{\theta \in \Theta} \ell(\theta)

err[i] \leftarrow \max_{\theta \in \Theta} \ell(\theta) - \min_{\theta \in \Theta} \ell(\theta)

\Delta \theta[i] \leftarrow \Delta \theta[i]/T

i \leftarrow (i+1) \, mod \, K

return \theta
```

Algorithm 1: Parameter tuning

to search for new solutions and partition it uniformly to 2 part + 1 points. These form a set Θ of parameter perturbations, out of which we select the ones minimizing the loss. Finally, we contract the range in which to search parameters by dividing it with the value K and move on to the next parameter. We stop cycling through parameters when the max loss difference between perturbations are smaller than the given tolerance. This process is outlined in Algorithm 1.

If the objective $\ell(\theta)$ is Lipshitz continuous with Lipshitz constant $\sup \|\nabla \ell(\theta)\| < \infty$, it is easy to see that the division of the parameter permutation radius by T every K iterations lets the algorithm run in amortized time $O\big(K(\operatorname{run}\,\ell(\theta))\log_T(\|\theta_{\max}-\theta_{\min}\|_{\infty}\sup\|\nabla\ell(\theta)\|)\big)$. For experiments in this work, and to reduce running time as much as possible, we empirically selected the parameters $part=2, T=2, \epsilon=0.01$, which yielded near-identical results to employing the more granular parameters $part=4, T=1.3, \epsilon=0.001$ in a randomly selected subset of 15 experiment settings.

Appendix B. Detailed Experiment Results

	No	ne	Μu	ılt	LFP	RO	FairI	Pers	FairF	ers-C	Fairl	Edit	FairE	Edit-C	FairV	Valk
	AUCp	Rule	AUCp	Rule	AUCp	Rule				pRule						
					Grap	h Filt	er PP	R.85								
ACM	.71	.48	.71	.74	.74	.92	.73	.95	.76	.95	.75	.81	.74	.92	.71	.81
Amazon	.81	.01	.79	.82	.59	.93	.75	.99	.73	.88	.68	.98	.72	.83	.81	.03
Ant	.78	.59	.77	.78	.79	.87	.58	.99	.77	.86	.76	.95	.78	.87	.77	.79
Citeseer	.84	.13	.83	.74	.77	.73	.69	.91	.82	.83	.87	.43	.82	.82	.84	.16
DBLP	.92	.75	.92	.77	.92	.86	.86	.60	.86	.60	.89	.66	.89	.66	.92	.09
Facebook0	.57	.91	.54	.74	.54	.76	.56	.90	.57	.83	.53	.98	.58	.87	.54	.77
Facebook686	.52	.94	.51	.74	.50	.70	.48	.93	.52	.90	.50	.99	.52	.90	.53	.95
Log4J	.78	.77	.77	.75	.76	.83	.62	.99	.68	.89	.74	.94	.76	.92	.77	.68
Maven	1.00	.50	1.00	.83	1.00	.84	.49	.94	.82	.91	.86	.91	.86	.91	1.00	.25
Pubmed	.90	.45	.82	.77	.81	.97	.64	.99	.78	.83	.86	.69	.79	.86	.88	.65
Squirrel	.64	.36	.64	.72	.64	.44	.67	.97	.73	.87	.75	.87	.75	.83	.65	.33
Twitter	.97	.13	.48	.80	.31	.84	.58	.99	.68	.82	.58	.99	.68	.83	.87	.33
Graph Filter PPR.99																
ACM	.70	.53	.70	.88	.76	.89	.65	.92	.73	.88	.72	.99	.73	.90	.70	.76
Amazon	.83	.01	.80	.95	.68	.84	.74	.99	.69	.91	.68	1.00	.71	.82	.82	.03
Ant	.71	.92	.70	.94	.70	.96		.99	.68	.94	.54	.99	.57	.96	.72	.76
Citeseer	.84	.40	.83	.90	.85	.84	.66	.95	.79	.81	.52	1.00	.82	.81	.84	.23
DBLP	.92	.66	.92	.73	.92	.84	.91	.70	.91	.70	.92	.68	.91	.69	.92	.09
Facebook0	.56	.93	.56	.96	.56	.96	.58	.97	.59	.94	.53	.97	.47	.88	.54	.77
Facebook686	.51	.98	.51	.98	.51	.98	.51	.99	.51	.98	.51	.99	.51	.98	.51	.96
Log4J	.74	.94	.74	.95	.74	.97	.65	.99	.65	.99	.67	.97	.56	.94	.74	.58
Maven	.98	.77	.98	.99	.98	1.00	.46	1.00	.93	.80	.87	.83	.89	.80	.98	.21
Pubmed	.79	.75	.73	.98	.72	.97	.55	1.00	.73	.86	.55	1.00	.75	.85	.70	.97
Squirrel	.63	.46	.63	.84	.63	.59	.64	.96	.70	.72	.71	.90	.70	.74	.63	.52
Twitter	.74	.69	.57	.98	.39	.98	.59	.99	.66	.85	.59	.99	.69	.81	.73	.15
					Gra	ph F	ilter H	K3								
ACM	.68	.49	.68	.41	.66	.91	.72	.99	.73	.94	.71	.96	.73	.93	.68	.60
Amazon	.88	.01	.87	.49	.46	.84		.92	.89	.79	.73	.97	.87	.88	.88	.03
Ant	.74	.48	.73	.41	.67	.88	.75	.95	.77	.86	.67	.99	.75	.89	.74	.86
Citeseer	.77	.12	.77	.42	.63	.89	.63	.96	.73	.89	.46	.99	.75	.87	.77	.15
DBLP	.89	.74	.89	.95	.89	.96	.89	.75	.88	.75	.90	.75	.90	.75	.89	.07
Facebook0	.56	.89	.52	.41	.41	.48	.53	.87	.52	.79	.47	.95	.49	.88	.54	.73
Facebook686	1	.98	.51	.38	.52	.53	.48	.85	.47	.87	.50	.96	.53	.88	.54	.91
Log4J	.70	.70	.70	.39	.66	.85	.64	.97	.67	.97	.65	.99	.69	.96	.70	.60
Maven	.97	.33	.97	.45	.94	.95	.77	.97	.77	.97	.80	.97	.80	.97	.97	.26
Pubmed	.83	.36	.77	.42	.67	.95	.71	.97	.75	.90	.58	.98	.75	.87	.82	.59
Squirrel	.64	.40	.64	.31	.64	.47	.67	.92	.67	.83	.68	.93	.68	.87	.64	.32
Twitter	.90	.05	.69	.43		.75		.90	.78	.77	.58	.99	.71	.86	.87	.15
							ilter H									
ACM	.68	.49	.68	.53		.98		.99	.73	.93	.72	.96		.94	.68	.60
Amazon	.88	.01	.87	.64		.96		.97	.87	.85	.75	.97	.87	.86	.88	.03
Ant	.74	.48	.74	.54	.72	.94		.98	.77	.85	.66	.99	.76	.88	.74	.86
Citeseer	.77	.12	.77	.55	.66	.82	.69	.95	.73	.89	.46	1.00	.75	.86	.77	.15
DBLP	.89	.74	.89	.85	.89	.93		.69	.85	.69	.90	.69	.90	.69	.89	.07
Facebook0	.56	.89	.53	.58	.51	.62	.55	.91	.52	.85	.47	.99	.49	.89	.54	.73
Facebook686	1	.98	.51	.55	.51	.52		.86	.50	.86	.51	.96	.52	.86	.54	.91
Log4J	.70	.70	.70	.52	.69	.89		.96	.68	.93	.64	.98	.68	.96	.70	.60
Maven	.97	.33	.97	.62	.97	.92	.78	.96	.78	.96	.84	.96	.84	.96	.97	.26
Pubmed	.83	.36	.78	.57	.73	.83		1.00	.75	.92	.58	.99	.74	.85	.82	.59
Squirrel	.64	.40	.64	.42	.64	.44		.84	.67	.74	.68	.91	.68	.84	.64	.32
Twitter	.90	.05	.71	.59	.42	.97	.61	.94	.73	.78	.58	.99	.69	.83	.87	.15

Table 6: Experiments for base graph filters without post-processing.

	No	ne	Mu	ılt	LFPI	RO	FairI	Pers	FairF	Pers-C	Fair	Edit	FairE	Edit-C	FairV	Valk
	AUCp	Rule	AUCp	Rule	AUC p	Rule	AUC	Rule	AUC	pRule	AUCI	Rule	AUC	pRule	AUCp	Rule
	'				Graph	Filte	er PPI	R.85S								
ACM	.70	.32	.70	.80	.72	.90	.62	.99	.69	.87	.69	.87	.68	.83	.70	.82
Amazon	.79	.01	.76	.82	.57	.93	.71	.96	.82	.82	.67	1.00		.82	.81	.03
Ant	.78	.60	.77	.77	.80	.86	.78	.97	.78	.89	.77	.96	.77	.86	.77	.79
Citeseer	.84	.14	.83	.78	.80	.75	.85	.91	.85	.84	.84	.40	.85	.84	.84	.15
DBLP	.90	.58	.90	.81	.89	.77	.89	1.00	.76	.79	.88	.77	.88	.76	.92	.09
Facebook0	.60	.94	.56	.72	.57	.73	.57	.86	.57	.81	.59	.96		.91	.59	.96
Facebook686		.93	.52	.72	.51	.68	.53	.93	.53	.90	.53	.97	.56	.88	.55	.94
Log4J	.76	.76	.76	.73	.74	.81	.75	.98	.75	.97	.75	.91	.75	.86	.78	.78
Maven	1.00	.48		.81	1.00	.83	.96	.99	.96	.98	1.00	.97		.97	1.00	.90
Pubmed	.92	.46	.84	.74	.80	.96	.88	.98	.88	.93	.91	.74		.86	.88	.65
Squirrel	.64	.37	.64	.75	.64	.47	.67	.95	.66	.90	.66	.94		.81	.64	.38
Twitter	.98	.13	.34	.76	.31	.78	.17	.92	.50	.92	.45	.99	1.00	.83	.95	.31
Graph Filter PPR.99S																
ACM	.68	.37	.68	.91	.73	.90	.56	.99	.61	.87	.68	.89		.81	.68	.86
Amazon	.79	.01	.77	.96	.64	.83	.61	.96	.64	.84	.65	1.00		.80	.79	.03
Ant	.78	.93	.77	.95	.77	.97	.77	.98	.74	.93	.36	.99		.96	.72	.76
Citeseer	.85	.41	.84	.92	.86	.86	.84	.94	.85	.81	.84	1.00		.81	.85	.40
DBLP	.89	.54	.89	.79	.88	.74	.84	1.00	.90	.78	.88	.63		.72	.89	.36
Facebook0	.64	.97	.61	.96	.61	.96	.60	.99	.59	.99	.53	1.00		.86	.64	.98
Facebook686	1	.98	.52	.97	.52	.97	.52	.98	.51	.97	.55	1.00		.98	.51	.96
Log4J	.76	.90	.74	.94	.74	.97	.75	.98	.75	.96	.60	.99		.98	.55	.54
Maven	1.00	.73		.99	1.00	.99	.97	.98	.96	.93	.98	.97	.99	.94	1.00	.95
Pubmed	.91	.75	.84	.98	.83	.97	.89	.99	.88	.90	.89	.98		.84	.70	.97
Squirrel	.64	.47	.64	.89	.64	.64	.62	.93	.62	.77	.62	.95		.78	.64	.49
Twitter	.98	.60	.43	.97	.40	.97	.65	.99	.96	.83	.55	1.00	1.00	.80	.73	.15
1 C2 f		40	0=	4.4			lter H		0.0	0=		0.0	0.0	0.0		0.1
ACM	.67	.40	.67	.44	.65	.94	.66	.96	.66	.87	.69	.99	.66	.92	.67	.81
Amazon	.87	.01	.86	.49	.44	.85	.15	.98	.78	.91	.84	1.00		.92	.88	.03
Ant	.74	.48	.73	.40	.70	.86	.73	.93	.74	.84	.75	.98		.88	.74	.86
Citeseer	.77	.12	.77	.42	.64	.90	.73	.95	.81	.89	.59	.99		.88	.77	.13
DBLP	.88	.87	.88	.92	.88	.91	.85	1.00	.85	.98	.90	.99	.90	.98	.88	.26
Facebook0	.58	.90	.52	.39	.41	.45	.53	.78	.58	.84	.53	.95		.86	.54	.73
Facebook686	1	.99	.51	.36	.51	.49	.52	.79	.53	.91	.51	.93		.93	.55	.95
Log4J	.70	.69	.69	.39	.64	.86	.62	.96	.61	.92	.76	.98		.96	.70	.83
Maven	.97	.27	.97	.43	.94	.96	.43	1.00	.43	1.00	.99	1.00		.99	.97	.26
Pubmed	.83	.36	.77	.41	.66	.95	.87	.97	.81	.89	.78	.98	.90	.87	.82	.59
Squirrel	.64	.41	.64	.32	.64	.49	.66	.91	.65	.85	.67	.96		.91	.64	.38
Twitter	.90	.05	.68	.41	.18	.67	.02	.94	.51	.96	.45	.98	1.00	.88	.88	.10
ACM	67	.38	67	E 7		.97	lter H .67		cc	.83	c o	.99	.66	0.1	67	76
ACM Amazon	.67 .86	.01	.67 .85	.57 .64	.66 .56	.96	.32	.96 .98	.66 .84	.87	.68 .70	1.00		.91 .87	.67 .88	.76 .03
Amazon	.74	.48	.74	.54	.72	.90 .94	.76	.90	.76	.84	.75	.98		.88	.00 .74	.03 .79
Citeseer	.77	.12		.55	.65	.84	.85	.95	.77	.89	.70	.90		.87	.74	.13
DBLP	.88	.12	.88	.55 .91	.88	.82	.85	.98	.85	.92	.90	.99 .97		.96	.88	.13
Facebook0	.59	.91	.54	.55	.52	.57	.57	.86	.60	.87	.51	1.00		.85	.54	.73
Facebook686	1															
	.55 .70	.99 .69	.52 .69	.53 .53	.50 .67	.50 .90	.55 $.72$.95 .93	.56 .70	.86 .86	.53 .75	.94 .98		.96 .95	.55 .70	.95 .86
Log4j Maven	.70	.09	.09		.07	.90 .89	.72	.93 .99	.70	.99	.73	.98		.98	.70	.26
Pubmed				.60												
Pubmed Squirel	.83	.36	.78	.56	.67	.96	.86 67	1.00	.88	.87	.79 66	.98		.86	.82	.59
Squirei Twitter	.64 .90	.42 .05	.64 .71	.43 .56		.45 .86	.67 .41	.92 .96	.65 .87	.79 .83	.66 .47	.96	.67 1.00	.88 .86	.64 .88	.39 .10
ı wittei	.90	.03	.11	.50	.55	.00	.41	.90	.01	.00	.41	.90	1.00	.00	.00	.10

Table 7: Experiments for base graph filters with sweep ratio post-processing.

Appendix C. Ablation Study

In this appendix we investigate the effect of adding granular retention of original posteriors in Equation 11. To this end, we perform additional experiments on the FairEdit and FairEdit-C approaches, in which we fix the parameter value $a_0=0$ to remove explicit retention of original priors—and hence original posteriors. In Table C we present the outcome of the two new variations under the names FairEdit0 and FairEdit0-C respectively. We remind that FairPers0 variants differ from FairEdit in that they use error-based instead of difference-based skewing of posteriors and KL-divergence from original posteriors as a training objective instead of mean absolute error. And FairPers variants further differ from FairPers0 in that they explicitly introduce a degree of freedom towards maintain original posteriors.

Table C summarizes a multiway comparison between prior editing approaches. Overall, all three types of prior editing yield similar average AUC and pRule values and Nemenyi ranks when used to satisfy the 80% pRule constraint. On the other hand FairEdit0 manages to improve AUC by only a small margin (\sim 1%) compared to FairPers when maximizing fairness. This indicates that concerns over posterior outliers and the interpretability of stochastic surrogate models are valid yet not too prevalent in the graphs we experiment on. At the same time, small improvements do not retract from the added value of accounting for these phenomena, as posterior outliers could arise in different graphs and the more intuitive interpretation of prior editing parameters can prove useful in terms of explainability. Finally, FairEdit's 3-4% AUC improvement compared to FairPers can be in large part be attributed to partially retaining priors, as recommended by Theorem 3.

	N	o Post-pro	ocessing	Sweep Ratio					
	AUC	\mathbf{pRule}	$ m pRule \geq 80\%$	AUC	\mathbf{pRule}	$ m pRule \geq 80\%$			
FairPers	.66 (4.5)	.93 (2.5)	.92	.66 (5.3)	.95 (2.7)	.96			
FairPers-C	.72 (3.2)	.86(4.6)	.79	.72 (4.7)	.88 (4.4)	.92			
FairEdit	.69 (4.0)	.92(2.4)	.88	.70 (4.6)	.94(2.4)	.92			
FairEdit-C	.72 (2.7)	.86(4.4)	.90	.77 (3.3)	.88(4.5)	.94			
FairEdit0	.67 (4.1)	.93(2.3)	.88	.67 (4.6)	.95(2.1)	.92			
FairEdit0-C	.72(2.5)	.86(4.6)	.90	.77 (3.3)	.87(4.9)	.94			

Table 8: Average AUC, pRule and fraction of experiments achieving 80% pRule for prior editing approaches (higher are better). Average Nemenyi ranks for measures in parenthesis (smaller are better), where rank differences greater than 1.3 are statistically significant. Different results are presented depending on the post-processeing of the base graph filter.

	FairE	dit0	FairE	dit0-C	FairE	Edit0	FairE	dit0-C
	AUCr	Rule	AUC	pRule	AUCı	Rule	AUC	pRule
			R.85	-			R.85S	
Acm	.75	.81	.73	.94	.69	.82	.68	.83
Amazon	.69	.98		.83		1.00	.80	.82
Ant	.75	.96	Į.	.88		.97	.77	.86
Citeseer	.87	.42		.82	.84	.40	.85	.83
Dblp	.89	.66		.66	.88	.77	.88	.76
Facebook0	.54	.98		.88	.54	.99	.62	.91
Facebook686		.99		.91	.50	1.00	.56	.89
Log4j	.74	.93		.92	.75	.91	.75	.86
Maven	.85	.91	.85	.91	1.00	.97	1.00	.97
Pubmed	.86	.69		.86	.91	.74	.91	.85
Squirel	.75	.87	.75	.85	.66	.94	.66	.81
Twitter	.58	.99	ļ.	.82	.58	1.00	1.00	.80
	.00		R.99	.02			R.99S	
Acm	.73	.97		.90	.67	.87	.68	.81
Amazon	.69	1.00		.81	.64	1.00	.79	.81
Ant	.53	.99		.97	.31	.99	.39	.97
Citeseer	.48	.99	!	.81	.47	1.00	.85	.81
Dblp	.92	.68		.69	.88	.63	.88	.72
Facebook0	.52	.99		.88	.53	1.00	.51	.85
Facebook686	1	.99	Į.	.98	.51	1.00	.48	.98
Log4j	.66	.97		.94	.61	.99	.50	.98
Maven	.87	.83		.80		.97	.99	.94
Pubmed	.54	1.00		.85	.89	.98	.93	.84
Squirel	.71	.90	.70	.73	.62	.95	.62	.78
Twitter	.59	.99	.69	.81	.55	1.00	1.00	.80
1 WILLEI	.00		K3	.01	.55		$\frac{1.00}{3S}$	
Acm	.72	.95		.93	.62	1.00	.66	.91
Amazon	.76	.99		.87	.67	1.00	.88	.84
Ant	.70	1.00	.76	.88	.67	.99	.76	.88
Citeseer	.45	1.00		.86	.46	1.00	.85	.87
Dblp	.90	.75	.90	.75	.90	.99	.90	.98
Facebook0	.47	1.00		.81	.47	1.00	.59	.82
Facebook686		.98	I	.89	.47	.97	.55	.93
Log4j	.65	.99		.96		.98	.76	.9 3
Maven	.79	.99		.97	.99	1.00	.99	.99
Pubmed	.53	1.00		.86	.53	1.00	.91	.86
Squirel	.68	.93	.68	.87	.67	.96	.67	.91
Twitter	.59	.99		.83	.64		1.00	.84
1 witter	.09		K7	.00	.04		$\frac{1.00}{67S}$.04
A area	.72			.93	60			01
Acm		.96				1.00	.66	.91
Amazon	.77	.99	I	.85	.67	1.00	.87	.83
Ant	.65	.98		.88		1.00	.75	.88
Citeseer	.44	1.00		.85	.46	1.00	.85	.86
Dblp Eaghaele0	.90	.69		.69	.90	.97	.90	.96
Facebook0	.47	.99		.83		1.00	.63	.84
Facebook686		.99	I	.86		.98	.55	.96
Log4j	.64	.98		.96		.97	.76	.94
Maven	.84	.96	l	.96		.98	.99	.98
Pubmed	.54	1.00		.85	.53	1.00	.90	.85
Squirel	.68	.92	I	.84		.96	.67	.88
Twitter	.58	.99	.71	.81	.64	1.00	1.00	.83

Table 9: Experiments of FairEdit with $a_0=0$ for both types of post-processing

References

- LINQS statistical relational learning group datasets. https://linqs.soe.ucsc.edu/data.
- Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In 2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), pages 475–486. IEEE, 2006.
- Reid Andersen, Fan Chung, and Kevin Lang. Local partitioning for directed graphs using pagerank. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 166–178. Springer, 2007a.
- Reid Andersen, Fan Chung, and Kevin Lang. Using pagerank to locally partition a graph. *Internet Mathematics*, 4(1):35–64, 2007b.
- Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. Fast incremental and personalized pagerank. *Proceedings of the VLDB Endowment*, 4(3), 2010.
- Amine Benelallam, Nicolas Harrand, César Soto-Valero, Benoit Baudry, and Olivier Barais. The maven dependency graph: a temporal graph-based representation of maven central. In 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), pages 344–348. IEEE, 2019.
- Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong, Ed H Chi, et al. Fairness in recommendation ranking through pairwise comparisons. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2212–2220, 2019.
- Dan Biddle. Adverse impact and test validation: A practitioner's guide to valid and defensible employment testing. Gower Publishing, Ltd., 2006.
- Asia J Biega, Krishna P Gummadi, and Gerhard Weikum. Equity of attention: Amortizing individual fairness in rankings. In *The 41st international acm sigir conference on research & development in information retrieval*, pages 405–414, 2018.
- Anders Björner and László Lovász. Chip-firing games on directed graphs. *Journal of algebraic combinatorics*, 1(4):305–328, 1992.
- Avishek Bose and William Hamilton. Compositional fairness constraints for graph embeddings. In *International Conference on Machine Learning*, pages 715–724, 2019.
- Toon Calders and Sicco Verwer. Three naive bayes approaches for discrimination-free classification. Data Mining and Knowledge Discovery, 21(2):277–292, 2010.
- Christine Cassel and Andrew Bindman. Risk, benefit, and fairness in a big data world. Jama, 322(2):105–106, 2019.
- David Chalupa. A memetic algorithm for the minimum conductance graph partitioning problem. arXiv preprint arXiv:1704.02854, 2017.

- Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pages 1725–1735. PMLR, 2020.
- Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Biq data*, 5(2):153–163, 2017.
- Fan Chung. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 9(1):1–19, 2005.
- Fan RK Chung and Fan Chung Graham. Spectral graph theory. Number 92. American Mathematical Soc., 1997.
- Enyan Dai and Suhang Wang. Fairgnn: Eliminating the discrimination in graph neural networks with limited sensitive attribute information. arXiv preprint arXiv:2009.01454, 2020.
- Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.
- Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation, 1(1):3–18, 2011.
- Hande Dong, Jiawei Chen, Fuli Feng, Xiangnan He, Shuxian Bi, Zhaolin Ding, and Peng Cui. On the equivalence of decoupled graph convolution network and label propagation. arXiv preprint arXiv:2010.12408, 2020.
- Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pages 259– 268, 2015.
- Daniel E Finkel and CT Kelley. Additive scaling and the direct algorithm. *Journal of Global Optimization*, 36(4):597–608, 2006.
- Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics reports*, 659:1–44, 2016.
- Salvador Garcia and Francisco Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9(Dec):2677–2694, 2008.
- Adnan Gavili and Xiao-Ping Zhang. On the shift operator, graph frequency, and optimal filtering in graph signal processing. *IEEE Transactions on Signal Processing*, 65(23): 6303–6318, 2017.
- Lise Getoor. Link-based classification. In Advanced methods for knowledge discovery from complex data, pages 189–207. Springer, 2005.

- James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.
- Darko Hric, Richard K Darst, and Santo Fortunato. Community detection in networks: Structural communities versus ground truth. *Physical Review E*, 90(6):062805, 2014.
- Darko Hric, Tiago P Peixoto, and Santo Fortunato. Network structure, metadata, and the prediction of missing nodes and annotations. *Physical Review X*, 6(3):031038, 2016.
- Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. Combining label propagation and simple models out-performs graph neural networks. arXiv preprint arXiv:2010.13993, 2020.
- FO Isinkaye, YO Folajimi, and BA Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273, 2015.
- Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1):1–33, 2012.
- Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.
- Jon Kleinberg, Jens Ludwig, Sendhil Mullainathan, and Ashesh Rambachan. Algorithmic fairness. In *Aea papers and proceedings*, volume 108, pages 22–27, 2018.
- Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. arXiv preprint arXiv:1810.05997, 2018.
- Kyle Kloster and David F Gleich. Heat kernel based community detection. In *Proceedings* of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1386–1395, 2014.
- Emmanouil Krasanakis, Eleftherios Spyromitros-Xioufis, Symeon Papadopoulos, and Yiannis Kompatsiaris. Adaptive sensitive reweighting to mitigate bias in fairness-aware classification. In *Proceedings of the 2018 World Wide Web Conference*, pages 853–862, 2018.
- Emmanouil Krasanakis, Symeon Papadopoulos, and Ioannis Kompatsiaris. Applying fairness constraints on graph node ranks under personalization bias. In *International Conference on Complex Networks and Their Applications*, pages 610–622. Springer, 2020a.
- Emmanouil Krasanakis, Emmanouil Schinas, Symeon Papadopoulos, Yiannis Kompatsiaris, and Andreas Symeonidis. Boosted seed oversampling for local community ranking. *Information Processing & Management*, 57(2):102053, 2020b.
- Brian Kulis, Sugato Basu, Inderjit Dhillon, and Raymond Mooney. Semi-supervised graph clustering: a kernel approach. *Machine learning*, 74(1):1–22, 2009.
- Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3): 033015, 2009.

- Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.
- Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.
- Jure Leskovec, Lada A Adamic, and Bernardo A Huberman. The dynamics of viral marketing. ACM Transactions on the Web (TWEB), 1(1):5, 2007.
- Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.
- Jure Leskovec, Kevin J Lang, and Michael Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World wide web*, pages 631–640. ACM, 2010.
- Hanbaek Lyu. Convergence of block coordinate descent with diminishing radius for non-convex optimization. arXiv preprint arXiv:2012.03503, 2020.
- Vincenzo Musco. Dataset for Generative Model of Software Dependency Graphs. 4 2016. doi: 10.5281/zenodo.49665. URL https://zenodo.figshare.com/articles/dataset/Dataset_for_Generative_Model_of_Software_Dependency_Graphs/6325993.
- Galileo Namata, Ben London, Lise Getoor, Bert Huang, and UMD EDU. Query-driven active surveying for collective classification. In 10th International Workshop on Mining and Learning with Graphs, volume 8, 2012.
- Eirini Ntoutsi, Pavlos Fafalios, Ujwal Gadiraju, Vasileios Iosifidis, Wolfgang Nejdl, Maria-Esther Vidal, Salvatore Ruggieri, Franco Turini, Symeon Papadopoulos, Emmanouil Krasanakis, et al. Bias in data-driven artificial intelligence systems—an introductory survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10(3): e1356, 2020.
- Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.
- Symeon Papadopoulos, Yiannis Kompatsiaris, Athena Vakali, and Ploutarchos Spyridonos. Community detection in social media. *Data Mining and Knowledge Discovery*, 24(3): 515–554, 2012.
- Leto Peel, Daniel B Larremore, and Aaron Clauset. The ground truth about metadata and community detection in networks. *Science advances*, 3(5):e1602548, 2017.
- Tahleen A Rahman, Bartlomiej Surma, Michael Backes, and Yang Zhang. Fairwalk: Towards fair graph embedding. In *IJCAI*, pages 3289–3295, 2019.
- Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In AAAI, 2015. URL http://networkrepository.com.

- Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs: Graph filters. In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pages 6163–6166. IEEE, 2013.
- Satu Elisa Schaeffer. Graph clustering. Computer science review, 1(1):27-64, 2007.
- Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender* systems handbook, pages 257–297. Springer, 2011.
- Daniel Spielman. Spectral graph theory. In *Combinatorial scientific computing*, number 18. Citeseer, 2012.
- Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90, 2004.
- Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998. ACM, 2008.
- Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. In *Sixth international conference on data mining (ICDM'06)*, pages 613–622. IEEE, 2006.
- Sotiris Tsioutsiouliklis, Evaggelia Pitoura, Panayiotis Tsaparas, Ilias Kleftakis, and Nikos Mamoulis. Fairness-aware link analysis. arXiv preprint arXiv:2005.14431, 2020.
- Huibing Wang, Lin Feng, Xiangzhu Meng, Zhaofeng Chen, Laihang Yu, and Hongwei Zhang. Multi-view metric learning based on kl-divergence for similarity measurement. *Neurocomputing*, 238:269–276, 2017.
- Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Wei Chen, and Tie-Yan Liu. A theoretical analysis of ndcg ranking measures. In *Proceedings of the 26th annual conference on learning theory (COLT 2013)*, volume 8, page 6, 2013.
- Joyce Jiyoung Whang, David F Gleich, and Inderjit S Dhillon. Overlapping community detection using neighborhood-inflated seed expansion. *IEEE Transactions on Knowledge and Data Engineering*, 28(5):1272–1284, 2016.
- Xiao-Ming Wu, Zhenguo Li, Anthony M So, John Wright, and Shih-Fu Chang. Learning with partially absorbing random walks. In *Advances in neural information processing systems*, pages 3077–3085, 2012.
- Jierui Xie, Stephen Kelley, and Boleslaw K Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *Acm computing surveys (csur)*, 45(4):43, 2013.
- Ke Yang and Julia Stoyanovich. Measuring fairness in ranked outputs. In *Proceedings of the* 29th International Conference on Scientific and Statistical Database Management, pages 1–6, 2017.

- Yuichi Yoshida. Cheeger inequalities for submodular transformations. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2582–2601. SIAM, 2019.
- Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P Gummadi. Fairness constraints: A flexible approach for fair classification. *J. Mach. Learn. Res.*, 20 (75):1–42, 2019.
- Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. Fa* ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1569–1578, 2017.