Fair Disaster Containment via Graph-Cut Problems

Michael Dinitz
Johns Hopkins University

Aravind Srinivasan
University of Maryland

Leonidas Tsepenekas
University of Maryland

Anil Vullikanti
University of Virginia

Abstract

Graph cut problems are fundamental in combinatorial Optimization, and are a central object of study in both theory and practice. Further, the study of *fairness* in Algorithmic Design and Machine Learning has recently received significant attention, with many different notions proposed and analyzed for a variety of contexts. In this paper we initiate the study of fairness for graph cut problems by giving the first fair definitions for them, and subsequently we demonstrate appropriate algorithmic techniques that yield a rigorous theoretical analysis. Specifically, we incorporate two different notions of fairness, namely *demographic* and *probabilistic individual* fairness, in a particular cut problem that models disaster containment scenarios. Our results include a variety of approximation algorithms with provable theoretical guarantees.

1 INTRODUCTION

Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E, where $n = |V|$ and every $e \in E$ has a cost $w_e \in \mathbb{R}_{\geq 0}$. In addition, we are given a designated "source" vertex $s \in V$. We are concerned with attempting to mitigate some sort of "disaster" that begins at s and infectiously spreads through the network via the edges. This means that vertices $v \in V$ that are connected to s (i.e., there exists an undirected $s-v$ path in G) are at some sort of risk or disadvantage.

A natural approach to mitigate the aforementioned spread is to remove edges from G, in an attempt to disconnect as many vertices of the graph from s as possible. Specifically, if we remove a cut-set or simply cut $F \subseteq E$ from the graph, we denote by $\text{prot}(V, E \setminus F, s)$ the set of vertices in V that are no longer connected to s in $G_F = (V, E \setminus F)$, and hence are *protected* from the infectious process. At a high-level, the edge removal strategy contains the disastrous event within the set $V \setminus \text{prot}(V, E \setminus F, s)$. Observe now that there is a clear trade-off between the cost $w(F)$[1] of the cut F and $|\text{prot}(V, E \setminus F, s)|$, i.e., the more edges we remove the more vertices we may be able to save.

The aforementioned trade-off naturally leads to the following optimization problem, which we call *Size Bounded Minimum Capacity Cut* or SB-MinCC for short. Given a graph G with designated source vertex s and a integer target value $T > 0$, we want to compute a cut $F \subseteq E$ with the minimum possible cost $w(F)$, such that at least T vertices of V are saved in $G_F = (V, E \setminus F)$, i.e. $|\text{prot}(V, E \setminus F, s)| \geq T$. This problem is NP-hard as shown in (Hayrapetyan et al., 2005). The work of Svitkina and Tardos (2004) gave a $O(\log^2 n)$-approximation algorithm for SB-MinCC, while Hayrapetyan et al. (2005); Eubank et al. (2004) gave constant factor bicriteria algorithms for it, i.e., algorithms that provide solutions that come within a constant factor of the optimal cut cost, but at the same time might not save at least T vertices.

Inspired by the recent interest revolving around algorithmic fairness, our goal in this paper is to incorporate such ideas in SB-MinCC, and initiate the discussion of fairness requirements for cuts in graphs. To the best of our knowledge, our work here is the first to combine fairness with this family of problems.

The first notion of fairness that we consider is the widely used *Demographic Fairness* one. The high-level idea behind this definition is that the set of elements that require "service" consists of various subsets—say demographic groups—and the solution should equally and fairly treat and represent each of these groups. In our case, if the vertices of the graph belong to different groups, we would like our solution to fairly separate vertices of each of them from the designated node s. *In this way, we will avoid outcomes that completely ignore certain groups for the sake of minimizing the objective*

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

[1] For a vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ and a subset $X \subseteq \{1, 2, \ldots, k\}$, we use $\alpha(X)$ to denote $\sum_{i \in X} \alpha_i$

function. Hence, we define the following problem.

DemFairCut: In addition to a graph $G = (V, E)$ with weights $\{w_e\}_{e \in E}$ and the source $s \in V$, for some integer $\gamma \geq 1$ we are given sets $V_1, V_2, \ldots, V_\gamma$ and values $f_1, f_2, \ldots, f_\gamma$, such that $\forall h \in [\gamma]$[2] we have $V_h \subseteq V$ and $f_h \in (0, 1]$. Note that each $v \in V$ may actually belong to multiple sets V_h. Letting $n_h = |V_h|$, the goal is to find a cut $F \subseteq E$ with the minimum possible $w(F)$, subject to the constraint that $|V_h \cap \text{prot}(V, E \setminus F, s)| \geq f_h \cdot n_h$ for all $h \in [\gamma]$. *In words, if each V_h is interpreted as a demographic, we want the minimum cost cut under the condition that at least an f_h fraction of the points in V_h are disconnected from s (for all h).*

Instantiating this definition with different values of f_h allows us to model a variety of fairness scenarios. For example, setting $f_h = 1/2$ would let us guarantee a solution that protects at least half the vertices of each V_h. Alternatively, we can set f_h to be a decreasing function of n_h/n, and thus yield a solution that focuses more on protecting smaller demographics. Moreover, notice that SB-MinCC is a special case of DemFairCut, where $\gamma = 1$ (we only have one demographic group) and $f_1 = \frac{T}{n}$. Hence, DemFairCut is NP-hard, since SB-MinCC is already known to be NP-hard.

The second notion of fairness we consider is called *Probabilistic Individual Fairness*, and was first introduced in the context of robust clustering (Harris et al., 2019; Anegg et al., 2020). According to it, the final solution should not simply be just one solution, but rather a distribution $\mathcal{D}$ over solutions. Then, considering each input element individually, the probability that it will get "good service" in a randomly drawn solution from this distribution, should be at most some given (fairness related) parameter. Obviously, sampling from this constructed distribution $\mathcal{D}$ must be achievable in polynomial time, and we call such distributions *efficiently-sampleable*. Under this notion of fairness, *we avoid outcomes that deterministically prevent satisfactory outcomes for certain individuals.*

Incorporating the above concept of fairness in SB-MinCC, implies that besides the global guarantee of saving at least T vertices, we also need to provide a stochastic guarantee for each individual vertex, ensuring it that in the final solution it will be disconnected from s with a certain probability. For instance, ensure that each vertex gets disconnected with probability at least $1/2$, and hence no specific vertex enjoys preferential treatment. The formal definition follows.

IndFairCut: In addition to a graph $G = (V, E)$ with weights $\{w_e\}_{e \in E}$, a target $T \in \mathbb{N}_{\geq 0}$ and source $s \in V$, for each $v \in V \setminus \{s\}$ we are also given a value $p_v \in [0, 1]$. The goal is to find an efficiently-sampleable distribution $\mathcal{D}$ over the cuts $\mathcal{F}(B) = \{F \subseteq E : w(F) \leq B \wedge |\text{prot}(V, E \setminus F, s)| \geq T\}$, such that $\Pr_{F \sim \mathcal{D}}[v \in \text{prot}(V, E \setminus F, s)] \geq p_v$ for each $v \in V \setminus \{s\}$, and B is the minimum possible.

Further, SB-MinCC is also a special case of IndFairCut, since we can always set $p_v = 0$ for all $v \in V \setminus \{s\}$ and make the stochastic constraints void. Hence, IndFairCut is also NP-hard.

Observation 1.1. In both problems, we can assume that the disastrous event simultaneously starts from a set of vertices S, instead of just a single designated vertex. This assumption is without loss of generality, since S can be merged into a single vertex s (by retaining all edges between S and $V \setminus S$), thus giving an equivalent formulation that matches ours.

1.1 Contribution and Outline

Our main contribution lies in introducing the first fair variants of graph-cut problems, together with approximation algorithms with provable guarantees for them.

In Section 2 we present a technique that is required in our approach for solving DemFairCut and IndFairCut. The key insight is that we can reduce these problems on general graphs to the same problems on trees, by using a tree embedding result of Räcke (2008).

In Section 3 we address demographic fairness. At first, we provide an $O(\log n)$-approximation algorithm for DemFairCut based on dynamic programming. The latter algorithm runs in polynomial time only when the number of groups γ is a constant. When γ is not a constant and can be any arbitrary value, we develop a different algorithm based on a linear programming relaxation together with a dependent randomized rounding technique. This result yields an $O\left(\frac{\log n \log \gamma}{\epsilon^2 \cdot \min_h f_h}\right)$-approximation for any $\epsilon > 0$. However, we mention that the covering guarantee it provides to each demographic V_h is only that at least $(1-\epsilon) f_h n_h$ vertices of it will be saved. Regarding the dependence on $\min_h f_h$, we believe that in realistic fairness related applications the covering fractions f_h should be relatively big, i.e., some constant $f_h = \Omega(1)$, since we care about protecting the vertices in the best way possible. Hence, the approximation ratio of our algorithm can be thought of as $O\left(\frac{\log n \log \gamma}{\epsilon^2}\right)$. Finally, we show that even on tree instances, DemFairCut with arbitrary γ is actually quite hard: it cannot be approximated better than $\Omega(\log \gamma)$. We do this by an approximation factor preserving reduction from Set Cover.

In Section 4 we provide an $O(\log n)$-approximation algorithm for IndFairCut. The high-level approach of

[2] We use $[k]$ to denote $\{1, \ldots, k\}$ for some integer $k \geq 1$

this result relies on the round-or-cut technique developed by Anegg et al. (2020), which we tailor in a way that suits the specific needs of our problem.

Finally, notice that since SB-MinCC is a special case of DemFairCut (with $\gamma = 1$), and also a special case of IndFairCut (when $p_v = 0$ for all v), our dynamic programming algorithm from Section 3 and the algorithm of Section 4, both provide a $O(\log n)$-approximation for SB-MinCC. This is an improvement over the best previously known $O(\log^2 n)$-approximation of Svitkina and Tardos (2004).

All missing proofs are placed in Appendix A.

1.2 Motivating Examples

Regarding demographic fairness, consider the following potential application. The vertices of the graph V would correspond to geographic areas across the globe, and an edge $(u, v) \in E$ would denote whether or not there is underlying infrastructure, e.g., highways or airplane routes, that can transport people between areas u and v. The disastrous event in this scenario is the spread of a disease in a global health crisis. If an area $u \in V$ is "infected", then it is natural to assume that neighboring areas (i.e., areas $v \in V$ with $(u, v) \in E$) can also get infected if we allow people to travel between u and v. A central planner will now naturally try to break a set of connections $F \subseteq E$ from the infrastructure graph, such that the total cost $w(F)$ of these actions will be as small as possible, while some guarantee on the number of protected areas $|\operatorname{prot}(V, E \setminus F, s)|$ is also satisfied. The value $w(F)$ can be interpreted as the economic cost of the proposed strategy F, e.g., the lost revenue of airline companies resulting from cancelling flights.

In terms of fairness, we can think of the areas V as coming from γ different countries, with V_h being the areas associated with country $h \in [\gamma]$. Then, a fair solution would not tolerate a discrepancy in how many areas are protected across different countries. For example, a fair approach would be to ensure that each country has at least half of its areas protected, since the less "infected" areas each country has, the more easily it can keep its local crisis under control.

As far as individual fairness is concerned, consider a computer network facing the spread of a computer virus. In this scenario, we want to minimize the cost of the connections removed, such that the infectious process is kept under control and thus a certain number of users T does not get infected. However, each user of the network would arguably prefer to be in the set of protected vertices. Our notion of individual fairness as studied in IndFairCut, will ensure exactly that in a stochastic sense, by using appropriate values p_v.

1.3 Related Work

The unfair variant of our problems, i.e., SB-MinCC, was studied in (Hayrapetyan et al., 2005; Svitkina and Tardos, 2004; Eubank et al., 2004). These papers also considered additional versions of SB-MinCC, where the goal was to maximize $|\operatorname{prot}(V, E \setminus F, s)|$ (equivalently minimize $|V \setminus \operatorname{prot}(V, E \setminus F, s)|$) subject to an upper bound constraint on $w(F)$.

The study of fairness in algorithmic design and machine learning has recently received significant attention. This is mainly due to the realization that the output of standard optimization algorithms can very well lead to solutions that are highly unfair and hurtful for the individuals or the groups involved. Examples of this include racial bias in Airbnb rentals (Badger, 2016), gender bias in Google's Ad Settings (Datta et al., 2015) and discrimination in housing ads in Facebook (Benner et al., 2019). There are two reasons why such unfortunate events occur. First, the training datasets may include implicit biases, and hence when algorithms are trained on them, they learn to perpetuate the underlying biases. Second, in many situations, even if the data is completely unbiased, merely optimizing an objective function does not suffice if fairness considerations are at play. In such cases, we must explicitly incorporate fairness constraints in our algorithm design process. Our work here tries to accomplish the latter.

Although algorithmic fairness has not yet been addressed in cut problems, there are other areas such as classification and clustering were examples of fair algorithms are abundant. For example, Chierichetti et al. (2017); Bercea et al. (2019); Bera et al. (2019); Huang et al. (2019); Backurs et al. (2019); Ahmadian et al. (2019) consider notions of demographic fairness in clustering, while Brubach et al. (2020); Anegg et al. (2020); Harris et al. (2019) focus on notions of individually-fair clustering. In the context of fair classification, one of the most seminal works with significant implications in other fields as well, is the paper of Dwork et al. (2012). This work studies individual fairness and its interplay with a notion of demographic fairness, namely statistical parity. Some excellent surveys on the topic of algorithmic fairness are (Barocas et al., 2019; Mehrabi et al., 2021).

2 REDUCTION TO TREE INSTANCES

In this section we show how both DemFairCut and IndFairCut can be effectively reduced to solving an appropriate problem on a tree instance.

Definition 2.1. We call an algorithm for DemFair-

CUT (ρ, α)-*bicriteria*, if for any given problem instance $\mathcal{I} = \{V, E, s, w, V_1, \ldots, V_\gamma, \vec{f}\}$ with optimal value $OPT_{\mathcal{I}}$, it returns a solution F such that **1)** $w(F) \leq \rho OPT_{\mathcal{I}}$, and **2)** $|\operatorname{prot}(V, E \setminus F, s) \cap V_h| \geq \alpha f_h n_h,\ \forall h \in [\gamma]$.

Lemma 2.2. *If we have a (ρ, α)-bicriteria algorithm for* DEMFAIRCUT *in trees, we can get a $(\rho \cdot O(\log n), \alpha)$-bicriteria algorithm for* DEMFAIRCUT *in general graphs.*

Our approach for tackling INDFAIRCUT uses as a black-box an algorithm for a new problem, which we call AUXCUT and we formally define below. In order to get an algorithm for general instances of AUXCUT, we again use a reduction to trees.

AUXCUT: We are given an undirected graph $G = (V, E)$, a designated vertex $s \in V$, a budget $B > 0$, and a target value $T \in \mathbb{N}_{\geq 0}$. In addition, each $e \in E$ has a weight $w_e \geq 0$, and each vertex $v \in V \setminus \{s\}$ has a value $a_v \geq 0$. The goal is to find a cut F with $w(F) \leq B$ and $|\operatorname{prot}(V, E \setminus F, s)| \geq T$, that maximizes $a(\operatorname{prot}(V, E \setminus F, s))$.

Definition 2.3. We say that an algorithm is $(1, 1, \rho)$-bicriteria for AUXCUT, if for any given instance $\mathcal{I} = (V, E, B, T, w, s, a)$ of the problem with optimal value $OPT_{\mathcal{I}}$, it returns a set of edges F, such that **1)** $w(F) \leq \rho B$, **2)** $|\operatorname{prot}(V, E \setminus F, s)| \geq T$ and **3)** $a(\operatorname{prot}(V, E \setminus F, s)) \geq OPT_{\mathcal{I}}$.

Lemma 2.4. *Given a $(1, 1, \rho)$-bicriteria algorithm for* AUXCUT *in tree instances, we get a $(1, 1, \rho O(\log n))$-bicriteria algorithm for* AUXCUT *in general graphs.*

3 ADDRESSING DEMOGRAPHIC FAIRNESS

In this section we tackle DEMFAIRCUT and present two algorithms for it. The first works when is γ a constant, and is an $O(\log n)$-approximation. The second addresses the case of an arbitrary γ, and for any $\epsilon > 0$ it is an $\left(O\left(\frac{\log n \log \gamma}{\epsilon^2 \cdot \min_h f_h}\right), 1 - \epsilon\right)$-bicriteria one.

3.1 Solving DEMFAIRCUT for $\gamma = O(1)$

Given Lemma 2.2, we focus on only solving the problem in tree instances. Specifically, we show that when $\gamma = O(1)$ the problem in trees can be solved optimally via dynamic programming. Without loss of generality, we can also assume that the given tree is rooted at s and it is binary (see Lemma 15.18 from (Williamson and Shmoys, 2011)). Before we describe our approach we need some additional notation. For a vertex v, let $\phi_h(v) = 1$ if $v \in V_h$ and 0 otherwise.

Our dynamic programming algorithm is based on a table M, where $M[v, k_1, k_2, \ldots, k_\gamma]$ represents the minimum cost of a cut in the subtree rooted at v, so that there are exactly k_h nodes from V_h that are connected to v. Let v_r be the right child of v, and let v_ℓ be the left child of v. Observe that the optimal solution either cuts neither of the edges from v to its children, just the left edge, just the right edge, or both of the edges. So, we set $M[v, k_1, k_2, \ldots, k_\gamma]$ to the minimum of the following:

1. $\min\left\{M[v_\ell, k_1^\ell, k_2^\ell, \ldots, k_\gamma^\ell] + M[u_r, k_1^r, k_2^r, \ldots, k_\gamma^r] : k_h^\ell + k_h^r + \phi_h(v) = k_h\ \forall h \in [\gamma]\right\}$
2. $\min\left\{w_{(v,v_\ell)} + M[v_r, k_1', k_2', \ldots, k_\gamma'] : k_h' + \phi_h(v) = k_h\ \forall h \in [\gamma]\right\}$
3. $\min\left\{w_{(v,v_r)} + M[v_\ell, k_1', k_2', \ldots, k_\gamma'] : k_h' + \phi_h(v) = k_h\ \forall h \in [\gamma]\right\}$
4. $w_{(v,v_r)} + w_{(v,v_\ell)}$ if $k_h = \phi_h(v)$ for all $h \in [\gamma]$, $+\infty$ otherwise.

The first case above corresponds to cutting neither of the edges (v, v_r), (v, v_ℓ), the second to cutting only (v, v_ℓ), the third to cutting only (v, v_r), and the fourth to cutting both.

In order to successfully fill in M, we begin by initializing $M[v, \phi_1(v), \phi_2(v), \ldots, \phi_\gamma(v)] = 0$ for all leaves v of the tree, and set all other table entries to $+\infty$. Then we proceed by filling the table bottom-up. There are at most $O(n^{\gamma+1})$ table entries, and to compute each one we need to access at most $2n^\gamma$ other ones. Thus, the total runtime is $O(n^{2\gamma+1})$. Finally, in order to find the optimal cut, we look for the minimum entry $M[s, k_1, \ldots, k_\gamma]$, with $k_h \leq (1 - f_h)n_h$ for all $h \in [\gamma]$.

Theorem 3.1. *When γ is a constant, we have an optimal dynamic programming algorithm for* DEMFAIRCUT *in trees, running in time $O(n^{2\gamma+1})$.*

Combining Theorem 3.1 with Lemma 2.2, we see that our approach achieves the following.

Theorem 3.2. *When $\gamma = O(1)$, we give a $O(\log n)$-approximation algorithm for* DEMFAIRCUT.

3.2 Solving DEMFAIRCUT for an Arbitrary γ

Given Lemma 2.2, we again focus on instances $\mathcal{I} = \{V, E, s, w, V_1, \ldots, V_\gamma, \vec{f}\}$, where the underlying graph $T = (V, E)$ is a tree. Moreover, we can assume without loss of generality that the tree is rooted at s. Before we proceed with the description of our algorithm, we need some more notation. For every $v \in V$ let $P(s, v) \subseteq E$

be the unique path from s to v in the tree, and $\ell(v) = |P(s,v)|$. In addition, for every $e = (u,v) \in E$ let $P_e = P(s, r(e))$, with $r(e) = \arg\min_{z \in \{u,v\}} \ell(z)$. In words, P_e contains the edges of the path that starts from s and finishes just before reaching e. The following linear program (LP) is then a valid relaxation of our problem.

$$\min \sum_{e \in E} w_e \cdot x_e \tag{1}$$

$$y_v = \sum_{e \in P(s,v)} x_e \qquad \forall v \in V \tag{2}$$

$$\sum_{v \in V_h} y_v \geq f_h \cdot n_h \qquad \forall h \in [\gamma] \tag{3}$$

$$0 \leq y_v, x_e \leq 1 \qquad \forall v \in V, e \in E \tag{4}$$

In the integral version of LP (1)-(4), $x_e = 1$ iff edge e is included in the cut. Now notice that because the underlying graph is a tree and the edge weights are non-negative, for any $v \in V$ the optimal solution would not choose more than one edge from $P(s,v)$. Therefore, by constraints (2) and (4) we see that $y_v = 1$ iff v is separated from s in the optimal outcome. Consequently, constraint (3) naturally captures the demographic covering requirements.

Our approach begins by solving LP (1)-(4) in order to get a fractional solution x, y. We then apply the following dependent randomized rounding scheme. We consider the edges of the tree in non-decreasing order of $|P_e|$, and for an edge e for which no other edge in P_e is already chosen for the cut, we remove it with probability $x_e/(1 - x(P_e))$ if $x(P_e) < 1$. The latter action is well-defined because for every $e' \in P_e$ we have $|P_{e'}| < |P_e|$, and hence e' is considered before e in the given ordering. Further, if an edge e is chosen to be placed in the cut, then all $v \in V$ with $e \in P(s,v)$ are now disconnected from s. In addition, due to the dependent nature of this process, no path $P(s,v)$ will have more than one edge of it in the solution.

Algorithm 1 demonstrates all necessary details of the rounding, with X_e being an indicator random variable denoting whether or not e is included in the solution, and Y_v an indicator random variable that is 1 iff v is disconnected from s in the final outcome.

Lemma 3.3. *When we randomly decide to include $e \in E$ in the cut, we do so with a valid probability.*

Proof. Let $e = (u,v)$, and without loss of generality assume $l(u) < l(v)$. This means that $P_e = P(s,u)$ and $P(s,v) = P(s,u) \cup \{e\}$. In addition, to consider a randomized decision for e we should also have $x(P_e) < 1$. Using constraints (2) and (4) for v we therefore get:

$$x_e + \sum_{e' \in P_e} x_{e'} \leq 1 \implies \frac{x_e}{1 - x(P_e)} \leq 1 \qquad \square$$

Algorithm 1: Randomized Rounding

For all $e \in E$ and $v \in V$, set $X_e \leftarrow 0$ and $Y_v \leftarrow 0$;
for *all $e \in E$ in non-decreasing order of $|P_e|$* **do**
  if $x(P_e) < 1$ *and* $X_{e'} = 0$ *for all* $e' \in P_e$ **then**
    Set $X_e \leftarrow 1$ with probability $x_e/(1 - x(P_e))$;
    if $X_e = 1$ **then**
      Set $Y_v \leftarrow 1$ for all $\{v \in V : e \in P(s,v)\}$;
    end
  end
end

Lemma 3.4. *For every $e \in E$ and $v \in V$, we have $\Pr[X_e = 1] = x_e$ and $\Pr[Y_v = 1] = y_v$.*

Proof. Let us begin with an $e \in E$ for which we never made a random decision because $x(P_e) \geq 1$, and hence $X_e = 0$. If $e = (u,v)$ with $l(u) < l(v)$, then $P_e = P(s,u)$ and $P(s,v) = P(s,u) \cup \{e\}$. Because of constraints (2) and (4) for u we first get $x(P_e) = 1$. Therefore, constraints (2) and (4) applied this time for v yield $x_e = 0$, which indeed gives $\Pr[X_e = 1] = x_e$.

Now consider edge e with $x(P_e) < 1$. Because for each $e' \in P_e$ we have $P_{e'} \subset P_e$, we also get $x(P_{e'}) < 1$. The latter means that for all other edges in P_e a random decision potentially takes place. Further, analysis of the algorithm's actions shows that $\Pr[X_e = 1]$ is

$$\Pr[X_e = 1 \mid X_{e'} = 0 \ \forall e' \in P_e] \cdot \Pr[X_{e'} = 0 \ \forall e' \in P_e]$$
$$= \frac{x_e}{1 - \sum_{e' \in P_e} x_{e'}} \prod_{e' \in P_e} \left(1 - \frac{x_{e'}}{1 - \sum_{e'' \in P_{e'}} x_{e''}}\right) \tag{5}$$

Let $e_1, \ldots, e_m$ the edges of P_e in increasing order of $|P_{e_j}|$. Then because $P_{e_j} = \{e_{j'} \mid j' < j\}$, (5) can be rewritten as a telescopic product of fractions:

$$\frac{x_e}{1 - \sum_{j=1}^{m} x_{e_j}} \prod_{j=1}^{m} \left(1 - \frac{x_{e_j}}{1 - \sum_{i=1}^{j-1} x_{e_i}}\right) = x_e$$

As for a vertex $v \in V$, we have $\Pr[Y_v = 1] = \Pr[\exists e \in P(s,v) : X_e = 1]$ because there is a unique path from s to it. Moreover, since our rounding will never put more than one edges of $P(s,v)$ in the cut, for all $S \subseteq P(s,v)$ with $|S| \geq 2$ we get $\Pr[X_e = 1, \forall e \in S] = 0$. Hence, by the inclusion-exclusion principle $\Pr[\exists e \in P(s,v) : X_e = 1] = \sum_{e \in P(s,v)} \Pr[X_e = 1] = \sum_{e \in P(s,v)} x_e = y_v$, where the last equality follows from constraint (2). $\square$

We will now analyze the satisfaction of the coverage constraints for the different demographics. If S_h is the number of vertices from V_h that are not connected to s in the solution, we see that $S_h = \sum_{v \in V_h} Y_v$. Using

Lemma 3.4 and constraint (3) gives $\mathbb{E}[S_h] \geq f_h n_h$. We thus need to calculate how much can S_h deviate from $\mathbb{E}[S_h]$. For that we will need the following two lemmas.

Lemma 3.5. *(Janson, 1998) Let $Z_1, \ldots, Z_m$ be Bernoulli random variables, where $\Pr[Z_i = 1] = z_i$ for every $i \in [m]$. Let Γ be the dependency graph on the Z_i. For $i \neq j$, Z_i and Z_j are dependent if there exists an edge between them in Γ, and we denote that as $i \sim j$. Let also $Z = \sum_{i=1}^{m} Z_i$, $\mu = \mathbb{E}[Z]$, $\Delta = \sum_{\{i,j\}: i \sim j} \Pr[Z_i = Z_j = 1]$, $\delta_i = \sum_{j \sim i} z_j$ and $\delta = \max_i \delta_i$. Then for any $\epsilon \in [0,1]$*

$$\Pr[Z \leq (1-\epsilon)\mu] \leq \exp\Big(- \min\Big(\frac{\epsilon^2 \cdot \mu^2}{8\Delta + 2\mu}, \frac{\epsilon \cdot \mu}{6\delta} \Big) \Big)$$

Lemma 3.6. *For every positive integer m and some sequence of non-negative numbers $a_1, a_2, \ldots$ we have $\sum_{i=1}^{m-1} (m-i) a_i \leq m \sum_{i=1}^{m} a_i$.*

Proof. We prove the statement via induction on m. For $m = 1$ it is trivial. Suppose that the lemma holds up to some $m = k$. We then prove it for $m = k+1$:

$$\begin{aligned}
\sum_{i=1}^{k+1-1} (k+1-i) a_i &= \sum_{i=1}^{k} \Big((k-i) a_i + a_i \Big) \\
&= \sum_{i=1}^{k} (k-i) a_i + \sum_{i=1}^{k} a_i \\
&= \sum_{i=1}^{k-1} (k-i) a_i + \sum_{i=1}^{k} a_i \\
&\leq k \sum_{i=1}^{k} a_i + \sum_{i=1}^{k} a_i \\
&\leq (k+1) \sum_{i=1}^{k} a_i \leq (k+1) \sum_{i=1}^{k+1} a_i
\end{aligned}$$

The first inequality uses the inductive hypothesis, while the last one the fact that $a_{k+1} \geq 0$. □

Lemma 3.7. *For all $h \in [\gamma]$ and any $\epsilon \in [0,1]$, we have $\Pr[S_h \leq (1-\epsilon)\mathbb{E}[S_h]] \leq e^{\frac{-\epsilon^2 \cdot f_h}{10}}$.*

Proof. Due to Lemma 3.4, the random variables Y_v for $v \in V_h$ are Bernoulli with $\Pr[Y_v = 1] = y_v$. Because of the tree structure they are also to some extent dependent. Our goal here is to apply Lemma 3.5 for S_h, and towards that end we need to upper bound the dependency factors δ, Δ. Since we do not know exactly the underlying dependency graph Γ, we assume that all pairs $Y_v, Y_{v'}$ are dependent. We begin by upper-bounding the parameter Δ of Lemma 3.5.

$$\begin{aligned}
\Delta &\leq \sum_{\{v,v'\} \in V_h} \Pr[Y_v = Y_{v'} = 1] \\
&\leq \sum_{\{v,v'\} \in V_h} \min(\Pr[Y_v = 1], \Pr[Y_{v'} = 1]) \\
&= \sum_{\{v,v'\} \in V_h} \min(y_v, y_{v'})
\end{aligned}$$

Now let $a_1, a_2, ..., a_{n_h}$ be the values y_v for all $v \in V_h$ in non-decreasing order. Then we have:

$$\begin{aligned}
\sum_{\{v,v'\} \in V_h} \min(y_v, y_{v'}) &= \sum_{i=1}^{n_h - 1} (n_h - i) a_i \\
&\leq n_h \sum_{i=1}^{n_h} a_i = n_h \cdot \mathbb{E}[S_h]
\end{aligned}$$

To get the first inequality we used Lemma 3.6. Therefore, we get $\Delta \leq n_h \cdot \mathbb{E}[S_h]$. Moreover, a straightforward upper bound for each δ_v is $\delta_v \leq \sum_{u \in V_h} y_u = \mathbb{E}[S_h]$. Thus, $\delta \leq \mathbb{E}[S_h]$. Finally, we also need bounds for the following two quantities, where $\mu = \mathbb{E}[S_h]$:

$$\begin{aligned}
\frac{\epsilon^2 \cdot \mu^2}{8\Delta + 2\mu} &\geq \frac{\epsilon^2 \cdot \mu^2}{8\mu \cdot n_h + 2\mu} = \frac{\epsilon^2 \cdot \mu}{8 n_h + 2} \\
&\geq \frac{\epsilon^2 \cdot n_h \cdot f_h}{8 n_h + 2} \geq \frac{\epsilon^2 \cdot f_h}{10} \\
\frac{\epsilon \cdot \mu}{6\delta} &\geq \frac{\epsilon \cdot \mu}{6\mu} = \frac{\epsilon}{6}
\end{aligned}$$

Since $\frac{\epsilon}{6} \geq \frac{\epsilon^2 \cdot f_h}{10}$ for any $\epsilon, f_h \in [0,1]$, Lemma 3.5 immediately gives the desired bound. □

To conclude, for some $\beta \geq 2$ we repeat Algorithm 1 independently $N = \frac{10 \log \gamma^\beta}{\epsilon^2 \cdot \min_h f_h}$ times, and in each run t of it (with $t \in [N]$) we compute a set of edges F_t that are chosen to be removed. Our final solution is set to be $F = \bigcup_t F_t$. Then we have the following.

Theorem 3.8. *For* DemFairCut *in trees and any $\epsilon \in (0,1)$, we give an $\Big(O\Big(\frac{\log \gamma}{\epsilon^2 \min_h f_h}\Big), 1-\epsilon\Big)$-bicriteria algorithm that runs in expected polynomial time.*

Proof. Focus on a specific demographic h, and let S_h^t the random variable denoting the number of nodes of V_h separated from s in $(V, E \setminus F_t)$. By Lemma 3.7 and the independent nature of the runs we get $\Pr\left[S_h^t \leq (1-\epsilon)\mathbb{E}[S_h^t],\ \forall t\right] \leq e^{\frac{-\epsilon^2 \cdot N \cdot f_h}{10}} \leq \frac{1}{\gamma^\beta}$. Thus, because $\mathbb{E}[S_h^t] \geq f_h n_h$ for all t, we have $\Pr\Big[\big|V_h \cap \text{prot}(V, E \setminus F, s)\big| \geq (1-\epsilon) f_h n_h\Big] \geq \Pr\left[\exists t : S_h^t > (1-\epsilon)\mathbb{E}[S_h^t]\right] \geq 1 - \frac{1}{\gamma^\beta}$. A union bound over all demographics would finally give $\Pr\Big[\big|V_h \cap \text{prot}(V, E \setminus F, s)\big| \geq (1-\epsilon) f_h n_h,\ \forall h \in [\gamma]\Big] \geq 1 - \frac{1}{\gamma^{\beta-1}}$.

By Lemma 3.4, in each run an edge e gets removed with probability x_e. Hence, with a union bound over all runs, the probability that e gets removed is at most Nx_e. Therefore, the total expected cost of our algorithm is $N\sum_{e\in E} w_e x_e$, and since LP (1)-(4) is a valid relaxation of the problem, we immediately get the desired approximation ratio on expectation. By Markov's inequality we can further prove that with probability at most $\frac{1}{c}$, we get a final cut of cost more than $cN\sum_{e\in E} w_e x_e$ for some constant $c > 1$.

Thus, with constant probability our algorithm satisfies both the ratio of $O(\frac{\log\gamma}{\epsilon^2 \min_h f_h})$, and the $1-\epsilon$ approximate satisfaction of the demographic constraints (specifically we fail to satisfy both of the above with probability at most $1/\gamma^{\beta-1} + 1/c$). Hence, repeating the whole process an expected logarithmic number of times, guarantees that we hit both targets. □

By combining Theorem 3.8 and Lemma 2.2, we see that our approach achieves the following.

Theorem 3.9. *For any given constant $\epsilon \in (0,1)$, we give an $\left(O\left(\frac{\log n \log\gamma}{\epsilon^2 \cdot \min_h f_h}\right), 1-\epsilon\right)$-bicriteria algorithm for* DemFairCut*, that runs in expected polynomial time.*

3.2.1 Hardness of DemFairCut with Arbitrary γ

Here we show that even in tree instances, DemFairCut with arbitrary γ is hard.

Set Cover: We are given a universe of elements U and a collection of m sets $\{S_1, S_2, \ldots, S_m\}$, where $S_i \subseteq U$ for every $i \in [m]$. The goal is to find $C \subseteq [m]$, such that $\bigcup_{i\in C} S_i = U$ and $|C|$ is minimized.

Theorem 3.10 (Dinur and Steurer (2014))**.** *It is NP-hard to approximate Set Cover instances of universe size n and $m \leq \text{poly}(n)$ sets within a factor better than $\ln n$.*

Theorem 3.11. *It is NP-hard to approximate* DemFairCut *with arbitrary γ on tree instances within a factor better than $\ln\gamma$.*

Proof. Suppose that we are given an instance of Set Cover. We create an instance of DemFairCut as follows. For every set S_i we create a vertex v_i. For every element $e \in U$ we create a demographic group $V_e = \{v_i \mid e \in S_i\}$. We set the covering requirement of the group V_e to be $1/|V_e|$, i.e., we want our solution to protect at least $|V_e| \cdot (1/|V_e|) = 1$ vertex from each V_e. Finally, we add the designated vertex s to the graph, and create edges (s, v_i) for every v_i. Note that the resulting graph is a tree.

Now consider the optimal Set Cover solution C^*. We claim that the set of edges $\{(s, v_i) \mid i \in C^*\}$ is a feasible solution for the constructed instance of DemFairCut. Take any demographic V_e for $e \in U$. Because C^* is a feasible Set Cover solution, it contains at least one S_j with $e \in S_j$. Therefore, we are going to include the edge (s, v_j) to our graph solution, and the vertex v_j from the group V_e is going to be protected. Finally, see that $|C^*| = |\{(s, v_i) \mid i \in C^*\}|$, and hence the cost of the optimal solution for the DemFairCut instance, say F^*, is at most $|C^*|$.

Now we argue that any solution F to the DemFairCut instance yields a feasible solution C_F for the Set Cover instance with $|F| = |C_F|$. Simply take $C_F = \{i \in [m] \mid (s, v_i) \in F\}$. It is clear that $|F| = |C_F|$. Now consider each $e \in U$. Since F is feasible for DemFairCut, at least one vertex $v_i \in V_e$ will be separated from s, and thus $(s, v_i) \in F$. Hence for that vertex v_i we have $e \in S_i$ by construction. Therefore, e is covered by C_F.

Suppose now that for some $\epsilon > 0$ we have an $(1-\epsilon)\ln\gamma$-approximation algorithm for DemFairCut on trees. Then given an instance of Set Cover, we first construct the instance of DemFairCut given by the above reduction and then run the given algorithm on that instance to get a solution F. Then, as discussed, we construct the corresponding Set Cover solution C_F, with $|F| = |C_F|$. By all the previous arguments we have $|C_F| = |F| \leq ((1-\epsilon)\ln\gamma)|F^*| \leq ((1-\epsilon)\ln|U|)|C^*|$. This contradicts Theorem 3.10. □

At a high-level, the previous theorem says that the best we can achieve for DemFairCut in trees is an approximation ratio of $\Omega(\log\gamma)$. Trivially this implies the following corollary.

Corollary 3.12. *Unless P=NP, the best approximation ratio we can achieve for general instances of* DemFairCut *with arbitrary γ is $\Omega(\log\gamma)$.*

4 ADDRESSING INDIVIDUAL FAIRNESS

The purpose of this section is to provide an algorithm for IndFairCut. To do so, we begin with a dynamic programming bicriteria algorithm for AuxCut on tree instances, which according to Lemma 2.4 implies an algorithm for AuxCut in general graphs. Subsequently, we show how the general graph algorithm can be incorporated in the round-or-cut framework of Anegg et al. (2020), and in this way we get as our final result a $O(\log n)$-approximation for IndFairCut.

At this point, we have to mention that the LP-based approach of Section 3.2 can also be applied here (by adding the extra constraint $y_v \geq p_v$ in LP (1)-(4)), yielding the same approximation ratio of $O(\log n)$.

However, such an approach would unavoidably lead to a bicriteria algorithm, since it will produce a solution that saves at least $(1-\epsilon)T$ vertices. On the other hand, the algorithm we present in what follows is a true approximation for IndFairCut.

4.1 A $(1, 1, O(\log n))$-Bicriteria for AuxCut

Suppose we have an instance $\mathcal{I} = (V, E, B, T, s, w, a)$ of AuxCut. Given Lemma 2.4, we focus on $G = (V, E)$ being a tree and present a dynamic programming algorithm for AuxCut in trees.

Without loss of generality, we can assume that the tree is rooted at s and is binary (see Lemma 15.18 from (Williamson and Shmoys, 2011)). Our algorithm tries to find a cut $F \subseteq E$ that minimizes $a(V \setminus \text{prot}(V, E \setminus F, s))$ subject to $w(F) \leq B$ and $|\text{prot}(V, E \setminus F, s)| \geq T$. Note that when we can compute a solution of optimal value to this minimization problem, minimizing $a(V \setminus \text{prot}(V, E \setminus F, s))$ is equivalent to maximizing $a(\text{prot}(V, E \setminus F, s))$. Therefore, the version of the problem we solve here is equivalent to the definition of AuxCut as given in Section 2.

Our approach relies on a table A. For every $v \in V$ let $T_v \subseteq V$ and $E_v \subseteq E$ be the vertices and the edges of the subtree that is rooted at v (with v included in T_v). Then, the entry $A[v, W, k]$ would represent the minimum possible $a(T_v \setminus \text{prot}(T_v, E_v \setminus F_v, v))$, for any cut $F_v \subseteq E_v$ with $w(F_v) = W$ and $|T_v \setminus \text{prot}(T_v, E_v \setminus F_v, v)| = k$ (see that the vertices of T_v connected to v in this cut are those in $T_v \setminus \text{prot}(T_v, E_v \setminus F_v, v)$). Let also v_r be the right child of v, and let v_ℓ be the left child of v. The optimal solution of $\mathcal{I}$ either cuts none of the edges from v to its children, just the left edge, just the right edge, or both edges. So we just have to set $A[v, W, k]$ to the minimum of the following:

1. $\min\Big\{A[v_\ell, W_\ell, k_\ell] + A[v_r, W_r, k_r] + a_v : W_\ell + W_r = W \text{ and } k_\ell + k_r + 1 = k\Big\}$
2. $A[v_r, W - w_{(v,v_\ell)}, k-1] + a_v$ if $W \geq w_{(v,v_\ell)}$ and $k > 1$, $+\infty$ otherwise.
3. $A[v_\ell, W - w_{(v,v_r)}, k-1] + a_v$ if $W \geq w_{(v,v_r)}$ and $k > 1$, $+\infty$ otherwise.
4. a_v if $w_{(v,v_\ell)} + w_{(v,v_r)} = W$ and $k = 1$, ∞ otherwise

The first case above corresponds to cutting neither of the edges (v, v_r), (v, v_ℓ), the second to cutting only (v, v_ℓ), the third to cutting only (v, v_r), and the fourth to cutting both.

To fill in A, we begin by initializing $A[v, 0, 1] = a_v$ for all leaves v of the tree, and all other entries to $+\infty$. Then we proceed by filling the table bottom-up. Assuming that the edge weights are integers, we see that A has n^2B entries, and in order to fill each of them, we need access to at most $2nB$ other entries. Hence, in total our approach requires $O(n^3B^2)$ time. Finally, in order to find the optimal cut, we look for the minimum entry $A[s, W, k]$, such that $W \leq B$ and $k \leq n - T$.

Corollary 4.1. *When the edge weights are integers and $B = \text{poly}(n)$, we can efficiently find an optimal solution of* AuxCut *in tree instances.*

To make sure the edge weights are integers and B is polynomially bounded, we use a standard discretization trick before running the dynamic program (Svitkina and Tardos, 2004). Specifically, for any $\epsilon > 0$, let $\lambda = \frac{\lceil m/\epsilon \rceil}{B}$, where $m = |E|$. Then for each edge $e \in E$ create a new weight $w'_e = \lfloor \lambda w_e \rfloor$. Also, set $B' = \lambda B = \lceil m/\epsilon \rceil$. Notice now that all new edge weights are integers and that B' is polynomial in n. Further, using these new values we create a new instance $\mathcal{I}' = (V, E, B', T, s, w', a)$ of AuxCut. It is easy to see that if there is a solution of edge-cost B for $\mathcal{I}$, then this solution has edge-cost B' in $\mathcal{I}'$. In addition, for every solution of $\mathcal{I}'$ whose edge-cost is at most B', its edge-cost in $\mathcal{I}$ is at most $(1+\epsilon)B$. Combining this with Corollary 4.1 gives the following.

Corollary 4.2. *Our approach provides a $(1, 1, 1+\epsilon)$-bicriteria algorithm for* AuxCut *in trees.*

Finally, by Corollary 4.2, Lemma 2.4 and the fact that ϵ is a constant, we get:

Theorem 4.3. *We provide a $(1, 1, O(\log n))$-bicriteria algorithm for* AuxCut.

4.2 A Round-or-Cut Solution for IndFairCut

Suppose we are given an instance $\mathcal{I} = (V, E, T, s, w, \vec{p})$ of IndFairCut with optimal value $OPT_{\mathcal{I}}$. For any value $B \geq 0$, let $\mathcal{F}(B) = \{F \subseteq E : w(F) \leq B \text{ and } |\text{prot}(V, E \setminus F, s)| \geq T\}$. In the rest of the section we demonstrate a process, which given $\mathcal{I}$ and a target value $B \geq 0$, operates as follows. It either returns an efficiently-sampleable distribution $\mathcal{D}$ over the cuts in the set $\mathcal{F}(O(\log n)B)$ such that $\Pr_{F\sim\mathcal{D}}[v \in \text{prot}(V, E \setminus F, s)] \geq p_v$ for every $v \in V \setminus \{s\}$, or returns "INFEASIBLE". If the latter happens, then it is guaranteed that $B < OPT_{\mathcal{I}}$.

Using the above process in a bisection search with step $(1+\epsilon)$ over the range $[0, w(E)]$, we can efficiently compute a value $B' \leq (1+\epsilon)OPT_{\mathcal{I}}$, such that the process will not return "INFEASIBLE" for B'. This will actually yield an efficiently-sampleable distribution over $\mathcal{F}(O(\log n)B')$ that satisfies the stochastic constraints for all vertices. Hence, we get our final result.

Theorem 4.4. *For any $\epsilon > 0$ and instance $\mathcal{I}$ with optimal value $OPT_{\mathcal{I}}$, we construct an efficiently sampleable distribution $\mathcal{D}$ over $\mathcal{F}(O(\log n)(1+\epsilon)OPT_{\mathcal{I}})$, such that $\Pr_{F\sim\mathcal{D}}[v \in \text{prot}(V, E\setminus F, s)] \geq p_v$ for all $v \in V\setminus\{s\}$. The runtime of our approach is $\text{poly}(n^{1/\epsilon})$.*

Therefore, since for our final result the aforementioned process is all that is required, we start describing its details. Notice now that for a given target value B, we are basically interested in verifying whether or not there is a feasible solution to $\mathcal{I}$ with edge-cost at most B. Hence, consider the following exponential-sized linear program, which we call PLP(B).

PLP(B)

$$\begin{aligned} \min\ & 0 \\ & \sum_{\substack{F\in\mathcal{F}(B):\\ v\in\text{prot}(V,E\setminus F,s)}} x_F \geq p_v && \forall v \in V\setminus\{s\} \\ & \sum_{F\in\mathcal{F}(B)} x_F = 1 \\ & 0 \leq x_F \leq 1 && \forall F \in \mathcal{F}(B) \end{aligned}$$

DLP(B)

$$\begin{aligned} \max & \sum_{v\in V\setminus\{s\}} p_v \cdot y_v - \mu \\ & \sum_{v\in\text{prot}(V,E\setminus F,s)} y_v \leq \mu && \forall F \in \mathcal{F}(B) \\ & 0 \leq y_v && \forall v \in V \\ & \mu \in \mathbb{R} \end{aligned}$$

If we interpret x_F as the probability of choosing the cut F from $\mathcal{F}(B)$, we see that B yields a feasible solution iff PLP(B) is feasible. This is because the first LP constraint captures the fairness requirements, and the second LP constraint the fact that the resulting solution should be a distribution over $\mathcal{F}(B)$. In addition, if PLP(B) is feasible, then there are only n values x_F with $x_F > 0$ (see Lemma 9 in (Karloff, 1991)), and hence the resulting distribution is efficiently-sampleable. Another important observation is that if PLP(B) is feasible, then clearly its optimal value is 0.

However, since solving PLP(B) is not doable in polynomial time, we focus on its dual, which we call DLP(B) and we present next to the primal LP.

Here note that DLP(B) is always feasible (e.g., set all variables to 0), and by LP duality DLP(B) has an optimal value of 0 iff PLP(B) is feasible. Further, see that DLP(B) is scale-invariant. In other words, if it has a feasible solution (y', μ') with strictly positive objective value, then DLP(B) is unbounded because $(ty', t\mu')$ will also be feasible for any $t > 0$. Consider now the following polytope that contains all feasible solutions of DLP(B) of objective value at least 1, i.e., $Q(B) = \Big\{(y,\mu) \in \mathbb{R}^{n-1}_{\geq 0} \times \mathbb{R} : \sum_{v\in V\setminus\{s\}} p_v y_v \geq \mu + 1 \wedge y(\text{prot}(V, E\setminus F, s)) \leq \mu,\ \forall F \in \mathcal{F}(B)\Big\}$. Based on the previous discussion we make the following very crucial observation.

Observation 4.5. PLP(B) is feasible iff $Q(B) = \emptyset$.

Using the algorithm of Section 4.1 we prove the following vital theorem.

Theorem 4.6. *There exists a poly-time algorithm that given a point $(y,\mu) \in \mathbb{R}^{n-1}_{\geq 0} \times \mathbb{R}$ satisfying $\sum_{v\in V\setminus\{s\}} p_v \cdot y_v \geq \mu + 1$, it either verifies that $(y,\mu) \in Q(B)$, or outputs a set $F \in \mathcal{F}(O(\log n)B)$ such that $\sum_{v\in\text{prot}(V,E\setminus F,s)} y_v > \mu$.*

Proof. We begin by constructing an instance $\mathcal{I}_{\text{aux}} = (V, E, B, T, s, w, y)$ of AuxCut, where the vertex weights correspond to the y values. Then, we run the algorithm of Section 4.1 on $\mathcal{I}_{\text{aux}}$. Suppose now that $F \subseteq E$ is the solution returned by the algorithm, for which by Theorem 4.3 we have $w(F) \leq O(\log n)B$ and $|\text{prot}(V, E\setminus F, s)| \geq T$. If $y(\text{prot}(V, E\setminus F, s)) > \mu$, then we return F as our answer, because we are guaranteed to have $F \in \mathcal{F}(O(\log n)B)$. If on the other hand $y(\text{prot}(V, E\setminus F, s)) \leq \mu$, then all $F' \in \mathcal{F}(B)$ have $y(\text{prot}(V, E\setminus F', s)) \leq \mu$, because the properties of the Section 4.1 algorithm ensure that $y(\text{prot}(V, E\setminus F, s)) \geq y(\text{prot}(V, E\setminus F', s))$. Therefore, $(y,\mu) \in Q(B)$. □

Given the existence of an algorithm like the one described in Theorem 4.6, Anegg et al. (2020) prove that with a round-or-cut approach we can either show that $Q(B) \neq \emptyset$ or that $Q(O(\log n)B) = \emptyset$. If $Q(B) \neq \emptyset$, then by Observation 4.5 we can infer $B < OPT_{\mathcal{I}}$ and return "INFEASIBLE". If on the other hand $Q(O(\log n)B) = \emptyset$, then again by Observation 4.5 we know that PLP($O(\log n)B$) is feasible. Further, in the latter case the framework of Anegg et al. (2020) provides a set $\mathcal{F}' \subseteq \mathcal{F}(O(\log n)B)$ with polynomial size, for which the following (poly-sized) LP is feasible.

$$\begin{aligned} \min\ & 0 \\ & \sum_{\substack{F\in\mathcal{F}':\\ v\in\text{prot}(V,E\setminus F,s)}} x_F \geq p_v && \forall v \in V\setminus\{s\} \\ & \sum_{F\in\mathcal{F}'} x_F = 1 \\ & 0 \leq x_F \leq 1 && \forall F \in \mathcal{F}' \end{aligned}$$

Finally, since the above can be efficiently solved, we obtain an efficiently-sampleable distribution $\mathcal{D}$ over $\mathcal{F}(O(\log n)B)$, such that $\Pr_{F\sim\mathcal{D}}[v \in \text{prot}(V, E\setminus F, s)] \geq p_v$ for all $v \in V\setminus\{s\}$.

Acknowledgements

Michael Dinitz was supported by NSF award CCF-1909111. Aravind Srinivasan was supported in part by NSF awards CCF-1422569, CCF-1749864, and CCF-1918749, as well as research awards from Adobe, Amazon, and Google. Leonidas Tsepenekas was supported in part by NSF awards CCF-1749864 and CCF-1918749, and by research awards from Amazon and Google. Anil Vullikanti's work was partially supported by NSF awards IIS-1931628, CCF-1918656, and IIS-1955797, and NIH award R01GM109718.

References

Ahmadian, S., Epasto, A., Kumar, R., and Mahdian, M. (2019). Clustering without over-representation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19.

Anegg, G., Angelidakis, H., Kurpisz, A., and Zenklusen, R. (2020). A technique for obtaining true approximations for k-center with covering constraints. In Bienstock, D. and Zambelli, G., editors, *Integer Programming and Combinatorial Optimization*.

Backurs, A., Indyk, P., Onak, K., Schieber, B., Vakilian, A., and Wagner, T. (2019). Scalable fair clustering. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 405–413.

Badger, E. (2016). How airbnb plans to fix its racial-bias problem. *The Washington Post*. September 8, 2016.

Barocas, S., Hardt, M., and Narayanan, A. (2019). *Fairness and Machine Learning*. fairmlbook.org. `http://www.fairmlbook.org`.

Benner, K., Thrush, G., and Isaac, M. (2019). Facebook engages in housing discrimination with its ad practices. *The New York Times*. March 28, 2019.

Bera, S., Chakrabarty, D., Flores, N., and Negahbani, M. (2019). Fair algorithms for clustering. In *Advances in Neural Information Processing Systems 32*, pages 4954–4965.

Bercea, I. O., Groß, M., Khuller, S., Kumar, A., Rösner, C., Schmidt, D. R., and Schmidt, M. (2019). On the Cost of Essentially Fair Clusterings. In *APPROX/RANDOM 2019*, volume 145, pages 18:1–18:22.

Brubach, B., Chakrabarti, D., Dickerson, J. P., Khuller, S., Srinivasan, A., and Tsepenekas, L. (2020). A pairwise fair and community-preserving approach to k-center clustering. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1178–1189.

Chierichetti, F., Kumar, R., Lattanzi, S., and Vassilvitskii, S. (2017). Fair clustering through fairlets. In *Advances in Neural Information Processing Systems 30*.

Datta, A., Tschantz, M. C., and Datta, A. (2015). Automated experiments on ad privacy settings. *Proc. Priv. Enhancing Technol.*, 2015(1):92–112.

Dinur, I. and Steurer, D. (2014). Analytical approach to parallel repetition. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, page 624–633.

Dwork, C., Hardt, M., Pitassi, T., Reingold, O., and Zemel, R. (2012). Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12.

Eubank, S., Guclu, H., Kumar, V. S. A., Marathe, M. V., Srinivasan, A., Toroczkai, Z., and Wang, N. (2004). Modelling disease outbreaks in realistic urban social networks. *Nature*, 429(6988):180—184.

Harris, D. G., Pensyl, T., Srinivasan, A., and Trinh, K. (2019). A lottery model for center-type problems with outliers. *ACM Trans. Algorithms*, 15(3).

Hayrapetyan, A., Kempe, D., Pál, M., and Svitkina, Z. (2005). Unbalanced graph cuts. In *Proceedings of the 13th Annual European Conference on Algorithms*, ESA'05, page 191–202, Berlin, Heidelberg. Springer-Verlag.

Huang, L., Jiang, S., and Vishnoi, N. (2019). Coresets for clustering with fairness constraints. In *Advances in Neural Information Processing Systems 32*, pages 7589–7600. Curran Associates, Inc.

Janson, S. (1998). New versions of Suen's correlation inequality. *Random Structures and Algorithms*, 13(3-4):467–483.

Karloff, H. (1991). *Linear Programming*. Birkhauser Boston Inc., USA.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, 54(6).

Räcke, H. (2008). Optimal hierarchical decompositions for congestion minimization in networks. In Dwork, C., editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 255–264. ACM.

Svitkina, Z. and Tardos, É. (2004). Min-max multiway cut. In Jansen, K., Khanna, S., Rolim, J. D. P.,

and Ron, D., editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 207–218, Berlin, Heidelberg. Springer Berlin Heidelberg.

Williamson, D. P. and Shmoys, D. B. (2011). *The Design of Approximation Algorithms*. Cambridge University Press, USA, 1st edition.

Supplementary Material: Fair Disaster Containment via Graph-Cut Problems

A MISSING PROOFS

For the next two proofs we need the following lemma.

Lemma A.1 (Räcke (2008)). *For any undirected $G = (V, E)$ with edge costs $w_e \geq 0$, we can efficiently construct a collection of trees $T_1 = (V, E_1)$, $T_2 = (V, E_2), \ldots, T_k = (V, E_k)$ with tree T_i having an edge-cost function $w^i : E_i \mapsto \mathbb{R}_{\geq 0}$, and find non-negative multipliers $(\lambda_1, \ldots, \lambda_k)$, such that $\sum_{i=1}^{k} \lambda_i = 1$ and $k = \text{poly}(|V|)$. Further, for any $S \subseteq V$ let $\delta(S)$ be the set of edges in E with exactly one endpoint in S, and $\delta_i(S)$ denote the set of edges in E_i with exactly one endpoint in S. Then, for any $S \subseteq V$:*

1. *$w(\delta(S)) \leq w^i(\delta_i(S))$ for every $i \in [k]$*
2. *$\sum_{i=1}^{k} \lambda_i w^i(\delta_i(S)) \leq O(\log n) w(\delta(S))$*

Proof of Lemma 2.2. If $\mathcal{I} = \{V, E, s, w, V_1, \ldots, V_\gamma, \vec{f}\}$ is the general instance, we first apply the result of Lemma A.1 in order to get a collection of trees $T_1 = (V, E_1), \ldots, T_k = (V, E_k)$, where each tree T_i has an associated edge weight function w^i. We then use the given algorithm and solve DemFairCut in each tree instance $\mathcal{I}_i = \{V, E_i, s, w^i, V_1, \ldots, V_\gamma, \vec{f}\}$, and get a solution $F_i \subseteq E_i$ in return. For the solution F_i we compute for $\mathcal{I}_i$, let $X_i = \text{prot}(V, E_i \setminus F_i, s)$, and note that the properties of the algorithm ensure $|X_i \cap V_h| \geq \alpha f_h n_h$, $\forall h \in [\gamma]$.

After running the algorithm in each tree instance, we find the tree T_m with $m = \arg\min_i w(\delta(X_i))$, and we set our solution for the general graph to be $\delta(X_m)$. This means that in our general solution $X_m \subseteq \text{prot}(V, E \setminus \delta(X_m), s)$. Combining this observation with the fact that $|X_m \cap V_h| \geq \alpha f_h n_h$ for all $h \in [\gamma]$, implies that in the solution for the general graph we again satisfy all demographic constraints up to an α violation. We now only have to reason about the cost of $\delta(X_m)$.

Let X^* be the set of vertices not connected to s in the optimal solution of $\mathcal{I}$. If OPT is the value of the latter, then $w(\delta(X^*)) \leq OPT$. Also, since X^* satisfies all γ demographic constraints exactly, the set $\delta_i(X^*)$ is a feasible solution for $\mathcal{I}_i$, and $OPT_{\mathcal{I}_i} \leq w^i(\delta_i(X^*))$. Hence, because $\delta_i(X_i) \subseteq F_i$:

$$w^i(\delta_i(X_i)) \leq \rho \cdot OPT_{\mathcal{I}_i} \leq \rho \cdot w^i(\delta_i(X^*)) \tag{6}$$

Using the definition of m and the first property of the trees from Lemma A.1 gives

$$w(\delta(X_m)) \leq \sum_{i=1}^{k} \lambda_i w(\delta(X_i)) \leq \sum_{i=1}^{k} \lambda_i w^i(\delta_i(X_i)) \tag{7}$$

Combining (6), (7) and the second property of Lemma A.1 yields

$$w(\delta(X_m)) \leq \rho \sum_{i=1}^{k} \lambda_i w^i(\delta_i(X^*)) \leq \rho \cdot O(\log n) \cdot w(\delta(X^*)) \leq \rho \cdot O(\log n) \cdot OPT \qquad \square$$

Proof of Lemma 2.4. Let $\mathcal{I} = (V, E, B, T, w, s, a)$ be an instance of AuxCut for a general graph. We first apply the result of Lemma A.1 in order to get a collection of trees $T_i = (V, E_i)$ with edge-weight functions w^i. Then, for each such tree we create an instance $\mathcal{I}_i = (V, E_i, B \cdot O(\log n), T, w^i, s, a)$, and we use the given bicriteria algorithm to solve AuxCut on it. Let $F_i \subseteq E_i$ the solution we get for $\mathcal{I}_i$, and for notational convenience let again $X_i = \text{prot}(V, E_i \setminus F_i, s)$. After that, we find the tree T_m with $m = \arg\max_i a(X_i)$, and we set our solution for the general graph to be $\delta(X_m)$. This means that in our general solution we again get $X_m \subseteq \text{prot}(V, E \setminus \delta(X_m), s)$.

At first, because of the properties of the algorithm used on $\mathcal{I}_i$, we have $|X_m| \geq T$, and therefore even in our solution for the general graph we end up saving at least T vertices.

Furthermore, because $\delta_i(X_i) \subseteq F_i$, the properties of the bicriteria algorithm give $w^i(\delta_i(X_i)) \leq \rho \cdot O(\log n) \cdot B$ for every i. From the first property in Lemma A.1 we thus get

$$w(\delta(X_m)) \leq w^m(\delta_m(X_m)) \leq \rho \cdot O(\log n) \cdot B$$

To conclude we need to show that $a(X_m) \geq OPT_{\mathcal{I}}$, where $OPT_{\mathcal{I}}$ the value of the optimal solution of $\mathcal{I}$. Let also X^* be the set of vertices not connected to s in the optimal solution of $\mathcal{I}$. Since X^* is the optimal such set of vertices, we have $|X^*| \geq T$ and $w(\delta(X^*)) \leq B$. Moreover, let $m^* = \arg\min_i w^i(\delta_i(X^*))$. The definition of m^* and the second property from Lemma A.1 give

$$w^{m^*}(\delta_{m^*}(X^*)) \leq \sum_{i=1}^{k} \lambda_i w^i(\delta_i(X^*)) \leq O(\log n) w(\delta(X^*)) \leq B \cdot O(\log n)$$

Hence $\delta_{m^*}(X^*)$ is feasible for $\mathcal{I}_{m^*}$ (recall that $|X^*| \geq T$), and since the given algorithm is a $(1, 1, \rho)$-bicriteria we get $a(X_m) \geq a(X_{m^*}) \geq OPT_{\mathcal{I}_{m^*}} \geq a(X^*) = OPT_{\mathcal{I}}$. □