Making Existing Clusterings Fairer: Algorithms, Complexity Results and Insights

Ian Davidson[1] **S. S. Ravi**[2]
[1] Computer Science Department, University of California, Davis
[2] Biocomplexity Institute & Initiative, University of Virginia and
Computer Science Department, University at Albany – SUNY
davidson@cs.ucdavis.edu, ssravi0@gmail.com

Abstract

We explore the area of fairness in clustering from the different perspective of modifying clusterings from existing algorithms to make them fairer whilst retaining their quality. We formulate the minimal cluster modification for fairness (MCMF) problem where the input is a given partitional clustering and the goal is to minimally change it so that the clustering is still of good quality and fairer. We show using an intricate case analysis that for a single protected variable, the problem is efficiently solvable (i.e., in the class **P**) by proving that the constraint matrix for an integer linear programming (ILP) formulation is totally unimodular (TU). Interestingly, we show that even for a single protected variable, the addition of simple pairwise guidance (to say ensure individual level fairness) makes the MCMF problem computationally intractable (i.e., **NP**-hard). Experimental results on Twitter, Census and NYT data sets show that our methods can modify existing clusterings for data sets in excess of 100,000 instances within minutes on laptops and find as fair but higher quality clusterings than fair by design clustering algorithms.

1 Introduction

Existing work on clustering and fairness takes a known clustering *algorithm* and modifies it produce fair results. The seminal work of Chierichetti et al. (2017) looked at k-center and k-median style algorithms whilst later work has explored other formulations such as spectral clustering (Kleindessner et al. 2019). However, there is a plethora of different clustering algorithms, with this decade old survey (Xu and Wunsch 2005) listing over 15 popular partitional clustering algorithms with a variety of settings, formulations and followings by end user communities. It is unlikely that fair versions of all these algorithms or new clustering algorithms will be developed. Furthermore, often clusterings results are already deployed. In these circumstances, one research direction is to modify an existing *clustering* to make it fairer whilst not unduly changing its quality. This paper considers this precise situation, where one already has a good clustering Π and the goal is to modify Π to improve its fairness with respect to a set P of protected variables. We focus on the simplest but most common type of protected variables, namely binary variables such as gender.

A Flexible Efficient Approach To Ensure Fairness. Our approach places upper and lower bounds on the number of protected status individuals in each cluster which can be any constant expression and these bounds may even be different for each cluster. For example, if a data set D has N special items and D has been partitioned into k clusters, then the number of special data items per cluster could be set to be approximately $\frac{N}{k}$, effectively balancing the protected status instances uniformly across all clusters. Alternatively, we could require each cluster to have approximately $\frac{N.|C_i|}{|D|}$ (where $|C_i|$ is the size of cluster i) protected status individuals, this would require that the ratio of the number of special items to the size of the cluster be (approximately) the same for all clusters. We now provide two fairness definitions. The first is useful to prove total unimodularity (TU) but in practice the second is more pragmatic and also gives rise to a TU constraint matrix (see Theorem 3.2):

Definition 1.1. *Let D be a dataset where each data item has a single binary protected attribute x. Let N_x denote the number of special data items in D. A partition of D into $k \geq 2$ is **strongly fair with respect to** x if in each cluster, the number of special items is either $\lfloor N_x/k \rfloor$ or $\lceil N_x/k \rceil$.*

This is useful in our intractability results and algorithm design. However, it is a strong requirement; hence we define a relaxed notion called α-fairness as follows:

Definition 1.2. *Consider a dataset D where each data item has a single binary protected attribute x. Let $D_x \subseteq D$ denote the subset of special data items and let $N_x = |D_x|$. Let α be a positive integer $< \lceil N_x/k \rceil$. A partition of D into $k \geq 2$ clusters is α-**fair** wrt. x, if in each cluster, the number of elements from D_x is in the range $[\lceil N_x/k \rceil - \alpha \ .. \ \lceil N_x/k \rceil + \alpha]$.*

ILP Formulations for the Minimal Cluster Modification for Fairness (MCMF) Problem. A natural way to formulate the MCMF problem is as a discrete optimization problem where the goal is to minimize the effect of modifying the clusters (see Section 3 for a precise formulation). Here, the effect can be in terms of the number of instances moved or even changes in the quality of the clustering. While solving an ILP is, in general, computationally intractable, for our formulation of the MCMF problem, we show that (see Theorem 3.2) the constraint matrix our fairness requirements introduce are *totally unimodular* (TU) (Schrijver 1998). This proof involves an intricate case analysis so is presented in (Davidson and Ravi 2019). As a consequence, one can use any polynomial time linear programming (LP) algorithm to obtain integer solutions to MCMF.

Summary of Main contributions.

(1) We define a novel minimal modification problem to produce fair clustering which can post-process the results of **any** partitional clustering method to make them fairer whilst ensuring their clustering quality is not affected significantly. This is different from existing work that attempts to take an existing clustering algorithm and produce a fair variant of it.

(2) We formulate the MCMF problem as an ILP and show that under our definitions of fairness, the formulation produces a constraint matrix that is **totally unimodular** (see Section 3). This constraint matrix can be paired with a variety of objectives to ensure clustering quality is not affected significantly (see Table 1). This leads to algorithms capable of modifying clusterings with millions of instances since there are well known LP solvers that run in polynomial time. Some variations to ensure fairness such as allowing overlapping clusters also have the TU property (see Section 4).

(3) If even larger clusterings are studied, in (Davidson and Ravi 2019) we show an algorithm whose worst-case run time is better than an LP solver (see Section 3).

(4) Interestingly, we show that though finding a feasible strong fair clustering (an example of group level fairness (Barocas and Selbst 2016)) is in **P** or finding a feasible clustering to satisfy commonly using popular **must-link** constraints (Basu, Davidson, and Wagstaff 2008) which can encode individual level fairness (Barocas and Selbst 2016) is in **P**, finding a feasible clustering to satisfy **both** requirements is computationally intractable (see Section 5, Theorem 5.1).

(5) Experimental results show that our method is computationally efficient (as expected) and our objectives useful for post-processing the results from k-means, k-medians and spectral clustering algorithms. The experiments are for Census/Adult (48K), NYT (300K) and Twitter Healthcare (58K) datasets. We show that our method of post-processing can produce similar fairness results as jointly finding a good but fair clustering (Chierichetti et al. 2017).

Organization. We begin by discussing related work in section Section 2. We then formulate the minimal clustering modification for fairness (MCMF) problem in Section 3 as an ILP and show that its constraint matrix is totally unimodular. We establish the complexity of achieving fairness while satisfying instance-level constraints in Section 5. We then present results from our experiments in Section 6. Conclusions and directions for future work are provided in Section 7. Reference (Davidson and Ravi 2019) contains a linear time algorithm the the MCMF problem.

2 Related Work

We briefly review two related work areas, namely fairness in machine learning (ML) and minimal modification of clustering. Fairness in ML is an emerging area that has received much attention in the context of supervised learning, often under different names such as algorithmic bias (Thanh, Ruggieri, and Turini 2011). More recently, in clustering (i.e., unsupervised learning), the issue of fairness generally aims at balancing protected individuals across clusters as we have.

All of the work below uses a similar fairness measure (shown below) and focuses on simple k-means/medians/centers algorithms with the exception of (Kleindessner et al. 2019) which explores spectral clustering. The idea of balancing protected individuals aims to address the disparate impact doctrine (Feldman et al. 2015; Friedler, Scheidegger, and Venkatasubramanian 2016) and was formalized in the seminal work of Chierichetti et al. (2017). They assumed that each object has one of two colors (red or blue). Letting the number of instances of each type in cluster i be R_i and B_i respectively, the fairness of a clustering is $\min(\min[\frac{R_1}{B_1}, \frac{B_1}{R_1}], \ldots, \min[\frac{R_k}{B_k}, \frac{B_k}{R_k}])$. Their work creates fairlets (groups of instances) which when post processed by k-center and k-medians are guaranteed to produce a specified level of fairness and achieve a constant factor approximation with respect to cluster quality. As seen in Table 4, by ensuring that cluster sizes are not unduly changed, the Chierichetti et al. (2017) measure of fairness and our measure can yield similar results. However, a stronger statement regarding their equivalence is left to future work.

The work of Backurs et al. (2019) showed how a fairlet decomposition algorithm can be implemented to run in nearly linear time. The work of Rösner and Schmidt (2018) extended the work of (Chierichetti et al. 2017) by allowing objects with more than two colors (i.e., three or more protected attributes) but assume that each object has only one color. Bera et al. (2019) also consider three or more colors and allow an object to have more than one color. They also allow users to specify upper and lower bounds on fairness measures for each cluster and develop clustering algorithms under any ℓ_p norm.

In this paper we take an alternative path of improving fairness by **post-processing** (i.e., by minimal modification of) clustering results produced by existing methods. The idea of minimal modification of clustering solutions has been explored before by ourselves (Kuo et al. 2017); however, that focus is on human-in-the-loop style settings where the domain expert can choose to adjust geometric properties of the cluster such as diameter. Hence, the theorems and results these two papers are fundamentally different. Further, the focus of this earlier work was on improving cluster quality by moving a small number of instances between clusters; it did not take fairness into consideration. Further, due to the use of the constraint programming, this earlier work scales only to data sets of size at most 1000. In contrast, our modification algorithm for improving fairness scales to very large data sets (hundreds of thousands of points) even on a laptop (see Section 6).

3 An ILP Formulation and Proof of Total Unimodularity

Here we show an ILP formulation of the MCMF problem which can find fairer clusters. The reader may also refer to (Davidson and Ravi 2019) for implementation details.

Measure	Meaning of Objective Function
w = 1	The number of instances moved.
$w_i = \sum_j \frac{d(i,C_j)}{k} - d(i,C^*)$	The increase in mean L2 (distortion) or L1 (median) distances (average distance from i to cluster j: $d(i,C_j)$).
$w_i = \sum_j E(i,j) : (i,j) \in \Pi_a$	The increase in external degree by moving instance i away from cluster Π_a.

Table 1: Several penalty schemes and their meaning when used in Equation (1). Each measures the increase if i is moved away from the cluster (C^*) to which it is assigned in Z^*. We assume that Z^* is optimal for the given objective.

We then show the resultant constraint matrix is totally unimodular (TU) and hence the ILP is efficiently solvable. Our formulation can be used to ensure **any** upper and lower bound on the number of protected variables in a cluster. These bounds need not be the same for each cluster and most importantly, since TU depends on the constraint matrix coefficients and not on the right-hand side of the constraints, these formulations are also efficiently solvable.

Our aim is the following: given a desirable existing clustering (defined in an $k \times n$ allocation matrix Z^*), find a minimal modification that makes the clustering fairer with respected to the single protected variable P. In our experiments, we consider a formulation to allow many protected variables. We also discuss other settings which lead to TU constraint matrices.

Objective. We wish to find another allocation matrix Z that is fairer but similar to Z^*. As z_i^* and z_i are both binary **column** vectors indicating what single cluster the i^{th} instance belongs to, $\sum_i (z_i^*)^T \times (z_i)$ counts the number of agreements between Z and Z^* which forms the basis of useful objectives. We can easily encode preferences/importance amongst instances by having a penalty (w_i) (see Equation (1)) if instance i is moved which can take on a variety of semantic meanings. See Table 1 for some examples. The first simply minimizes the number of instances moved, the second is useful for centroid based methods such as k-means since it minimizes the increase in distortion (assuming that the existing solution minimizes the distortion) and the last is useful for graph based formulations since it minimizes increase in cut cost (again assuming that the existing solution minimizes the mincut). Of course, domain experts can easily encode their own preference schemes.

$$argmin_Z \sum_i w_i[1 - (z_i^*)^T \times (z_i)] \tag{1}$$

Adding Constraints With Slack Variables. The aim of the constraints are two-fold, to balance the protected variable whilst also restricting Z to be a legal cluster allocation matrix. Note that we use the encoding where indicator vectors are stacked column-wise, that is $z_{i,j} = 1$ iff instance j is assigned to cluster i. We encoded the protected status as a vector P of length n with an entry of 1 if the instance has the status otherwise 0. Our first two constraints require that the distribution of the protected variable be upper and lower bounded. For example, to follow our definition of strict fairness (see Definition 1.1) we would have the constraint $\lfloor \frac{|P|}{k} \rfloor \le \sum_j p_i z_{i,j} \le \lceil \frac{|P|}{k} \rceil \ \forall i$ with u (upper) and l (lower) being the slack variables. In the following equation we generalize this to any upper and lower bounds and note they can vary depending on the cluster. (We use $|P|$ to denote the number of non-zero entries in the vector P.) The last set of constraints below (i.e., $\sum_i z_{i,j} = 1 \ \forall\, j$) simply require that Z is a valid allocation matrix. Note again that the instances are stacked column-wise in Z.

$$\sum_j p_j z_{i,j} + u_i = U_i, \ \forall i \tag{2}$$

$$-\sum_j p_j . z_{i,j} + l_i = -L_i, \ \forall i \tag{3}$$

$$\sum_i z_{i,j} = 1, \ \forall j \tag{4}$$

Note that when $U_i = \lceil |P|/k \rceil$ and $L_i = \lfloor |P|/k \rfloor$, for $1 \le i \le k$, Lemma 3.2 points out that **there is always** a solution to the above set of constraints. We will make a similar observation regarding a relaxed version of the fairness requirement later.

Total Unimodularity of Constraint Matrix. It is well known (Schrijver 1998) that if the *constraint matrix* of an ILP is totally unimodular (TU) then we can solve the problem using an *LP* (linear program) solver and the solution will still be integral. Further, linear programming problems can be solved in $O(n\,(n+d)^{1.5}\,L)$ time, where n is the number of variables, d is the number of constraints and L is the total number of bits needed to encode all the constants specified in the LP (Vaidya 1989). This running time is clearly polynomial in the input size.

In the above equations, there are kn **regular variables** (namely, z_{11}, z_{12}, ..., z_{1n}, ..., z_{k1}, z_{k2}, ..., z_{kn}) and $2k$ **slack variables** (namely u_1, ..., u_k and l_1, ..., l_k). For the purpose of constructing the constraint matrix C, we will use the following order of these $kn + 2k$ variables: $\langle z_{11}, z_{12}, \ldots, z_{1n}, \ldots, z_{k1}, z_{k2}, \ldots, z_{kn}, u_1, \ldots, u_k, l_1, \ldots, l_k \rangle$. Matrix C has $2k + n$ rows (one corresponding to each constraint) and $nk + 2k$ columns (one corresponding to each variable). In C, we will list the $2k$ constraints corresponding to Equations (2) and (3) in the order specified by those equations. This is followed by the n constraints in the order specified by Equation (4). Note that each entry of C is from $\{-1, 0, +1\}$. In each row of C (which specifies one constraint), we will list the coefficients of the $kn + 2k$ variables in the order specified above. We refer to the first kn columns of C as **regular variable columns** and the last $2k$ columns as **slack variable columns**. Using this terminology, we can prove the following lemma.

Lemma 3.1. *(a) In any regular variable column of C, there are at most three non-zero elements. (b) In any slack variable column of C, there is exactly one element with value 1; the other entries in that column are 0.*

Proof: See (Davidson and Ravi 2019).

To prove the TU property of the constraint matrix C, we will use the following result, which is Theorem 19.3 in (Schrijver 1998).

Theorem 3.1. *TU Identity (Schrijver 1998) Let C be a matrix such that all its entries are from $\{0, +1, -1\}$. Then C is totally unimodular, i.e., each square submatrix of C has determinant 0, $+1$, or -1 if every subset of rows of C can be split into two parts A and B so that the sum of the rows in A minus the sum of the rows in B produces a vector all of whose entries are from $\{0, +1, -1\}$.* ■

Theorem 3.2. ***The matrix C formed by the coefficients of the constraints used to encode Equations (2) through (4) is totally unimodular.***

Proof: See (Davidson and Ravi 2019).

An Alternative More Efficient Algorithm Only For Strict Fairness. It is also possible to obtain another efficient algorithm for MCMF using the following idea. Given an arbitrary distribution of the special items into k clusters, we use Lemma 3.2 (below) to identify which clusters have an "excess" amount of special items and which ones are "deficient" with respect to special items. It can be seen using Lemma 3.2, the total number of excess items gives the lower bound on the number of special items that must be moved to achieve strong fairness. The algorithm provides an optimal solution by ensuring that the number of special items moved between clusters is equal to the lower bound. As the details of the algorithm involve many cases, a description of the algorithm is given in (Davidson and Ravi 2019). This algorithm's worst-case running time is **better** than that of an LP solver.

The following result shows that there exists a distribution of special items into clusters that is a necessary and sufficient condition for a strongly fair clustering of D.

Lemma 3.2. ***[A necessary and sufficient condition for Strong Fairness.]*** *Let D be a data set with one binary protected attribute x. Let $D_x \subseteq D$ denote the subset of special data items and let $N_x = |D_x|$. Let q and r be non-negative integers such that $N_x = qk + r$ with k being the number of clusters and $0 \leq r \leq k - 1$. A partition of D into k clusters is strongly fair with respect to x if and only if it has exactly r clusters each with $\lceil N_x/k \rceil$ special data items and $k - r$ clusters each with $\lfloor N_x/k \rfloor$ special data items.*

Proof: See (Davidson and Ravi 2019).

4 Other Variations that Are and Are Not TU

Here we discuss variations some of which lead to constraint matrices with the TU property while others do not. It is important to bear in mind that the proof of TU only depends on the coefficients of the constraint matrix. It does not depend on the objective function (which is why we can have the variations such as those in Table 1); nor does it depend on the right hand side of the constraints.

Allowing Overlapping Clusters. A desirable situation to enforce fairer cluster is to allow an instance to belong to multiple (s) clusters. Our slack variable formulations easily facilitates an instance belongs to at most s clusters. This has the benefit of spreading the protected individuals to multiple clusters, that is we have $\sum_i z_{i,j} = s, \; \forall j$ but since this does **not** change the coefficients of the constraint matrix (only the constant in the equality) this formulation is also TU.

Multiple Protected Variables. When there are $r \geq 2$ protected variables, $r - 1$ additional sets of constraints similar to Equations (2) and (3) must be added. Since the same data item may have many protected attributes, it is not clear whether the resulting constraint matrix satisfies the TU property. Determining if this case satisfies the TU property will be left for future work.

Weighted or Continuous Protected Variables. The proofs of TU require the constraint matrix to contain entries of only $\{-1, 0, +1\}$. In general, any work that requires degrees of protection (i.e., age) even if encoded in ordinal form cannot be encoded as TU matrix to our knowledge.

5 Difficulty of Satisfying Group and Individual Level Fairness

Our measure of strong fairness (Definition 1.1) is a group (cluster) level measure (Barocas and Selbst 2016). An alternative measure of fairness is individual level (Barocas and Selbst 2016) where we require similar individuals to be treated/clustered the same. This can be encoded as the popular **must-link** (ML) constraints (Wagstaff and Cardie 2000; Basu, Davidson, and Wagstaff 2008) where the constraint ML(a, b) requires data items a and b to be in the same cluster.

As shown in Theorem 3.2 satisfying strong fairness is computationally tractable. Similarly the **feasibility** problem with respect to ML constraints (i.e., given a data set D, an integer k and a set S of ML constraints, can D be partitioned into k clusters so that all the ML constraints in S are satisfied?) can also be solved efficiently (Davidson and Ravi 2007). However, we now show satisfying **both** requirements is computationally intractable (Theorem 5.1). We start with a definition of the corresponding feasibility problem.

Feasibility of Strongly Fair Clustering under ML Constraints (FSFC-ML)

Instance: A dataset D where each item has a set of attributes, a protected attribute x, an integer $k \leq |D|$, a set S of ML constraints.

Question: Can D be partitioned into k clusters so that the resulting clustering (i) is strongly fair with respect to x and (ii) satisfies all the ML constraints in S?

The following result points out that FSFC-ML is computationally intractable.

Theorem 5.1. *Problem FSFC-ML is **NP**-complete.*

Proof: We use a reduction from the 3-PARTITION problem (Garey and Johnson 1979). The details are in (Davidson and Ravi 2019).

A consequence of Theorem 5.1 is that the minimum modification problem where the goal is to achieve group level fairness (as per our definition) and individual level fairness is computationally intractable.

6 Experimental Results

To illustrate the usefulness of our method we explore several large data sets (Adult/Census, Twitter Healthcare and NYT)

on both k-means, k-medians and spectral clustering algorithms. Since no other work attempts to post process results to make them fairer we do not present the standard "Us vs Them" tables of results but instead attempt to illustrate our work's uses, limitations and comparisons to fair-by-design clustering algorithms. We attempt to answer the following:
Q1. What is the impact of our modification approach on real world data sets? Can our objectives in Table 1 find fairer clusters whilst **also** retaining high quality clustering?
Q2. How does making existing clusterings fairer compare to approaches that find fair clusterings to begin with (e.g., Chierichetti et al. (2017))?
Q3. What is the approximate run time of our method and the impact of increasing the number of instances and clusters?

We begin with an illustrative data set ($\approx$ 50k instances) used by many previous fairness papers and the move onto a larger collection of data sets ($\approx$ 58K and 300k instances).

6.1 Q1 - Effects of Post-Processing

Here we first analyze the well studied `Adult` dataset (e.g., (Chierichetti et al. 2017; Backurs et al. 2019)) that consists of 48,842 individuals (males 66.8%, females 33.2%) from the UCI repository (Dheeru and Karra Taniskidou 2017).
Case Study: Post-Processing Results of k-Means. The best clustering result of partitioning this data into 5 clusters using k-means is shown in Table 2. We immediately see that the first two clusters are desirable from a marketing perspective as they consist of highly educated individuals with high gains (related to income) who can be targeted for better loans, credit cards, ads etc. to them. However, they are overwhelmingly male, with no more than 21% of the total population per cluster being female. Note the proportion of females in this data set is 33.2%.

To make these first two clusters fairer we apply our method by placing bounds on the first and second cluster's protected status ratios to be 0.5 ± 0.05 with the remaining clusters' proportion of females to be their current values as reported in Table 2 ± 0.15. This is achieved by setting the U_i and L_i bounds in Equations (2) and (3). We then applied the minimal modification method with the **second objective** in Table 1 as it is compatible with the k-means objective. The results are shown in Table 3. Since we used k-means clustering we measure the impact of our modified clustering in terms of the increase of the distortion (the objective function used by k-means). We found that female instances from Cluster 4 were placed in Cluster 1 and Cluster 2.

The results show several interesting insights:
1. We find that when only modifying to make clusters fairer, the distortion only increased by 2%. This indicates our objective function in Table 1 is useful at ensuring the clustering quality is not diminished.
2. However, the description and sizes of the clusters do change (highlighted by bold in Table 3) sometimes adversely. For example, the second cluster now becomes less desirable from a marketing perspective as it contains less educated individuals who are not married. This motivates our next experiments on multiple protected attributes.
Experiments with multiple protected attributes to overcome challenges. The last item in the above list is a chal-

Cluster	1	2	3	4	5
Female	21%	12%	25%	51%	14%
Size	5352	2776	15180	20182	5352
age	42	47	43	31	46
educ.	Bachelors	Bachelors	HS-grad	Some-college	Some-college
status	Married	Married	Married	Never	Married
occup.	Prof	Sales	Craft	Prof	Exec
gain	3910	2887	353	233	2556

Table 2: For k-means and census dataset. A description of the best clustering found using $k = 5$ (minimized distortion over 1000 random restarts) and the fraction of the protected variable (females) per cluster. The distortion of the solution is 110402.48. This is the given clustering we shall minimally modify to obtain results in Tables 3 and 4.

Cluster	1	2	3	4	5
Female	45%	45%	34%	25%	28%
Size	5923	**6321**	**10231**	**14001**	**12366**
age	41	43	44	35	45
educ.	Bachelors	**HS-grad**	HS-grad	Some-college	**Bachelors**
status	Married	**Never**	Married	Never	Married
occup.	Prof	Sales	Craft	Prof	**Sales**
gain	2834	2532	1431	452	2641

Table 3: For k-means, census dataset and our method. A description of the clusters found using our method (using second objective in Table 1) by minimally modifying the clustering described in in Table 2. The distortion of this solution increased approximately 2% to 112400.68. Compare with Table 2. Interesting changes between that table are bolded.

Attribute Focus	Distortion Increase	Fairness Decrease per (Chierichetti et al. 2017)
Education	2.1%	1.3%
Marital Status	1.8%	2.5%
Education + Marital Status	8.0%	3.6%
Keep Cluster Sizes $\pm 0.05\%$	15.4%	0.1%
Education + Keep Cluster Sizes $\pm 0.05\%$	19.8%	0.8%
Marital Status + Keep Cluster Sizes $\pm 0.05\%$	20.3%	0.9%
Education + Marital Status + Keep Cluster Sizes $\pm 0.05\%$	23.6%	0.8%

Table 4: k-means, census dataset and our method with multiple constraints on variables. The distortion increase of the modified to be fairer (for gender) clustering over the clustering in Table 2 but we now require other properties in Table 2 to be retained.

lenge with just balancing a single protected variable. To address this, we can constrain other variables, even though they are not protected. It is important to realize that we can constrain the size of the clusters by creating a *dummy protected variable* that every instance possesses. Thus to better ensure fairness (wrt to gender) across the clusters whilst retaining other properties of the two desirable clusters we also constrain `education, marital-status`. The increase in distortion for these more complex experiments is shown in Table 4. Not surprisingly, the requirement of keeping cluster sizes similar to their previous values produces a greater increase in distortion. Next we measure the fairness of our clusterings using the classic fairness measure of (Chierichetti et al. 2017). As expected we find (Table 4 last column) no large difference as both measures are based on cardinality. In question Q2 we explore whether the two methods produce different results.

More Data Sets and Experiments With k-Means and Spectral Clustering. We now explore two larger data sets: (i) the NYT Articles Bags of Words Data Set (300,000 instances) and (ii) Twitter Data of Health News (58,000 instances). Each data set is represented by the 1000 most frequent words including gender (male, female), race (black, hispanic, white) and age (elderly, young). The former data set is already processed whilst we processed the latter using the BOW toolkit (https://www.cs.cmu.edu/~mccallum/bow/). For each data set we find the best $k = 10$ clustering using plain k-means and spectral clustering + k-means (both from 1000 random restarts). We then reported the increase in distortion and cut cost by ensuring fairness across all clusters for a variety of key words mentioned in Table 5 and Table 6. For spectral clustering (von Luxburg 2006) we created a fully connected graph based on the cosine distance between bags of words vectors and then created a spectral embedding into 10 dimensional space and used k-means to find 10 clusters. We used our **third objective** function in Table 1 which is in principle similar to the spectral clustering objective function (from a graph cut perspective); see Tables 5 and 6.

As before we found that modifying a clustering to ensure fairness for a **single** protected attribute can be achieved by minimally increasing the objective function of the algorithm. However, balancing **multiple** protected attributes produces a greater increase than for the sum of the increase for the same two variables. For example in Table 5 balancing `Female` and `Black` produced a distortion increase of 5.9% but just `Female` or just `Black` produces increases of 1.3% and 1.9% respectively.

6.2 Q2 - Direct Fair Clustering Comparison

Here we answer the important question of how post-processing an existing clustering to make it fairer compares to finding fair clusters to begin with. In Table 4 we showed that the classic measure of fairness (Chierichetti et al. 2017) (see section 2) is similar to our own as they are both cardinality based. However, this is different from the question of does post-processing to increase fairness find the same, better or worse clusterings as attempting to find a fair clustering to begin with. To explore this question, we used the scalable version of (Chierichetti et al. 2017), that is, the work

Word Focus	Distortion Increase	Cut Cost Increase
Base Clustering Method	0	0
`Female`	1.3%	1.8%
`Black`	1.9%	2.3%
`Elderly`	2.3%	3.1%
`Female, Black`	5.9 %	7.2%
`Female, Elderly`	6.8%	8.3%
`Black, Elderly`	7.1%	8.9%
`Female, Black, Elderly`	13.9%	17.3%
`Female` + Cluster Sizes $\pm$ 5%	18.1%	20.3%

Table 5: k-means, spectral clustering and our method for NYT data set. The increase in distortion if we minimally modify the clustering of the NYT Articles Bag of Words Data Set with 10 groups using k-means and spectral clustering. Each row shows the increase in distortion and cut-cost caused by a fairness requirement.

of (Backurs et al. 2019) which implements fair k-medians. The work on fair spectral clustering (Kleindessner, Awasthi, and Morgenstern 2019) could be a suitable comparison but for our data sets of 48k, 58k and 300k instances it was not scalable as the resultant affinity matrices were nearly 300Gb large (i.e., to encode a 300k x 300k matrix of short integers).

We performed two experiments. Firstly, we ran both methods (k-medians[1] plus ours (objective function 2 with L2 distance in Table 1) and (Backurs et al. 2019) for k-medians) on the *same* collection of bootstrapped samples (50% of the original data set size) for our three data sets and measured the normalized Rand Index between the clusterings found by the two methods. If the Rand Index were 1 the clusterings found are identical. Table 7 (2nd column) shows our methods do not find the exact same clustering. However, if we post-process (using our method and a constraint to retain cluster sizes) the result of the (Backurs et al. 2019) method it does not unduly change the resultant clustering (column 3 Table 7).

We next explored how the output of the two methods are different. To achieve this we plot the census data experiments in a 2D scatter plot where one dimensions is the objective of the k-medians algorithm whilst the other is the fairness criterion used by the algorithms. Since the two notions of fairness used are similar but not identical we have two plots in Figure 1. We find that as expected each method is better at optimizing its own measure of fairness but our method is on average better at finding more compact clusters (according to the objective of k-medians). This is not unexpected as the work of (Backurs et al. 2019) guarantees fairness but has a weaker approximation bound than MATLAB's k-medians implementation.

6.3 Q3 - Scalability

The TU proof ensures that an ILP formulation can be solved by an LP solver, but this can still take polynomial time. For our previous data sets we found that instances of the `Adult`

[1]The theory and applied literature use different terms for the same algorithm. We use the k-medoid MATLAB algorithm which is referred to the k-medians algorithm in the theory literature.

Word Focus	Distortion Increase	Cut Cost Increase
Base Clustering Method	0	0
`Female`	2.4%	2.1%
`Elderly`	3.2%	3.8%
`Female, Elderly`	7.4%	8.8%
`Female` + Cluster Sizes ± 5%	17.3%	19.4%

Table 6: k-means and spectral clustering and our method for Healthcare Data Set. The increase in distortion if we minimally modify the clustering of the Twitter Healthcare Data Set with 10 groups using k-means and spectral clustering. Each row shows the increase in distortion and cut-cost caused by adding a fairness requirement.

Data Set	Adjusted RI	Change in Fairness (number of instances moved) after post-processing results from (Backurs et al. 2019).
Adult/Census	0.95	0.18% (0.05%)
NYT	0.75	1.1% (0.13%)
Healthcare	0.85	1.3% (0.11%)

Table 7: Comparison of post-processing for fairness versus searching for fair clusterings for 350 bootstrap samples each of the Census, NYT and Twitter data sets. The second column shows the Rand Index (RI) between the clustering each method finds averaged over 100 bootstrap samples. The third column shows how applying our method *after* finding a fair clustering decreases the fairness.

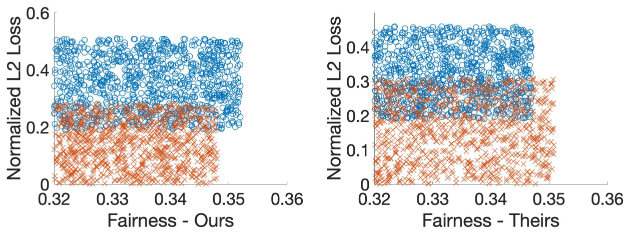

Figure 1: For 350 boostraps of the Census data set, the comparison of our method (crosses) vs Chierichetti et al. (2017) (circles) of the k-medians loss versus our measure of fairness (left) and Chierichetti et al. measure (right).

k	**run-time**
2	3.8s/4.3s
4	6.1s/6.5s
8	8.3s/9.1s
16	27.30s/23.1s
32	75.43s/85.1s

n	**run-time**
1000	0.20s/0.29s
2000	0.81s/0.95s
4000	1.11s/1.45s
8000	3.23s/4.93s
16000	16.41s/18.55s
32000	69.32s/73.81s

Table 8: Scalability and NYT (left) and Twitter Health care (right) data sets. The mean run time over 100 experiments on a single core of a MacBook Pro laptop (i5 processor) for a randomly created subset of the data sets. **Left:** 10,000 instances data set and varying numbers of clusters. **Right:** 5 clusters and varying sized data set.

data set took under one minute to run on a single core of a MacBook laptop. For the larger NYT data set time was under 5 minutes and for the Twitter data set it was 4 minutes. Here we wish to see how the run time of our algorithm is affected by increasing the number of clusters and number of instances. We explore the laptop run time for the NYT and Twitter Healthcare data sets in Table 8 of various samples. We averaged results over 100 experiments with 25 experiments each balancing Female, Black, Elderly and Hispanic to match the population ratios.

7 Discussion and Conclusions

We explored the novel idea of post-processing the results of existing clustering algorithms to make them fairer. We formulated the problem as an ILP and showed using an intricate case analysis that the resultant constraint matrix is totally unimodular (TU). This means that we can solve the ILP using an LP solver and thus obtain a polynomial time algorithm. We showed some variations such as a relaxed condition for fairness, overlapping clusters and importance penalty functions also have TU constraint matrices. However, the TU requirement means that interesting settings such as continuous protected variables may not be efficiently solvable.

Our complexity results showed an interesting conundrum. Though finding a strictly fair clustering for a single protected status variable (a type of group level fairness) is tractable and though finding a clustering to satisfy popular **must-link** constraints (which can encode individual-level fairness) is also tractable, satisfying **both is intractable**.

Our experiments aimed to shed light on the strengths and limitations of the approach and the general problem of making clusterings fairer. We found that though we were able to improve the fairness of large data sets efficiently on standard laptops some as big as 300K instances in 5 minutes or under, we observed several interesting phenomena when attempting to find fair clusters. Firstly, making existing clusterings fair for a single protected variable can be achieved with minimal decrease in the clustering quality for a variety of clusterings produced by fundamentally different algorithms (k-means and spectral clustering). But this could have the effect of unduly influencing the composition of the clustering (e.g., cluster 2 in Table 3). We showed how this could be addressed by using our formulation to balance multiple variables (even though they are not protected) including the cluster sizes. However, balancing multiple protected variables can decrease the cluster quality substantially. We show that our measure of fairness does not produce fundamentally different results than that of the seminal work in the field (Chierichetti et al. 2017) by showing (for example) that post-processing the results of their output minimally changes the clustering. However, our method does have the benefit of not being tied to a particular clustering algorithm and is scalable due to our TU result.

Acknowledgments: We thank the referees for helpful comments. This work was supported in part by NSF Grants IIS-1908530 and IIS-1910306 titled: “Explaining Unsupervised Learning: Combinatorial Optimization Formulations, Methods and Applications”.

References

Backurs, A.; Indyk, P.; Onak, K.; Schieber, B.; Vakilian, A.; and Wagner, T. 2019. Scalable fair clustering. *To Appear in ICML.*

Barocas, S., and Selbst, A. D. 2016. Big data's disparate impact. *California Law Review* 671:671–732.

Basu, S.; Davidson, I.; and Wagstaff, K. 2008. *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press.

Bera, S. K.; Chakrabarty, D.; and Negahbani, M. 2019. Fair algorithms for clustering. *arXiv preprint arXiv:1901.02393*.

Chierichetti, F.; Kumar, R.; Lattanzi, S.; and Vassilvitskii, S. 2017. Fair clustering through fairlets. In *Proc. NeurIPS*, 5036–5044.

Davidson, I., and Ravi, S. S. 2007. The complexity of non-hierarchical clustering with instance and cluster level constraints. *Data Min. Knowl. Discov.* 14(1):25–61.

Davidson, I., and Ravi, S. S. 2019. Making existing clusterings fairer: Algorithms, complexity results and insights. Technical report, University of California, Davis, CA . https://web.cs.ucdavis.edu/~davidson/Publications/TR_AAAI2020.pdf.

Dheeru, D., and Karra Taniskidou, E. 2017. UCI machine learning repository.

Feldman, M.; Friedler, S. A.; Moeller, J.; Scheidegger, C.; and Venkatasubramanian, S. 2015. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, 259–268.

Friedler, S. A.; Scheidegger, C.; and Venkatasubramanian, S. 2016. On the (im)possibility of fairness. *CoRR* abs/1609.07236.

Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-completeness*. San Francisco: W. H. Freeman & Co.

Kleindessner, M.; Awasthi, P.; and Morgenstern, J. 2019. Fair k-center clustering for data summarization. *To Appear in ICML.*

Kleindessner, M.; Samadi, S.; Awasthi, P.; and Morgenstern, J. 2019. Guarantees for spectral clustering with fairness constraints. *To Appear in ICML.*

Kuo, C.-T.; Ravi, S.; Dao, T.-B.-H.; Vrain, C.; and Davidson, I. 2017. A framework for minimal clustering modification via constraint programming. In *AAAI*, 1389–1395.

Rösner, C., and Schmidt, M. 2018. Privacy preserving clustering with constraints. *arXiv preprint arXiv:1802.02497*.

Schrijver, A. 1998. *Theory of linear and integer programming*. John Wiley & Sons.

Thanh, B. L.; Ruggieri, S.; and Turini, F. 2011. k-NN as an implementation of situation testing for discrimination discovery and prevention. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, 502–510.

Vaidya, P. M. 1989. Speeding-up linear programming using fast matrix multiplication (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, 332–337.

von Luxburg, U. 2006. A tutorial on spectral clustering. Technical Report TR-149, Max Planck Institute for Biological Cybernetics, Germany.

Wagstaff, K., and Cardie, C. 2000. Clustering with instance-level constraints. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA.*, 1097–1192.

Xu, R., and Wunsch, D. C. 2005. Survey of clustering algorithms. *IEEE Transactions on Neural Networks* 16(3):645–678.

Part 1: Proofs of Results in the Main Part of the Paper

8 Statement and Proof of Lemma 3.1

Before the statement and proof of this Lemma, we introduce some terminology which is also used in the proof of Theorem 3.2 in the next section.

We note that the total number of constraints resulting from Equations (2)–(4) is $2k + n$. This is because there are two equations for each of the k clusters (Equations (2) and (3)) and there is one equation (Equation (4)) for each of the n data items. We refer to the first $2k$ equations as **Type 1** constraints and the last n equations as **Type 2** constraints. Further, among the Type 1 constraints, we will refer to the ones corresponding to upper bounds (i.e., the constraints in which slack variables $u_1, \ldots, u_k$ appear) as **positive Type 1** constraints since the coefficients $p_1, \ldots, p_n$ appear with a '$+$' sign. Likewise, among the Type 1 constraints, we will refer to the ones corresponding to lower bounds (i.e., the constraints in which slack variables $l_1, \ldots, l_k$ appear) as **negative Type 1** constraints. For each positive Type 1 constraint, note that there is a corresponding negative Type 1 constraint; such constraints are referred to as **companion pairs**.

Further, in the above equations, there are kn **regular variables** (namely, $z_{11}, z_{12}, \ldots, z_{1n}, \ldots, z_{k1}, z_{k2}, \ldots, z_{kn}$) and $2k$ **slack variables** (namely $u_1, \ldots, u_k$ and $l_1, \ldots, l_k$). For the purpose of constructing the constraint matrix C, we will use the following order of these $kn + 2k$ variables: $\langle z_{11}, z_{12}, \ldots, z_{1n}, \ldots, z_{k1}, z_{k2}, \ldots, z_{kn}, u_1, \ldots, u_k, l_1, \ldots, l_k \rangle$. Matrix C has $2k + n$ rows (one corresponding to each constraint) and $nk + 2k$ columns (one corresponding to each variable). In C, we will list the $2k$ Type 1 constraints first (in the order specified by Equations (2) and (3)) followed by the n Type 2 constraints (in the order specified by Equation (4)). Rows of C corresponding to Type 1 and Type 2 constraints will be referred to as **Type 1 rows** and **Type 2 rows** respectively. Note that each entry of C is from $\{-1, 0, +1\}$. In each row of C (which specifies one constraint), we will list the coefficients of the $kn + 2k$ variables in the order specified above. We refer to the first kn columns of C as **regular variable columns** and the last $2k$ columns as **slack variable columns**.

We are now ready to state and prove the following the lemma.

Statement of Lemma 3.1: (a) In any regular variable column of C, there are at most three non-zero elements. (b) In any slack variable column of C, there is exactly one element with value 1; the other entries in that column are 0.

Proof: (a) Consider any regular variable column and suppose it corresponds to variable z_{ij}. In Type 1 rows of that column, the variable z_{ij} appears with coefficient p_j or $-p_j$ which may both be non-zero. In Type 2 rows, variable z_{ij} appears with coefficient $+1$ in the row corresponding to data item j. Thus, there are at most three non-zero elements in the column.

(b) Each slack variable appears in exactly one Type 1 row; in that row, its coefficient is $+1$. No slack variable appears in any Type 2 row. Thus, in any column corresponding to a slack variable, there is exactly one non-zero element (namely, a $+1$). ■

9 Statement and Proof of Lemma 3.2

Statement of Lemma 3.2: Let D be a data set with one binary protected attribute x. Let $D_x \subseteq D$ denote the subset of special data items and let $N_x = |D_x|$. Let q and r be non-negative integers such that $N_x = qk + r$ with k being the number of clusters and $0 \leq r \leq k - 1$. A partition of D into k clusters is strongly fair with respect to x if and only if it has exactly r clusters each with $\lceil N_x/k \rceil$ special data items and $k - r$ clusters each with $\lfloor N_x/k \rfloor$ special data items.

Proof: The "if" part of the theorem is obvious since each cluster has either $\lceil N_x/k \rceil$ or $\lfloor N_x/k \rfloor$ special items. We now prove the "only if" part. Since $N_x = qk + r$ and $0 \leq r \leq k - 1$, we have $q = \lfloor N_x/k \rfloor$. If $r = 0$, then $\lceil N_x/k \rceil = \lfloor N_x/k \rfloor = q$. If $r \geq 1$, then $\lfloor N_x/k \rfloor = q$ and $\lceil N_x/k \rceil = q + 1$. The reader should keep these observations in mind throughout the proof.

We will first prove that in any clustering that is strongly fair with respect to x, there are exactly r clusters that have $\lceil N_x/k \rceil$ special data items. We do this by considering two cases based on the value of r.

Case 1: $r = 0$.

In this case, $N_x = qk$ and $q = \lfloor N_x/k \rfloor = N_x/k$. Thus, the strong fairness condition requires that each cluster must have exactly $q = N_x/k$ special data items. We can think of this as having exactly k clusters each with $q = \lfloor N_x/k \rfloor$ special data items and $r = 0$ clusters each with $\lceil N_x/k \rceil$ special data items.

Case 2: $r \geq 1$.

In this case, $q = \lfloor N_x/k \rfloor$ and since $r \geq 1$, $\lceil N_x/k \rceil = q + 1$. Thus, the strong fairness condition requires that each of the k clusters must have either q or $q + 1$ special data items. Let α clusters have $q + 1$ special data items. We must show that $\alpha = r$. To see this, note that there are α clusters with $q + 1$ special items and $k - \alpha$ clusters with q special items. Hence, the total number of special items in all the clusters is $\alpha(q + 1) + (k - \alpha)q = qk + \alpha$. Since the total number of special items is exactly $qk + r$, we have $qk + r = qk + \alpha$; that is, $\alpha = r$ as required. This completes the proof that exactly r clusters have $\lceil N_x/k \rceil$ special items.

It is now easy to show that the remaining $k - r$ clusters must have exactly $\lfloor N_x/k \rfloor$ special items each. This follows from the facts that exactly r clusters have $\lceil N_x/k \rceil$ special items and each cluster must have either $\lceil N_x/k \rceil$ or $\lfloor N_x/k \rfloor$ special items. This completes our proof of Lemma 3.2. ■

10 Encoding Scheme and Proof of Total Unimodularity

Encoding Scheme Used for ILP. The allocation matrix Z is encoded as nk variables and n constraints. The upper and lower bounds constraints have k variables. Table 9 shows the various ways we encode the constraints. For the counting constraints (Equations (2) and (3)) there are k constraints (one for each cluster) and n equations (one for each data item). It is important to realize all equations are essentially counting constraints then the coefficient to these will be either: i) identity matrices or ii) matrices of all zeros. For example, p_i is encoded as an $k \times k$ identity matrix if instance i has the protected status otherwise it is a $k \times k$ matrix of all zeros, thus the encoding of p_i (and $-p_i$) consists of n **vertically** concatenated matrices of size $k \times k$ since these constraints are for all clusters. This allows us to effectively count the number of protected instances in each cluster.

Constraint	**Size**	**Coefficients (Encoding Mechanism)**
$PZ \leq U$ (Eqn. 2)	$k \times kn$	n matrices size $k \times k$ horizontally concatenated. I encodes protected status.
$-PZ \geq P$ (Eqn, 3)	$k \times kn$	As above but $-I$ used instead of I.
$\sum_i z_{i,j} = 1 \ \forall j$ (Eqn. 4)	$n \times kn$	For row i only column entries $(i-1) * k + 1 \ldots (i) * k$ set to 1.
Slack U (Eqn. 2)	$k \times k$	Identity matrix
Slack L (Eqn. 3)	$k \times k$	Identity matrix

Table 9: The encoding details for the variables in Equations (2), (3) and (4).

Statement of Theorem 3.2: The matrix C formed by the coefficients of the constraints used to encode Equations (2) through (4) is totally unimodular.

Proof: In proving the result, we use the terminology regarding the constraints and variables introduced in the previous section.

As mentioned earlier, each entry of C is from $\{-1, 0, +1\}$. Consider any subset X of rows of C. We will show that X can be partitioned into two sets A and B to satisfy the condition mentioned in Theorem 3.1. Our partitioning scheme is as follows.

1. All positive Type 1 rows of X are put into A.
2. If a negative Type 1 row appears along with its companion positive Type 1 row in X, then the negative Type 1 row is also put into A.
3. If a negative Type 1 row appears in X *without* its companion positive Type 1 row, then the negative Type 1 row is put into B.
4. All Type 2 rows of X are put into B.

It is possible that the above construction causes one of A or B to be empty; in that case, the sum of the rows in that part is a row vector with all zeros.

Let row vectors $S(A)$ and $S(B)$ denote the sums of the rows in sets A and B respectively. Our goal is to show that all the elements of the row vector $Q = S(A) - S(B)$ are from $\{-1, 0, +1\}$. To prove this, consider any entry α of the vector Q. There are two main cases depending on the type of column corresponding to α.

Case 1: Entry α corresponds to a slack variable (u_i or l_i) column. In this case, from Lemma 3.1, we know that only one element in the corresponding column of C has the value 1 and the rest have the value 0. So, if the row corresponding that element appears in A, then the value of α will be $+1$; otherwise, it will be -1.

Case 2: Entry α corresponds to a regular variable z_{ij}. In this case, from Lemma 3.1, we know that in the matrix C, there are at most three non-zero elements in the column corresponding to z_{ij}. There are four subcases since the number of non-zero entries in this column corresponding to the given set of rows X can be 0, 1, 2 or 3.

Case 2.1: Among the rows in X, there are no non-zero elements in the column corresponding to α. In this case, the corresponding entries in $S(A)$ and $S(B)$ are both 0 and hence the value of α is 0.

Case 2.2: Among the rows in X, there is only one non-zero element in the column corresponding to α. In this case, the non-zero element ($+1$ or -1) appears in exactly one of $S(A)$ and $S(B)$, so the the value of α is $+1$ or -1.

Case 2.3: Among the rows in X, there are two non-zero elements in the column corresponding to the entry α. There are three possible subcases here.

Case 2.3.1: The non-zero elements are from two Type 2 rows. From Lemma 3.1, we know that in any column of a Type 2 row, there is at most one non-zero entry. So, this case cannot arise.

Case 2.3.2: The non-zero elements are from two Type 1 rows. Here, the non-zero elements must be from two companion Type 1 rows (since rows that are not companions don't have a non-zero entry in the same column). Here, our construction adds them both to A and hence the corresponding entry of $S(A)$ is 0. Since there are no other non-zero entries in that column, the corresponding entry of $S(B)$ is also 0; that is, the value of α is zero.

Case 2.3.3: Suppose the non-zero elements correspond to a Type 1 row and a Type 2 row. Note that our construction placed the Type 2 row in B. Hence the corresponding entry in $S(B)$ is $+1$. If the other row is a positive Type 1 row, it was placed in A and the corresponding entry in $S(A)$ is also $+1$, thus making the value of α to be 0. If the other row is a negative Type 1 row, it was also placed in A (since its companion is not in X), and the corresponding entry in $S(B)$ is 0, thus again making the value of α to be 0. This completes all the subcases of Case 2.3.
Case 2.4: Among the rows in X, there are three non-zero elements in the column corresponding to the entry α. In this case, X must contain two Type 1 companion rows and a Type 2 row which contains a 1 in the column corresponding to α. By our construction, the two companion rows were placed in A and the elements of those rows in the column corresponding to α are $+1$ and -1; thus, the corresponding entry of $S(A)$ is 0. Since the Type 2 row is in B and has the element 1 in the column corresponding to α, the corresponding entry of $S(B)$ is $+1$. Hence, the value of α is -1. This completes the proof of Theorem 3.2. ■

11 An Alternative Algorithm for MCMF

Idea behind the algorithm. Given an arbitrary distribution of the special items into k clusters, we use Lemma 3.2 to identify which clusters have an "excess" amount of special items. We move the excess items into a temporary container and appropriately distribute the items in the container to other clusters which are "deficient" with respect to special items. As will be shown using Lemma 3.2, the total number of excess items gives the lower bound on the number of special items that must be moved to achieve strong fairness. The algorithm provides an optimal solution by ensuring that the number of special items moved between clusters is equal to the lower bound. The identification of clusters which have excess items and those that are deficient depend on the initial distribution of the special items in the k clusters.

Notation used in the description of the algorithm: In specifying this algorithm, we assume that we need to only deal with special data items. (Data items that are not special play no role in determining fairness.) Thus, the input to the algorithm is an arbitrary partition Π of D_x have $k \geq 1$ clusters denoted by C_1, C_2, …, C_k, with cluster C_j containing β_j special items, $1 \leq j \leq k$. We also assume that the clusters are numbered 1 through k so that $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_k$. (This can be ensured in $O(k \log k)$ time by sorting the clusters.) The output of the algorithm is a partition Π' of D_x into k clusters such that Π' is strongly fair with respect to the protected attribute x. The algorithm constructs Π' by moving the *minimum* number of special items between clusters. The steps of our algorithm (which we call OPT-Modification) for the minimal modification problem are described below.

Steps of Algorithm OPT-Modification:

1. For each cluster C_j in π if $\beta_j = \lceil N_x/k \rceil$ or $\beta_j = \lfloor N_x/k \rfloor$, then **output** "$\pi$ is strongly fair" and **stop**.
2. Let $N_x = qk + r$, where $q \geq 0$ and $0 \leq r \leq k-1$. Use Case 1 or Case 2 depending upon the value of r.

Case 1: $r = 0$. Here, $N_x = qk$. (In this case, the algorithm must ensure that each cluster has exactly N_x/k special items.)
(a) Let clusters C_1, …, C_t have $> N_x/k$ special items. (Other clusters have $\leq N_x/k$ special items.)
(b) From each cluster C_j, $1 \leq j \leq t$, move $\beta_j - N_x/k$ special items into a temporary container T.
(c) For each cluster C_p such that $\beta_p < N_x/k$, move $N_x/k - \beta_p$ special items from T into C_p.

Case 2: $r > 0$. Here, $N_x = qk + r$. (In this case, as required by Lemma 3.2, the algorithm must ensure that exactly r clusters have $\lceil N_x/k \rceil$ special items and $r - k$ clusters have $\lfloor N_x/k \rfloor$ special items.)
(a) Partition the clusters into 4 groups Γ_1, Γ_2, Γ_3 and Γ_4 as follows. (Some of the groups may be empty.)
- Let Γ_1 consist of clusters C_1, …, C_t with $> \lceil N_x/k \rceil$ special items.
- Let Γ_2 consist of clusters C_{t+1}, …, C_p with exactly $\lceil N_x/k \rceil$ special items. (Thus, groups Γ_1 and Γ_2 together have p clusters.)
- Let Γ_3 consist of clusters C_{p+1}, …, C_m with exactly $\lfloor N_x/k \rfloor$ special items.
- Let Γ_4 consist of the remaining clusters, that is, C_{m+1}, …, C_k with $< \lfloor N_x/k \rfloor$ special items.

(b) Use one of Cases 2.1, 2.2 or 2.3 depending upon the comparison between p and r.
Case 2.1: $p < r$ (i.e., Groups Γ_1 and Γ_2 together have $< r$ clusters).
(i) From each cluster C_j in Γ_1, move $\beta_j - \lceil N_x/k \rceil$ special items into a temporary container T.
(ii) For each of the first $r - p$ clusters C_j in $\Gamma_3 \cup \Gamma_4$, move $\beta_j - \lceil N_x/k \rceil$ special items from T into C_j.
(iii) For each of the other clusters C_j in $\Gamma_3 \cup G_4$, move $\beta_j - \lfloor N_x/k \rfloor$ special items from T into C_j.
Case 2.2: $p = r$ (i.e., Groups Γ_1 and Γ_2 together have exactly r clusters).
(i) From each cluster C_j in Γ_1, move $\beta_j - \lceil N_x/k \rceil$ special items into a temporary container T.
(ii) For each of the clusters C_j in Γ_4, move $\beta_j - \lfloor N_x/k \rfloor$ special items from T into C_j.

Case 2.3: $p > r$ (i.e., Groups Γ_1 and Γ_2 together have $> r$ clusters). Use one of the subcases 2.3.1, 2.3.2 or 2.3.3 depending on how t compares with r.
Case 2.3.1: $t > r$ (i.e., group Γ_1 has more than r clusters).
(i) From each cluster C_j in Γ_1, move $\beta_j - \lceil N_x/k \rceil$ special items into a temporary container T.
(ii) From each cluster C_j, $r \leq j \leq p$, move $\beta_j - \lfloor N_x/k \rfloor$ special items into T.
(iii) For each cluster $C_j \in \Gamma_4$ move $\beta_j - \lfloor N_x/k \rfloor$ special items from T into C_j.
Case 2.3.2: $t = r$ (i.e., group Γ_1 has exactly r clusters).
(i) From each cluster $C_j \in \Gamma_1$, move $\beta_j - \lceil N_x/k \rceil$ special items into the temporary container T.
(ii) From each cluster $C_j \in \Gamma_2$, move $\beta_j - \lfloor N_x/k \rfloor$ = 1 special item into T.
(iii) For each cluster $C_j \in \Gamma_4$, move $\beta_j - \lfloor N_x/k \rfloor$ special items from T into C_j.
Case 2.3.3: $t < r$ (i.e., group Γ_1 has $< r$ clusters).
(i) From each cluster $C_j \in \Gamma_1$, move $\beta_j - \lceil N_x/k \rceil$ special items into the temporary container T.
(ii) From each cluster C_j, $r - t + 1 \leq j \leq p$, move $\beta_j - \lfloor N_x/k \rfloor$ = 1 special item into T.
(iii) For each cluster $C_j \in \Gamma_4$, move $\beta_j - \lfloor N_x/k \rfloor$ special items from T into C_j.

3. Output the modified partition $\Pi' = \langle C_1, C_2, \ldots, C_k \rangle$.

12 Statement and Proof of Theorem 5.1

Statement of Theorem 5.1: Problem FSFC-ML is **NP**-complete.

Proof: It is easy to see that FSFC-ML is in **NP** since one can guess a clustering C of D into k clusters and verify that it satisfies the two required conditions.

To prove **NP**-hardness, we use a reduction from the 3-PARTITION problem (Garey and Johnson 1979). An instance of the 3-PARTITION problem is specified by two positive integers m and B, a set $A = \{a_1, a_2, \ldots, a_{3m}\}$ of positive integers such that $B/4 < a_i < B/2$, $1 \leq i \leq 3m$ and $\sum_{i=1}^{3m} a_i = mB$. The question is whether A can be partitioned into m subsets such that the sum of each subset is equal to B. It is known that 3-PARTITION is strongly **NP**-complete; that is, the number of bits needed to represent B and each integer in a_i are polynomial functions of $\log m$, the number of bits needed to represent m. Also note that the condition $B/4 < a_i < B/2$ implies that whenever there is a solution to a 3-PARTITION instance, each of the m subsets in the partition has exactly three integers from A.

The reduction from 3-PARTITION to FSFC-ML is as follows.
(a) For each integer $a_i \in A$, we create a set S_i containing a_i data items, $1 \leq i \leq m$. For each pair of data items p and q in S_i, we create the ML constraint ML(p, q), $1 \leq i \leq m$. (These constraints ensure that in any feasible solution to the FSFC-ML instance, all the data items in S_i must be in the same cluster.)
(b) The data set D for the FSFC-ML instance is given by $D = \cup_{i=1}^{m} S_i$. There is one protected attribute x and for each data item in D, the value of the protected attribute is 1. Thus, the number of special data items is mB.
(c) The number of clusters is set to m.

Using the fact that the numbers of bits needed to represent B and each integer in a_i ($1 \leq i \leq m$) are polynomial functions of $\log_2 m$ (the number of bits needed to represent m), it can be seen that the above construction can be carried out in polynomial time. We now show that there is a solution to the FSFC-ML instance if and only if there is a solution to the 3-PARTITION instance.

Suppose there is a solution to the 3-PARTITION instance. Let X_1, X_2, …, X_m denote the partition of the set A into m subsets. As mentioned earlier, each subset X_j has exactly three integers from A, $1 \leq j \leq m$. Let block X_j contain integers a_{j_1}, a_{j_2} and a_{j_3}. Then for $1 \leq j \leq m$, cluster C_j in the solution to the FSFC-ML instance consists of the data items $S_{j_1} \cup S_{j_2} \cup S_{j_3}$. Clearly, this satisfies all the ML constraints since for each set S_i ($1 \leq i \leq 3m$), all the elements of S_i are in the same cluster. To show that the resulting clustering is strongly fair with respect to the protected attribute x, we start by noting that the number of special data items is mB. Since there are m clusters, we have $\lfloor mB/m \rfloor$ = $\lceil mB/m \rceil$ = B. Thus, the strong fairness condition requires that each cluster should have exactly B data items that are special. Since the sum of the integers in each subset X_j is B, each cluster contains exactly B data items. Since each data item is special, it follows that each cluster has exactly B data items that are special. In other words, the clustering is strongly fair with respect to x and satisfies all the ML constraints. Thus, we have a solution to the FSFC-ML instance.

For the converse, assume that we have a solution to the FSFC-ML instance. Let the m clusters be denotes by C_1, C_2, …, C_m. As argued above, the strong fairness condition requires that each cluster must have exactly B data items that are special. Since each data item in D is special, it follows that each cluster has exactly B data items. Further, the ML constraints require that for each S_i, $1 \leq i \leq 3m$, all the data items in S_i must be in the same cluster. Since $B/4 < |S_i| < B/2$ and each cluster has exactly B data items, it follows that each cluster has all the data items from *exactly three* of the data sets from S_1, S_2, …, S_{3m}. Suppose cluster C_j contain the sets S_{j_1}, S_{j_2} and S_{j_3}, $1 \leq j \leq m$. From the cluster C_j, we construct the subset A_j consisting of the integers a_{j_1}, a_{j_2} and a_{j_3} that correspond to the three sets S_{j_1}, S_{j_2} and S_{j_3}. Since each cluster has exactly B data items, it follows that the sum of the three integers in A_j is exactly B, $1 \leq j \leq m$. In other words, we have a solution to the 3-PARTITION instance, and this completes our proof of Theorem 5.1. ■