Data Mining and Knowledge Discovery (2023) 37:1404–1440
https://doi.org/10.1007/s10618-022-00893-6

Making clusterings fairer by post-processing: algorithms, complexity results and experiments

Ian Davidson[1] · Zilong Bai[1] · Cindy Mylinh Tran[1] · S. S. Ravi[2,3]

Received: 15 September 2021 / Accepted: 25 October 2022 / Published online: 17 December 2022

Abstract

While existing fairness work typically focuses on fair-by-design algorithms, here we consider making a fairness-unaware algorithm's output fairer. Specifically, we explore the area of fairness in clustering by modifying clusterings produced by 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 but fairer. We show that for a single binary protected status variable, the problem is efficiently solvable (i.e., in the class **P**) by proving that the constraint matrix for an integer linear programming formulation is totally unimodular. Interestingly, we show that even for a single protected variable, the addition of simple pairwise guidance for clustering (to say ensure individual-level fairness) makes the MCMF problem computationally intractable (i.e., **NP**-hard). Experimental results using 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 clusterings that

Responsible editor: Toon Calders, Salvatore Ruggieri, Bodo Rosenhahn, Mykola Pechenizkiy and Eirini Ntoutsi.

A preliminary version of this paper appeared in AAAI-2020 (Davidson and Ravi 2020).

✉ Ian Davidson
indavidson@ucdavis.edu

Zilong Bai
zlbai@ucdavis.edu

Cindy Mylinh Tran
cmltran@cs.ucdavis.edu

S. S. Ravi
ssravi0@gmail.com

[1] Computer Science Department, University of California Davis, Davis, CA 95616, USA

[2] Biocomplexity Institute and Initiative, University of Virginia, Charlottesville, VA 22904, USA

[3] Department of Computer Science, University at Albany – SUNY, Albany, NY 12222, USA

are as fair but are of higher quality than those produced by fair-by-design clustering algorithms. Finally, we explore a challenging practical problem of making a historical clustering (i.e., zipcodes clustered into California's congressional districts) fairer using a new multi-faceted benchmark data set.

Keywords Clustering · Protected status · Fairness · Algorithms · Complexity

1 Introduction

As machines help and even replace humans in decision making processes on other humans, the need to ensure that the algorithms used by these machines are fair becomes critical. Consider a future scenario where job, loan and even grant applications are automatically adjudicated by machines. When a person's application is denied, their natural questions are "Was it fair? Did other people as good as me also get denied?". This is an example of individual level fairness (Dwork et al. 2012) where fairness is measured between a pair of individuals. As another example, policy makers wish to ensure that under-represented groups are not being discriminated against. To achieve this, they would like to ensure that the fraction of successful applications for each under-represented group is above some threshold (such as the fraction of the group in the population). This is an example of group level fairness where we measure fairness on a group of individuals (Chierichetti et al. 2017). For a more detailed discussion on fairness issues in machine learning, we refer the reader to Barocas et al. (2017). In this paper, our focus is on fairness in clustering.

Motivation. Existing work on clustering and fairness takes a known clustering *algorithm* and modifies it into a fair-by-design algorithm that produces 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). Subsequently, many other researchers have studied fair-by-design clustering algorithms for different types of clustering and fairness criteria (e.g., Abbasi et al. 2021; Ziko et al. 2021; Flores 2019; Chhabra et al. 2021). However, there is a plethora of different clustering algorithms, with a 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. Further, the no free lunch theorems for learning and optimization (Schaffer 1994; Wolpert and Macready 1997) in machine learning indicate a need for a variety of algorithms. It is unlikely that fair versions of all these algorithms or future clustering algorithms will be developed. Finally, historical clusterings (such as electoral maps) already deployed and often created by humans, permeate the society.

In these circumstances, one research direction is to modify an *existing clustering* (be it from a machine or historical) to make it fairer whilst not unduly changing its quality. For example, congressional districts can be considered a historical clustering of zipcode areas which may be unfair and can be minimally modified to make them fairer. Similarly, a clustering produced by a popular algorithm for which there is no fair variant can be modified to become fairer. 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 status variables (PSVs). We focus on the simplest but most common type of protected variables, namely binary variables such as gender.

A Flexible and Efficient Approach to Ensure Fairness. Our approach places upper and lower bounds on the number of protected status individuals in each cluster. These bounds, which can be computed from the input, may even be different for each cluster. This allows us to encode group-level fairness. Of the three classic definitions of group-level fairness (Barocas et al. 2017), namely independence, separation and sufficiency, only the first is appropriate for unsupervised learning. For a PSV A, and a given set of clusters $C_1, C_2, \ldots, C_k$, the classic definition of independence is simply that for $1 \leq i \leq k$, $P(A \mid C_i) = P(A)$; the notion of disparate impact (Barocas and Selbst 2016; Feldman et al. 2015) softens this requirement to $(1-\alpha)P(A) \leq P(A \mid C_i) \leq (1+\alpha)P(A)$, $1 \leq i \leq k$, where α is typically chosen as 0.2.

Consider a data set D where there is one protected status variable x. The data items in D for which the variable x has value 1 will be referred to as **special** items. Suppose D, with N_x special items, has been partitioned into k clusters, denoted by $C_1, C_2, \ldots, C_k$. Then the number of special data items per cluster could be set to be approximately N_x/k, to effectively balance the protected status instances uniformly across all clusters. This gives rise to a strong definition of fairness:

Definition 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 $k \geq 2$ clusters is **strongly fair with respect to** x if in each cluster, the number of special items is either $\lfloor N_x/k \rfloor$ or $\lceil N_x/k \rceil$.

The above is useful in our intractability results and algorithm design. However, it is a strong requirement; hence we define a relaxed notion called α-fairness below. Here, we require each cluster to have approximately $\frac{N_x|C_i|}{|D|}$ special items, where $|C_i|$ is the size of cluster C_i; 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 and $P(x) = \frac{N_x}{|D|}$.

Definition 2 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. Let $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ be a vector such that α_i is a positive integer $< \lceil \frac{N_x|C_i|}{|D|} \rceil$, $1 \leq i \leq k$. A partition of D into $k \geq 2$ clusters is α-**fair with respect to** x, if in cluster C_i, the number of special items is in the range $\left[\lceil \frac{N_x|C_i|}{|D|} \rceil - \alpha_i \,..\, \lceil \frac{N_x|C_i|}{|D|} \rceil + \alpha_i \right]$, $1 \leq i \leq k$.

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 (expressed as an integer linear program (ILP)) where the goal is to minimize the effect of modifying the clusters (see Sect. 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. Solving an ILP is, in general, computationally intractable (Garey and Johnson 1979). However, for our formulation of the MCMF problem, we show that (see Theorem 2) the constraint matrix resulting from our fairness requirements is *totally unimodular* (TU) (Schrijver 1998). As a consequence, one can use

any polynomial time linear programming (LP) algorithm to obtain integer solutions to MCMF.

Summary of Main contributions.

1. Rather than taking an existing and create a fair variant, we define a novel optimization problem (referred to as MCMF) which can post-process the results of **any** partitional clustering method to make the clusters fairer whilst ensuring that the cluster quality is not affected significantly.
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 Sect. 3). This leads to algorithms capable of modifying clusterings with millions of instances.
3. For modifying clusters of even larger data sets, we present an algorithm whose running time is better than that of an LP solver (see Sect. 3).
4. Our complexity result for the MCMF problem shows an interesting conundrum. Finding a group level fair clustering is in the computational class **P** as is finding an individual level fair clustering (using must-link constraints). However, finding a group level and individual level fair clustering is intractable (see Sect. 5, Theorem 3).
5. Experimental results show that our method is computationally efficient (as expected) and that our objectives are useful for post-processing the results from k-means, k-medians and spectral clustering algorithms.
6. Finally, we explore a challenging case study of making a historical clustering (i.e., zipcodes clustered into California's congressional districts) fairer using a new multi-faceted benchmark data set. This involved defining new spatial continuity constraints and testing our method on a significantly challenging problem. This data set will be publicly made available for others to study.

Organization. We begin by discussing related work in Sect. 2. We then formulate the minimal clustering modification for fairness (MCMF) problem in Sect. 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 Sect. 5. We then present results from our experiments on classical data sets in Sect. 6. Results of our case study on the congressional district data from California are presented in Sect. 7. Conclusions and directions for future work are provided in Sect. 8. Three sections of appendices are included. "Appendix A" provides a short introduction to ILPs and also presents an example of a constraint matrix arising from our ILP formulation for the MCMF problem. "Appendix B" contains a more efficient algorithm for the MCMF problem where the goal is to ensure strong fairness. "Appendix C" contains additional information about the congressional district data from California and the experimental settings used in our work for that data set.

An earlier version of this paper was published in the proceedings of AAAI-2020 (Davidson and Ravi 2020). We have added the following to the conference version.

– The AAAI-2020 version does not include proofs for any of the formal results. The journal version includes all the proofs.

– We point out a simple relationship between our notion of strict fairness (for post-processing for fairness) versus the notion of balance [used in earlier work on fairness by design (Chierichetti et al. 2017)].
– In The AAAI-2020 version, we only briefly mention that for the strict fairness case, there is an algorithm that is more efficient than solving the LP. This version provides the details of the algorithm.
– We go into greater detail regarding the counter-intuitive result that satisfying individual level fairness is tractable as is group-level fairness, but satisfying both makes the problem computationally intractable.
– Sect. 7 which adds a new fairness data set for clustering and discusses our results for CA congressional districts is completely new. It considers multiple PSVs while the AAAI-2020 version only considered a single PSV.

2 Related work and background

2.1 Previous work on fairness and minimal modification of clustering

We briefly review two related areas of work, 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 et al. 2011). More recently, in clustering (i.e., unsupervised learning), the issue of fairness generally aims at balancing protected status individuals across clusters as we have.

All of the work below uses a similar fairness measure (discussed 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 et al. 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). If k denotes the number of clusters and the number of instances of each type in cluster i are R_i and B_i respectively, the fairness of a clustering is given by

$$\min\left\{\min\left[\frac{R_1}{B_1},\frac{B_1}{R_1}\right],\ldots,\min\left[\frac{R_k}{B_k},\frac{B_k}{R_k}\right]\right\}.$$

The work of Chierichetti et al. (2017) refers to the above quantity as the **balance** of the given clustering and the quantity $\min\left[\frac{R_j}{B_j},\frac{B_j}{R_j}\right]$ as the balance of cluster C_j. Their work creates fairlets (groups of instances) which when post-processed by k-center and k-medians algorithms are guaranteed to produce a specified level of fairness (balance) and achieve a constant factor approximation with respect to cluster quality. In Sect. 2.2, we present a simple observation that relates the balance measure and our measure of strict fairness when all clusters have nearly the same size. As will be seen in Sect. 6 (Table 4), by ensuring that cluster sizes are not unduly changed, the balance measure

of Chierichetti et al. (2017) and our fairness measure can yield similar results in experiments with data sets.

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 but assumes 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. Other references that discuss fair-by-design non-hierarchical clustering algorithms include (Flores 2019; Chen et al. 2019; Ahmadian et al. 2020; Ahmadi et al. 2020; Mahabadi and Vakilian 2020). Incorporating fairness into an algorithm for agglomerative hierarchical clustering is studied in Chhabra and Mohapatra (2020).

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, the focus there is on human-in-the-loop style settings where the domain expert can choose to adjust geometric measures of a cluster such as diameter. Hence, the results in the current paper and those in Kuo et al. (2017) 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 around 1000. In contrast, our modification algorithm for improving fairness scales to larger data sets (hundreds of thousands of points), even on a laptop (see Sect. 6).

2.2 An observation relating our fairness measure to balance

We presented the definition of balance for a cluster and a given clustering used by Chierichetti et al. (2017) in the previous section. Here, we present a simple observation that relates our strict measure of fairness and their notion of balance when all the clusters are of equal (or nearly equal) sizes. One of the results mentioned in Chierichetti et al. (2017) is the following.

Observation 1 *Let X be a set of points where each point is colored red or blue. For any partition Π of X into $k \geq 1$ clusters,* $\text{balance}(\Pi) \leq \text{balance}(X)$.

To see a relationship between the balance measure and our strict notion of fairness, consider the following example with a single protected status variable denoted by x. Suppose a data set D has a total of kN_x objects where N_x are special; that is, N_x objects have protected attribute value of 1. The parameter $k \geq 2$ here is the number of clusters into which D will be partitioned. For convenience, we assume that N_x is a multiple of k. We think of the special objects in D as Red objects and non-special objects as Blue objects. Since the number of Red objects in D is N_x, the number of blue objects in $D = (k-1)N_x$. Thus, there are at least as many blue objects as red objects. Using the definition of balance, we have

$$\text{balance}(D) = N_x/[(k-1)N_x] = 1/(k-1).$$

By Observation 1, no clustering of D can have a balance value greater than $1/(k-1)$.

Suppose D is partitioned into k clusters of equal size; thus, the size of each cluster is N_x. By our definition of strict fairness, each cluster has N_x/k red objects and $N_x - (N_x/k) = N_x(1-1/k)$ blue objects. Therefore, the balance of any cluster C_i in this clustering is given by

$$\text{balance}(C_i) \;=\; \frac{(N_x/k)}{(N_x - N_x/k)} \;=\; 1/(k-1)$$

which is exactly the best possible balance value possible for D.

When clusters are of different sizes, we can argue that for clusters whose size is close to the average size of a cluster, the balance value is close to the best possible balance. Since $|D| = kN_x$ and the number of clusters is k, the average size of a cluster is N_x. Suppose we have a cluster C_i whose size is αN_x for some $\alpha \approx 1$. (Thus, $|C_i|$ is close to the average cluster size.) Since C_i has N_x/k red items and $|C_i| - N_x/k$ blue items, we have

$$\text{balance}(C_i) \;=\; \frac{(N_x/k)}{(|C_i| - N_x/k)} \;=\; \frac{(N_x/k)}{(\alpha N_x - N_x/k)} \;=\; 1/(\alpha k - 1).$$

As α approaches 1, the above balance value gets close to the best possible balance value of $1/(k-1)$. We thus have:

Proposition 1 *When clusters are of equal (or nearly equal) size and special objects are taken as the Red items, distributing the special items equally among the clusters (i.e., our notion of fairness) gives the best (or close to the best) possible balance.*

3 An ILP formulation and proof of total unimodularity

Here we present an ILP formulation[1] of the MCMF problem which can find fairer clusters. We then show that the resulting 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 special instances (i.e., number of instances whose PSV value is 1) in a cluster. These bounds need not be the same for each cluster and most importantly, since the TU property 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 a $k \times n$ allocation matrix Z^*, where k is the number of clusters and n is the number of instances), find a minimal modification that makes the clustering fairer with respect to the single protected variable P. In our experiments, we consider a formulation that allows many PSVs. Later in this paper, we discuss other settings that can lead to TU constraint matrices.

[1] For the convenience of readers, a short introduction to ILP formulations is provided in Sect. A.1 of the appendix.

Table 1 Several penalty schemes and their meaning when used in Eq. (1)

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

Each measures the increase if instance j is moved away from the cluster (C^*) to which it is assigned in Z^*. We assume that Z^* is optimal for the given objective

Objective. We wish to find another allocation matrix Z that is fairer but similar to Z^*. As z_j^* and z_j are both binary **column** vectors indicating the cluster to which instance j belongs, $\sum_j (z_j^*)^T \times (z_j)$ counts the number of agreements between Z and Z^*; this forms the basis for useful objectives. We can easily encode preferences/importance amongst instances by having a penalty (w_j) (see Eq. (1)) if instance j is moved. This penalty value can represent different objectives, and some examples are shown in Table 1. The first example 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. The minimization objective of our formulation is given by

$$argmin_Z \sum_j w_j[1 - (z_j^*)^T \times (z_j)], \tag{1}$$

where Z is the collection of all valid allocation matrices (over which the objective is to be minimized), w_j is the weight of moving instance j to a different cluster, z_j^* is the initial allocation vector for instance j and z_j (to be found) is the allocation vector for instance j in the modified clustering, $1 \leq j \leq n$.

Adding Constraints With Slack Variables. The aim of the constraints is two-fold: to balance the protected instances across clusters 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 encode protected status as a vector P of length n with an entry of 1 if the instance has the status and 0 otherwise. We use $|P|$ to denote the number of non-zero entries in the vector P. (Thus, $|P|$ is the number of special items in the dataset.) Our first two constraints (in the formulation given below) require that the distribution of the protected variable be upper and lower bounded. For example, to follow our definition of strong fairness (see Definition 1), we would have the constraint $\lfloor \frac{|P|}{k} \rfloor \leq \sum_j p_j z_{i,j} \leq \lceil \frac{|P|}{k} \rceil \quad \forall i$. In solving ILPs, such inequality constraints are changed to equality constraints using *slack* variables (Schrijver 1998); we use u_i and l_i respectively as the slack variables in

the upper and lower bound constraints for $\sum_j p_j z_{i,j}$. In the following specification of constraints, we generalize this to any upper and lower bounds and note they can vary depending on the cluster. 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 and that u_i and l_i represent the slack variables in the following constraints.

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

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

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

Note that when $U_i = \lceil |P|/k \rceil$ and $L_i = \lfloor |P|/k \rfloor$, for $1 \leq i \leq k$, Lemma 2 (presented later in this section) points out that **there is always** a solution to the above set of constraints.

An example to illustrate the resulting set of constraints and the constraint matrix is given in "Appendix A.2".

Total Unimodularity of Constraint Matrix. It is well known (Schrijver 1998) that if the *constraint matrix* of an ILP is totally unimodular (TU), we can solve the ILP 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.

We first introduce some terminology to present a proof of the TU property of the constraint matrix resulting from Eqs. (2)–(4). We note that the total number of constraints is $2k+n$. This is because there are two equations for each of the k clusters [Eqs. (2) and (3)] and there is one equation (Eq. (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**.

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 ⌋, 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 ⌋ has $2k+n$ rows (one corresponding to each constraint) and $nk+2k$ columns (one corresponding to each variable). In ⌋, we will list the $2k$ constraints

corresponding to Eqs. (2) and (3) in the order specified by those equations. This is followed by the n constraints in the order specified by Eq. (4). Note that each entry of $\rfloor$ is from $\{-1, 0, +1\}$. In each row of $\rfloor$ (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 $\rfloor$ as **regular variable columns** and the last $2k$ columns as **slack variable columns**. Using this terminology, we can prove the following lemma.

Lemma 1 *(a) In any regular variable column of $\rfloor$, there are at most three non-zero elements. (b) In any slack variable column of $\rfloor$, 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, $+1$).

□

To prove the TU property of the constraint matrix $\rfloor$, we will use the following result, which is Theorem 19.3 in Schrijver (1998). In the literature, this result is also referred to as the Ghouila-Houri characterization (Berge 1972).

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

Proof In proving the result, we use the terminology regarding the constraints and variables introduced above.

As mentioned earlier, each entry of $\rfloor$ is from $\{-1, 0, +1\}$. Consider any subset X of rows of $\rfloor$. We will show that X can be partitioned into two sets A and B to satisfy the condition mentioned in Theorem 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 1, we know that only one element in the corresponding column of $\rfloor$ 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 $+1$; if the row does not appear in A, the value of α will be -1. If the row does not correspond to the chosen slack variable, the value of α is 0.

Case 2: Entry α corresponds to a regular variable z_{ij}. In this case, from Lemma 1, we know that in the matrix $\rfloor$, 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 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 B (since its companion is not in X). Thus, the corresponding entries in both $S(B)$ and $S(B)$ are zero, thus 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 2. □

Our formulation of MCMF considered simple optimization objectives (see Table 1) that use a single weight (penalty) to each instance regardless of the cluster to which the instance is moved. One can also consider formulations where there is a vector of penalty values associated with an instance so that the penalty is different for different clusters. As long as the resulting objective function is linear, the TU property of the constraint matrix implies that the corresponding optimization problem is efficiently solvable.

The following result presents a necessary and sufficient condition for the distribution of the special items across clusters in any strongly fair clustering of D. This condition can be used to develop an alternative efficient algorithm for modifying a given clustering to achieve strict fairness.

Lemma 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 The "if" part of the lemma 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 2. □

An Alternative More Efficient Algorithm Only for Strong Fairness. It is 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 2 (above) 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 2 that 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 and its running time analysis are given in Sect. B of the appendix. From the running time of the LP-based algorithm given in Sect. 3 (in the paragraph entitled "Total Unimodularity of Constraint Matrix"), one can see that this algorithm's worst-case running time, namely $O(n + k \log k)$, is asymptotically **better** than that of an LP solver. However, this algorithm is only for the strong fairness condition while the LP-based algorithm can handle more general form of fairness mentioned in Sect. 1.

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 (say, $s \geq 2$) clusters. Our slack variable formulations easily facilitates an instance belongs to s clusters. This has the benefit of spreading the protected individuals to multiple clusters. The corresponding constraint is $\sum_i z_{i,j} = s, \ \forall j$. This does **not** change the coefficients of the constraint matrix; it only changes the constant on the right side of the equality. Hence this formulation also has the TU property.

Multiple Protected Variables. When there are $r \geq 2$ protected variables, $r - 1$ additional sets of constraints similar to Eqs. (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 (e.g., age) even if encoded in ordinal form cannot be encoded as TU matrices to our knowledge.

5 Computational intractability of satisfying group- and individual-level fairness

Here we show an interesting but counter-intuitive result. We showed earlier that satisfying group level fairness is in the computational class **P** (from Lemma 2). It is known that satisfying instance level fairness, assuming that pairwise guidance is given in the form of must-link constraints, is also in the computational class **P** (Davidson and Ravi 2007). However, as we will show, when attempting to satisfy both types of fairness, the problem becomes computationally intractable. We formalize these claims below, but an intuitive explanation is as follows. Suppose we allocate people to satisfy individual level fairness, that is, everyone is assigned to the cluster to which they belong. However, there may be too few (or too many) males in one cluster to satisfy group level fairness. Then we may be required to move a male from one cluster to another violating their individual fairness. To fix this, we must move another person to this new cluster and in doing so violate their individual fairness or perhaps group level fairness for the cluster. The critical point is that these two measures of fairness are not necessarily compatible with each other: satisfying one may violate the other.

Our measure of strong fairness (Definition 1) is a group (cluster) level measure (Barocas and Selbst 2016). An alternative measure is individual-level fairness (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 et al. 2008) where the constraint ML(a, b) requires data items a and b to be in the same cluster.

As shown in Lemma 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 3). 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 3 *Problem FSFC-ML is **NP**-complete.*

Proof It is easy to see that FSFC-ML is in **NP** since one can guess a clustering $\rfloor$ 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 $\log_2 m$, 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.

1. 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.)
2. 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.
3. 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 $X_1, X_2, \ldots, X_m$ 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 the subset 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 denoted by $C_1, C_2, \ldots, C_m$. 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 $S_1, S_2, \ldots, S_{3m}$. 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 3. □

A consequence of Theorem 3 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 on classic data sets

To illustrate the usefulness of our method, we consider several classic data sets (Adult/Census, Twitter Healthcare and NYT) using 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 versus Them" tables of results; instead, we attempt to illustrate our work's uses, limitations and comparisons to fair-by-design clustering algorithms. We attempt to answer the following questions:
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 a 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 then move onto two larger collections 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% and females: 33.2%) from the UCI repository (Dua and Graff 2017).

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. The clustering was done using the attributes Age, Education, Status, Occupation and Gain. 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. 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 the 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 appropriately 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 in the distortion (the objective function used by k-means).
The results show several interesting insights:

Table 2 For k-means and Census dataset

Cluster	1	2	3	4	5
Female	21%	12%	25%	51%	14%
Size	5352	2776	15180	20182	5352
Age	42	47	43	31	46
Education	Bachelors	Bachelors	HS-grad	Some-college	Some-college
Status	Married	Married	Married	Never	Married
Occupation	Prof	Sales	Craft	Prof	Exec
Gain	3910	2887	353	233	2556

Clustering was done using the attributes Age, Education, Status, Occupation and Gain. 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 minimally modify to obtain results in Tables 3 and 4

Table 3 For k-means, Census dataset and our method

Cluster	1	2	3	4	5
Female	45%	45%	34%	52%	10%
Size	5923	**6321**	**12366**	**10231**	**14001**
Age	41	43	44	35	45
Education	Bachelors	**HS-grad**	HS-grad	Some-college	**Bachelors**
Status	Married	**Never**	Married	Never	Married
Occupation	Prof	Sales	Craft	Prof	**Sales**
Gain	2834	2532	1431	452	2641

Clustering was done using the attributes Age, Education, Status, Occupation and Gain. A description of the clusters found using our method (using second objective in Table 1) by minimally modifying the clustering described in Table 2. The distortion of this solution increased approximately 2% to 112400.68. Compare with Table 2. Interesting changes between this table and Table 2 are in boldface

1. We find that when only modifying to make clusters fairer, the distortion only increased by 2%. This indicates that our objective function in Table 1 is useful in ensuring that 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 points out a challenge with balancing just one 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 sizes of the clusters by creating a *dummy protected variable* that every instance possesses. Thus, to better ensure fairness (with respect to gender) across the clusters whilst retaining other properties of the two desirable clusters, we also constrain Education and Marital-status attributes.

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 (Sect. 6.2) 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/ethnicity (black, hispanic, white) and age (elderly, young). The former data set is already processed whilst we processed the latter using the BOW toolkit.[2] 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 report the increase in distortion and cut cost by ensuring fairness across all clusters for a variety of keywords 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, created a spectral embedding into a 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 the sum of the increases 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) is similar to our own as they are both cardinality based (see Sect. 2). However, this is different from the following question: does post-processing to increase fairness find the same, better or worse clusterings compared to finding a fair clustering to begin with? To explore this question, we used a scalable version of the fair k-median algorithm in Chierichetti et al. (2017) as discussed in the work of Backurs et al. (2019). The work on fair spectral clustering (Kleindessner et al. 2019) could be a suitable comparison but for our data sets of 48k, 58k and 300k instances, it was not scalable as the resulting affinity matrices were nearly 300Gb large (i.e., to encode a 300k x 300k matrix of short integers).

[2] https://www.cs.cmu.edu/~mccallum/bow/.

Table 4 Census dataset, k-means and our method with multiple constraints on variables

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 ±0.05%	15.4%	0.1%
Education + Keep Cluster Sizes ±0.05%	19.8%	0.8%
Marital Status + Keep Cluster Sizes ±0.05%	20.3%	0.9%
Education + Marital Status + Keep Cluster Sizes ±0.05%	23.6%	0.8%

The distortion increase of the modified to be fairer (with respect to gender) clustering over the clustering in Table 2 is shown under the requirement that other properties in Table 2 must be retained

Table 5 Results of applying k-means, spectral clustering and our method for NYT data set

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 ± 5%	18.1%	20.3%

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 is shown. Each row shows the increase in distortion and cut-cost caused by a fairness requirement

We performed two experiments. Firstly, we ran both methods (k-medians[3] 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 (RI) between the clusterings found by the two methods. If the Rand Index is 1, then 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 of Table 7).

We next explored how the outputs of the two methods are different. To achieve this, we plot the results of the Census data experiments in a 2D scatter plot where one

[3] Theory and applied literatures 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.

Table 6 Results from k-means and spectral clustering and our method for Twitter Healthcare Data Set

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 $\pm$ 5%	17.3%	19.4%

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 is shown. Each row shows the increase in distortion and cut-cost caused by adding a fairness requirement

Table 7 Comparison of post-processing for fairness versus searching for fair clusterings for 350 bootstrap samples each of the Census, NYT and Twitter Healthcare data sets

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%)

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. In the last column, the numbers in parentheses show the number of instances moved as a percentage of the cluster size

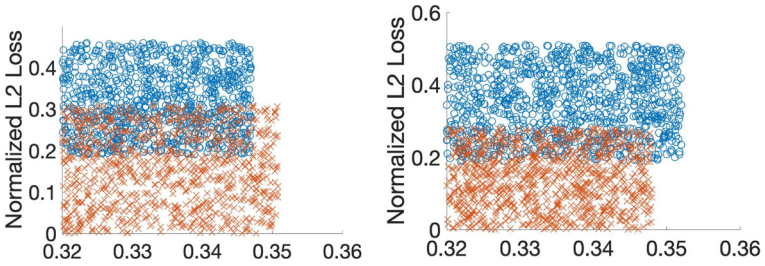


Fig. 1 For 350 bootstraps of the Census data set, the comparison of our method (crosses) versus Chierichetti et al. (2017) (circles) of the k-medians loss versus our measure of fairness (left) and Chierichetti et al. measure (right)

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 Fig. 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.

Table 8 Scalability results of our algorithm using NYT (left) and Twitter Health care (right) data sets

k	Run-time	n	Run-time
2	3.8 s/4.3 s	1000	0.20 s/0.29 s
4	6.1 s/6.5 s	2000	0.81 s/0.95 s
8	8.3 s/9.1 s	4000	1.11 s/1.45 s
16	27.30 s/23.1 s	8000	3.23 s/4.93 s
32	75.43 s/85.1 s	16000	16.41 s/18.55 s
		32000	69.32 s/73.81 s

The min/max 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:** Data set with 10,000 instances and varying numbers of clusters. **Right:** 5 clusters and varying data set sizes

6.3 Q3—Scalability

Theorem 1 shows that our ILP formulation has the TU property and hence can be solved by an LP solver. However, as indicated in Sect. 3, this can still take $O(n(n+d)^{1.5}L)$ time in the worst-case, where n is the number of variables, d is the number of constraints and L is the number of bits needed to specify all the constants specified in the ILP (Vaidya 1989). In our experiments, we found that instances of the `Adult` data set took under one minute to run on a single core of a MacBook laptop. For the larger NYT data set, the time was under 5 min and for the Twitter Healthcare data set, it was about 4 min. Here we wish to see how the execution time of our algorithm is affected by increasing the number of clusters and number of instances. In Table 8, we show the execution time on a laptop for the NYT and Twitter Healthcare data sets using sample of various sizes. We averaged results over 100 experiments with 25 experiments each balancing Female, Black, Elderly and Hispanic attributes to match the population ratios.

7 Case study: experiments on real world congressional districts data

Prior work on fairness in clustering has focused on classic academic data sets that we experimentally compared against in the previous section. In this section, to test the usefulness of our work, we explore the fairness of California's congressional districts with respect to a number of PSVs (e.g., socioeconomic, housing, and demographic information) collected from the Census Bureau's 2018 American Community Survey Bureau (2020a). This data set will be made publicly available and will serve as a good test bed application for fair clustering.

7.1 Data description and experimental goals

California is divided into 53 congressional districts (CDs) which are redrawn every decade (CNMP 2020; NCSL 2019). Each CD consists of a subset of 1769 ZCTAs (zip code tabulation areas) shown in Fig.[4] 7b (Bureau 2020b). We can view the CDs

[4] This figure appears in "Appendix C".

Table 9 Some of the 63 PSVs from our curated dataset and the number of CDs that violate disparate impact with respect to that PSV (see Definition 3)

Protected Status Variable (PSV)	#CDs ViolatingDisparate Impact
Population 16 years and over in labor force	10
Population 16 years and over NOT in labor force	9
Private wage and salary worker	17
Government worker	18
Self-employed in own not incorporated business worker	29
Male citizen over 18	14
10–14 yr old	12
Race: Vietnamese	47
Total Household Income Over 100k	36
Age: 55+	20

as the clusters and the ZCTAs as the instances. Typically, each of the 1769 distinct ZCTAs in California is assigned exclusively to *one* CD, but some ZCTAs on the boundaries of CDs can be shared by multiple congressional districts. In our experiments, if a ZCTA is shared by multiple congressional districts, its population is distributed equally amongst those CDs to create multiple ZCTA-Instances. See "Appendix C.1" for details regarding the most recent congressional districts. The current major focus of the selection committee that creates the congressional districts is to ensure relatively equal sizes of populations across the 53 CDs (Ballotpedia 2020). However, this causes unfairness in terms of *disparate impact* (Feldman et al. 2015) for some protected status variables (PSVs). Table 9 shows several PSVs for which unfairness occurs due to disparate impact. To analyze fairness issues in the context of congressional districts, we use the following version of the standard definition of disparate impact.

Definition 3 (Disparate Impact in Congressional Districts) The notion of disparate impact indicates unfairness but not by design [U.S. Equal Employment Opportunity Commission 2007 (Commission 2007)]. Disparate impact in the congressional districts (CDs) setting occurs if a CD's population of protected status (PS) individuals is outside the bounds of $\pm 20\%$ of the PS individuals' occurrence in the population. This codifies the classic legal precedent known as the "80% rule" that if the selection rate of protected status individuals is less than 80% of that for individuals of unprotected status, disparate impact occurs.

In the rest of this section, we attempt to redistrict CDs by modifying the ZCTAs assigned to them. This modification redistributes the population to address unfairness due to disparate impact. Specifically, our experiments explore the following questions.

- **Q1**. Can we minimally modify (i.e., redistrict) the ZCTAs of the 53 CDs in California to make them fairer with respect to a single PSV? Further, how do the shapes of the CDs change?
- **Q2**. When improving fairness with respect to *different* PSVs, does our method redistrict the *same* ZCTAs to make the CDs fairer?

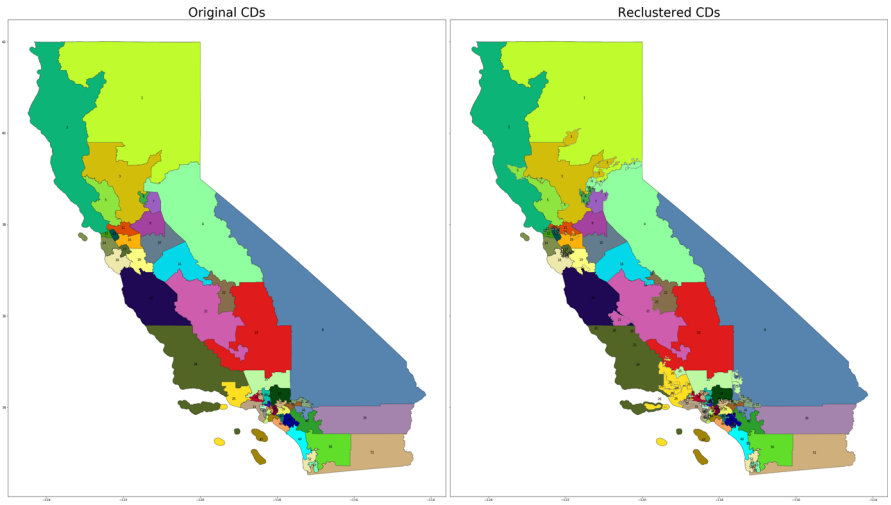


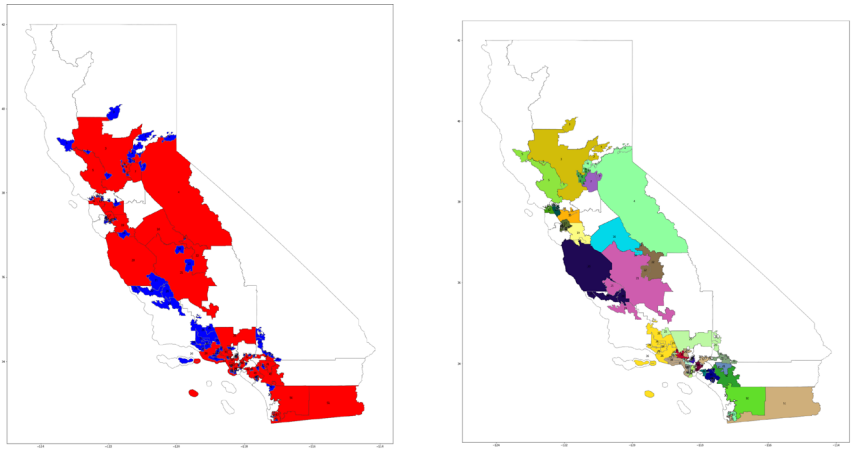
Fig. 2 Side by side comparison of the original congressional district boundaries and the ones after re-clustering with respect to *Race: White*. Numbers on the left figure indicate the congressional district number of each region whilst numbers on the right indicate the congressional district ZCTAs that have moved. (These numbers can be seen by magnifying the two panels)

– **Q3**. Can we make the 53 CDs fairer with respect to multiple PSVs at the same time? Further, how does the shapes of the CDs change?

To explore **Q1**, we summarize results generated by our method under some spatial constraints (described in Sect. 7.2) to make the CDs fairer with respect to different PSVs (see Table 10). To explore the second parts of **Q1** and **Q2**, we visualize the changes of shapes of the CDs by highlighting the resultant redistricted ZCTAs with respect to different PSVs in Figs. 2, 3, 4 and 5. To explore **Q3**, we explore different pair-wise combinations of PSVs in Table 12. We also highlight the resultant redistricted ZCTAs in making the CDs fairer with respect to 5 PSVs simultaneously in Fig. 6.

7.2 Additional constraints

Due to spatial requirements, one cannot arbitrarily move a ZCTA from one CD to another; for example, moving a ZCTA currently allocated to San Jose (which is located towards the Northern part of California) to Los Angeles (which is in the Southern part of the state) is not permitted. To facilitate this, we add a simple allocation constraint encoded in a matrix M. Here, M is an $n \times k$ matrix, where n and k represent the number of ZCTAs and CDs respectively. An entry of $M_{i,j} = 1$ indicates ZCTA i can be allocated to CD j. The sum of each row in M is lower bounded by 1, and is often equal to 2 for ZCTAs that border two CDs. (In some unusual cases, the sum of the entries in a row of M can be as high as 4 for ZCTAs on the boundary of that many CDs.) For $1 \leq i \leq n$ and $1 \leq j \leq k$, letting z_{ij} denote a $\{0, 1\}$-valued variable that indicates whether the redistricting step allocates ZCTA i to CD j, the constraint that



(a) All the ZCTAs (marked by the blue regions) redistricted during re-clustering (with respect to *Race: White*) and the new CDs (illustrated in red) to which they are redistricted.

(b) The new CDs after re-clustering with respect to *Race: White* are highlighted. All the other CDs are unchanged and are colored white.

Fig. 3 Re-districting ZCTAs to make CDs fairer with respect to the PSV *Race: White* while ensuring spatial continuity of CDs through constraints based on the M matrix. Numbers in the figures (which can be seen by magnifying the figures) indicate the congressional district number of each region

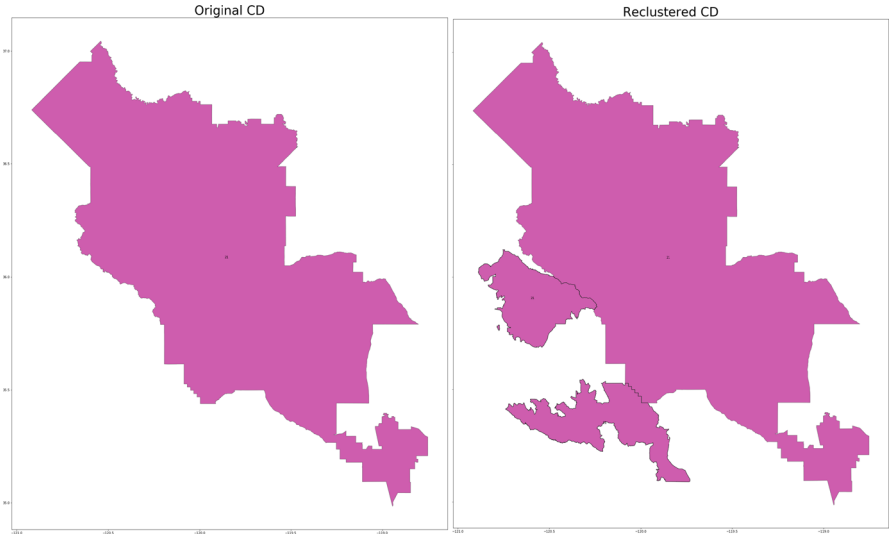


Fig. 4 Detailed view of the shape of the original Congressional District #21 before (**left**) and after (**right**) the re-clustering step. (Re-clustering was done with respect to *Race: White*)

Table 10 The explored PSVs and which constraints yield feasible solutions in balancing each PSV

PSV	Ensuring Spatial Continuity (PSVs for all CDs must be ±20% of non-PSVs)	Relaxation of Fairness for *every* CD (80% of CDs must be fair)	Relaxation of Spatial Continuity
Race: White	✓	✓	✓
Female	✓	✓	✓
People over 16 Not in Labor Force	✓	✓	✓
Foreign Born	×	×	✓
Hispanic/Latino	×	×	✓
Citizen	✓	✓	✓
Age: 55+	✓	✓	✓
Total Household Income over 100k+	×	×	✓
Language Other Than English Spoken at Home	×	×	✓

The PSVs were selected by sorting the PSVs in decreasing order of variance with respect to population across the CDs and choosing the top nine PSVs

encodes the spatial requirement is simply:

$$z_{i,j} \leq M_{i,j}, \quad 1 \leq i \leq n, \ 1 \leq j \leq k \tag{5}$$

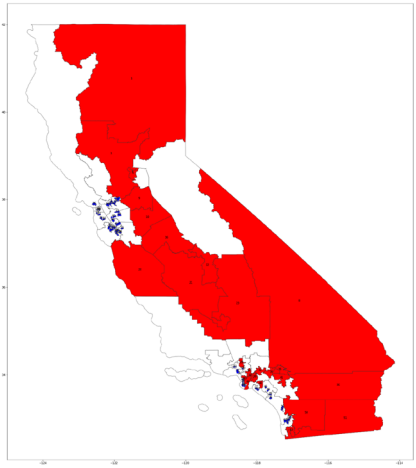
We add in a constraint to ensure the CD sizes in terms of population stay approximately the same.

7.3 Experimental results for single PSVs

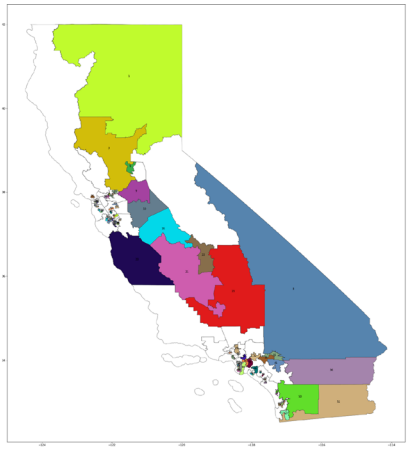
We present the results of experiments that explore which PSVs can be balanced individually to make the CDs fairer according to disparate impact. First we overview the results for balancing one PSV as an example and then balance many PSVs simultaneously.

Results for Fairness wrt- *Race:White*. The left panel of Fig. 2 shows the boundaries of the CDs in the original allocation and the right panel of the figure shows the new boundaries of the CDs when our method is used (along with spatial constraints given by Eq. (5)) to balance the PSV *Race: White*. To provide more details about the results, Fig. 3a shows all ZCTAs that were redistricted during re-clustering and their new CDs after the re-clustering step. Figure 3b illustrates the specific ZCTAs redistricted and the new congressional districts to which they belong. Note how the new ZCTAs and the corresponding CDs are contiguous. This is because of the spatial constraints mentioned above. Figure 4 provides a detailed view of the shape of one congressional district before and after re-clustering.

Results for Fairness wrt Remaining PSVs. The number of redistricted ZCTAs varies with the PSVs. The specific number of ZCTA-instances that needed to be redistricted



(a) Highlighted in blue are the ZCTAs that were redistricted and in red, their new CDs after re-clustering with respect to the PSV *Total Household Income Over 100k*.



(b) Reclustered CDs with respect to the PSV *Total Household Income Over 100k*. Each ZCTA and new CD pair are colored the same. Note that it is much harder to visually see this as the ZCTA and its CD are no longer required to be spatially next to one another.

Fig. 5 Re-districting ZCTAs to make CDs fairer with respect to the PSV *Total Household Income Over 100k* Numbers in the figures indicate the congressional district number of each region. For this experiment, the continuity constraint (see Sect. 7.2) is relaxed

for each PSV is shown in Table 11. The specific subset of ZCTAs redistricted also varies across the investigated PSVs. The results in Table 11 demonstrate that it is easier to make the CDs fair with respect to some PSVs (e.g., *Gender: Female* needs only 6 ZCTAs to be redistricted) whilst others are more challenging (e.g., *Race: White* needs 108 ZCTAs to be redistricted). Table 10 shows that for some PSVs (e.g., *Language other than English Spoken at Home*) redistricting for fairness couldn't be done when spatial continuity constraints were imposed. Table 11 also shows the number of ZCTAs that were redistricted for such PSVs when spatial constraints were relaxed.

7.4 Exploring fairness across multiple PSVs

Here, we explore the question of redistricting so that fairness is achieved with respect to each of two or more PSVs. We first consider pairs of PSVs and then discuss the case of a subset consisting of 5 PSVs.

We considered each pair from the set of nine PSVs shown in the leftmost column of Table 10. Our goal was to check whether fairness could be achieved with respect to each of the two PSVs. Our formulation produced feasible solutions to some pairs of PSVs while other pairs were infeasible. The results are shown in Table 12, where the cells in Cyan represent feasible pairs while the ones in Yellow are infeasible pairs. (The numerical value in each cell indicates the correlation between the corresponding pair of PSVs.) This exploration is important in understanding the compatibility between

Table 11 The difficulty of re-districting to ensure fairness for a single PSV

PSV	Constraint Type	No. of ZCTA-Instances Redistricted
Female	Ensuring Spatial Continuity	6
Not in Labor Force	Ensuring Spatial Continuity	13
Citizen	Ensuring Spatial Continuity	16
Age 55+	Ensuring Spatial Continuity	46
Race: White	Ensuring Spatial Continuity	108
Language Other Than English Spoken at Home	Relaxation of Spatial Continuity	31
Foreign Born	Relaxation of Spatial Continuity	32
Income 100k+	Relaxation of Spatial Continuity	59
Hispanic/Latino	Relaxation of Spatial Continuity	107 (This needed the re-districting of some ZCTAs in the center of CDs.)

For each PSV, the number of ZCTAs redistricted during re-clustering is shown. For the last four PSVs in the table, spatial continuity constraints also needed to be relaxed to achieve fairness

the fairness of different PSVs. For instance, ensuring fairness with respect to the PSV *Gender: Female* may cause unfairness for another PSV.

For PSV pairs in which fairness could be achieved under spatial continuity constraints for each individual PSV, our method was also able to achieve fairness for both PSVs. This may appear to be surprising, but Table 12 also shows that these variables are highly correlated. Hence the tasks of balancing fairness for the two PSVs are mutually complementary.

Our experiments also showed that redistricting to achieve fairness with respect to each of the five PSVs in the set {*Female*, *Race: White*, *Age: 55+*, *Citizen*, *Not in Labor Force*} was feasible. The notion that one can simultaneously achieve fairness for each of five PSVs is not intuitive. Nonetheless, we observe from Table 12 that this particular set of PSVs is highly positively correlated. The re-clustering step for this set of five PSVs used the spatial continuity constraints and caused 129 ZCTAs to be re-districted. The ZCTAs that were re-districted and the resulting new CDs are shown in the left and right panels of Fig. 6.

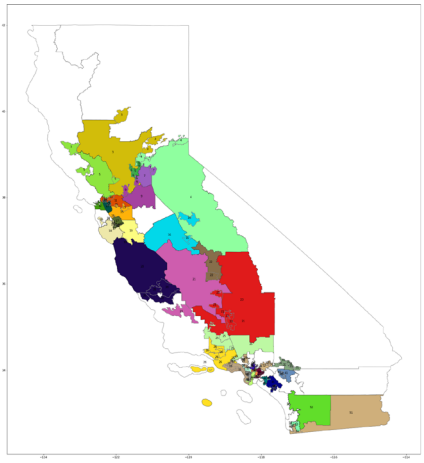
8 Discussion, conclusions and future work

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 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 that some variations such as a relaxed condition for fairness and overlapping clusters also lead to TU constraint matrices. However, we also observed that some interesting settings such as continuous protected variables may not lead to TU matrices.

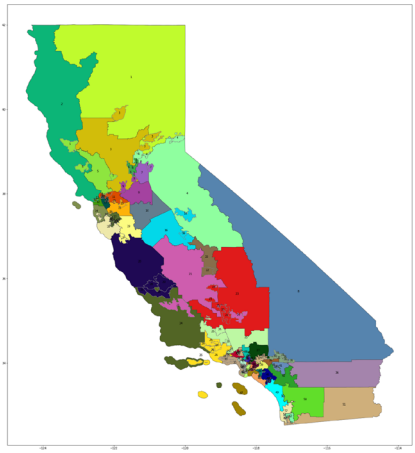
Table 12 The numerical value in a cell indicates the pairwise correlations between PSVs whilst the color of the cell indicates if there exists a feasible re-clustering between the corresponding pair of PSVs

PSV	Over 100k	Female	Race: White	Age > 55	Citizen	Not in labor force	Other language	Hispanic /Latino	Foreign Born
Over 100k		0.7618	0.7361	0.8258	0.8315	0.7097	0.4838	0.3004	0.5912
Female			0.8994	0.9362	0.9738	0.9700	0.8634	0.7944	0.8674
Race: White				0.8816	0.9108	0.8895	0.6615	0.6793	0.6473
Age > 55					0.9625	0.9415	0.7206	0.6175	0.7589
Citizen						0.9614	0.7560	0.6728	0.7796
Not in labor force							0.8233	0.7636	0.8198
Other Language								0.9138	0.9731
Hispanic /Latino									0.8086
Foreign Born									

The PSV combinations in yellow are infeasible while the ones in Cyan are feasible. The spatial continuity constraints have been relaxed to achieve fairness. (For space reasons, names of many PSVs have been abbreviated)



(a) The ZCTAs that were redistricted for re-clustering to make all 5 PSVs (*Female*, *Race: White*, *Age: 55+*, *Citizen*, *Not in labor force*) fairer simultaneously.



(b) The new 53 congressional districts after re-clustering to make all 5 PSVs fairer simultaneously.

Fig. 6 Re-districting ZCTAs to make CDs fairer with respect to 5 PSVs (*Female*, *Race: White*, *Age: 55+*, *Citizen*, *Not in labor force*) simultaneously. Numbers in the figures indicate the congressional district number of each region. For this complex problem the continuity constraint (see Sect. 7.2) is relaxed

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 finding a clustering to satisfy popular **must-link** constraints (which can encode individual-level fairness) is also tractable, satisfying **both is computationally 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 (e.g., some data sets as big as 300K instances could be processed in 5 min 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 (e.g., 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 showed that our measure of fairness does not produce results that are fundamentally different from those 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.

Future Work: Our theoretical work considered a fairness measure for a single PSV. A challenging research direction is to extend our work to allow multiple PSVs. Also, developing appropriate notions of fairness and investigating how clusters can be modified to improve fairness in the context of non-partitional clusterings with single or multiple PSVs is an interesting topic for future work. We showed that for a certain combination of group and individual-level fairness specifications, the feasibility questions may become computationally intractable. It is of interest to investigate the algorithmic and complexity aspects of combining other forms of fairness criteria.

Acknowledgements We thank the referees for carefully reading the paper and providing very helpful feedback. We also thank Professor Seshadhri Comandur (University of California, Santa Cruz) for pointing out that the sufficient condition that we use for establishing the TU property of the constraint matrix in Sect. 3 is called the Ghouila-Houri characterization in the literature. This work was supported in part by NSF Grants IIS-1908530 and IIS-1910306 titled: "Explaining Unsupervised Learning: Combinatorial Optimization Formulations, Methods and Applications".

A Additional material for Sect. 3

A.1 A short introduction to integer linear programs

Many combinatorial optimization problems can be expressed as Integer linear programs (ILPs) (Garey and Johnson 1979; Schrijver 1998). An ILP is specified by a set of variables that are constrained to take on integer values, a linear objective function of these variables and a collection of linear constraints that each solution must

satisfy. In general, unless **P** = **NP**, no efficient algorithms are possible for solving ILPs (Garey and Johnson 1979). However, the availability of well known software tools (e.g., Gurobi 2020), which incorporate many heuristic search methods, make it possible to use ILPs to solve problems of moderate size arising in practice. We now provide an example of a combinatorial problem that can be formulated as an ILP.

Example: We consider the **Knapsack** problem, where we are given a collection of n objects. Each object o_i has a weight w_i (pounds) and value d_i (dollars), $1 \leq i \leq n$. We are also given a knapsack whose total capacity is W (pounds). The goal is to choose a subset of the items so that the total weight of the items is at most W and the total value of all the items is a *maximum* subject to this constraint. This problem is known to be **NP**-complete (Garey and Johnson 1979). An ILP for this problem can be developed as follows.

Our ILP formulation uses n variables denoted by $x_1, x_2, \ldots, x_n$. Each of these variables takes on a value from $\{0, 1\}$ with the following interpretation: object o_i is added to the knapsack iff $x_i = 1$. With these variables, the optimization goal can be expressed as:

Maximize $\sum_{i=1}^{n} d_i x_i$.

Since each d_i is a (given) constant, this objective function is linear.

We now discuss the constraints. First, the total weight of the chosen items must be at most the capacity of the knapsack. This constraint can be expressed as follows:

$\sum_{i=1}^{n} w_i x_i \leq W$

Note that each w_i is a given constant. Thus, this constraint is linear. The other constraint, which restricts the value of each x_i, is as follows: $x_i \in \{0, 1\}, \quad 1 \leq i \leq n$.

This completes the ILP formulation of an ILP for the Knapsack problem. Many examples of such formulations are discussed in standard texts on algorithms and related topics (e.g., Garey and Johnson 1979; Cormen et al. 2009; Vazirani 2001). Many methods for solving ILPs are discussed in Schrijver (1998).

A.2 An example to illustrate the ILP formulation for MCMF

We present an example to show the constraint matrix that arises in the ILP formulation of MCMF presented in Sect. 3. For simplicity, we will construct this example assuming that we need strict fairness.

In this example, we have a set $S = \{s_1, s_2, s_3, s_4, s_5\}$ consisting of 5 instances. Of these, instances s_1, s_2 and s_3 are special (i.e., their PSV values, denoted by p_1, p_2 and p_3 respectively, are all 1) while s_4 and s_5 are not special (i.e., $p_4 = p_5 = 0$). The initial clustering of S has two clusters C_1 and C_2, where $C_1 = \{s_1, s_2, s_3\}$ and $C_2 = \{s_4, s_5\}$.

Since the number of special items $N_x = 3$ and the number of clusters $K = 2$, under strict fairness, each cluster must have either $\lceil N_x/K \rceil = \lceil 3/2 \rceil = 2$ special items or $\lfloor N_x/K \rfloor = \lfloor 3/2 \rfloor = 1$ special item. Thus, the given clustering (which has all the three special items in C_1) is not strictly fair, and we need to modify it to achieve fairness.

From the discussion in Sect. 3, let $z_{i,j}$ denote the {0,1}-valued variable that is set to 1 if in the modified clustering, instance s_j is assigned to cluster C_i, $1 \leq j \leq 5$ and $i = 1, 2$. (Otherwise, $z_{i,j}$ is set to 0.) As discussed above, the upper and lower

bounds on the number of special items in each cluster are 2 and 1 respectively. Let u_i and l_i denote the slack variables for Cluster C_i, $i = 1, 2$. Using Eqs. (2) through (4) for the two clusters, and noting that only p_1, p_2 and p_3 are 1, we get the following constraints:

$$\begin{aligned} z_{1,1} + z_{1,2} + z_{1,3} + u_1 &= 2 \\ -z_{1,1} - z_{1,2} - z_{1,3} + l_1 &= -1 \\ z_{2,1} + z_{2,2} + z_{2,3} + u_2 &= 2 \\ -z_{2,1} - z_{1,2} - z_{2,3} + l_2 &= -1 \\ z_{1,1} + z_{2,1} &= 1 \\ z_{1,2} + z_{2,2} &= 1 \\ z_{1,3} + z_{2,3} &= 1 \\ z_{1,4} + z_{2,4} &= 1 \\ z_{1,5} + z_{2,5} &= 1 \end{aligned}$$

Note that the constraint matrix uses only the coefficients on the left side of each constraint above. As indicated in Sect. 3, we use the following ordering of the variables so that each constraint (which becomes a row of the constraint matrix) can be expressed as a linear combination of the variables in this order:

$$\langle z_{1,1}, z_{1,2}, z_{1,3}, z_{1,4}, z_{1,5}, z_{2,1}, z_{2,2}, z_{2,3}, z_{2,4}, z_{2,5}, u_1, u_2, l_1, l_2 \rangle$$

Thus, the resulting constraint matrix C has 9 rows (one corresponding to each constraint) and 14 columns (one corresponding to each variable). Variables that don't appear in a constraint have their coefficients as 0 in the constraint matrix. From the above constraints, we get the following constraint matrix C with 9 rows (corresponding to the constraints) and 14 columns (corresponding to the variables).

1	1	1	0	0	0	0	0	0	0	1	0	0	0
−1	−1	−1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	1	1	1	0	0	0	1	0	0
0	0	0	0	0	−1	−1	−1	0	0	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0

B An alternative algorithm for modifying a given clustering to achieve strong fairness

In Sect. 3, we mentioned that it is possible to develop a faster algorithm for achieving strong fairness. In that section, we also presented the basic ideas behind the algorithm. Here, we provide a description of the algorithm.

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 strong fairness.) Thus, the input to the algorithm is an arbitrary partition Π of D_x have $k \geq 1$ clusters denoted by $C_1, C_2, \ldots, C_k$, 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. It first moves the minimum number of special items into a temporary container T and then redistributes those items to clusters which need additional special items to satisfy the fairness condition. 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 $C_1, \ldots, C_t$ 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 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 $C_1, \ldots, C_t$ with $> \lceil N_x/k \rceil$ special items.
- Let Γ_2 consist of clusters $C_{t+1}, \ldots, C_p$ 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 $C_{p+1}, \ldots, C_m$ with exactly $\lfloor N_x/k \rfloor$ special items.
- Let Γ_4 consist of the remaining clusters, that is, $C_{m+1}, \ldots, C_k$ 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 $\Pi' = \langle C_1, C_2, \ldots, C_k \rangle$.

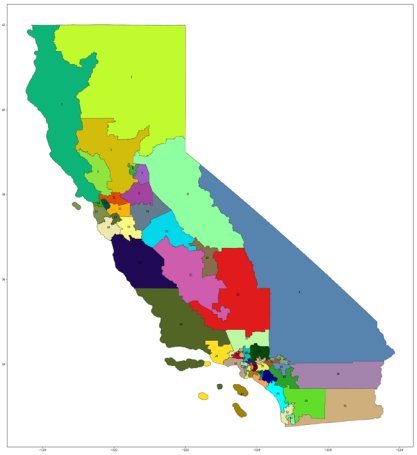
Running Time of Algorithm OPT-Modification: As mentioned earlier, sorting the list of clusters so that their sizes are in non-increasing order can be done in $O(k \log k)$ time. The remaining part of the algorithm consists of several cases, exactly one of which is executed (depending on the values of N_x and k). In each case, the algorithm moves the excess special data items from some of the clusters into a temporary container and redistributes the items from that container to clusters that are deficient with respect to special data items. Since there are at most n special items, the time needed for redistribution step is $O(n)$. Thus, the running time of this algorithm is $O(n + k \log k)$.

Limitations of the Above Algorithm: Algorithm OPT-Modification has two limitations. First, the algorithm handles only strong fairness. Second, it minimizes the number of special data items moved from one cluster to another; it cannot handle more general minimization objectives mentioned in Table 1. The LP-based algorithm discussed in Sect. 3 overcomes both of these limitations but has an asymptotically larger running time.

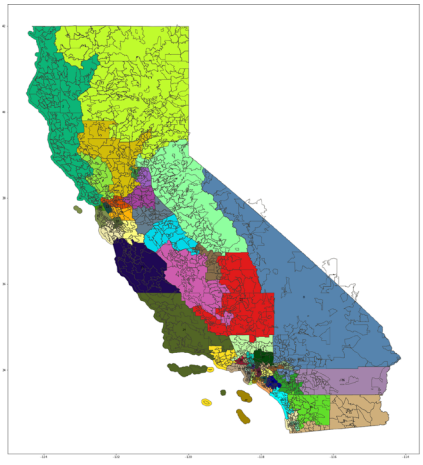
C Other details about California's congressional districts dataset and our experimental settings

C.1 Background on congressional districts

One of the most important aspects of any election is how the voters are represented. How lines are drawn to separate congressional districts can heavily impact an election and how well everyone is represented. As an example, consider the 2016 presidential election where the boundaries of the electoral college led to the election of a candidate even though they did not win the popular vote. California is divided into 53 congressional districts (CDs) (CNMP 2020), and these congressional districts are redrawn every decade (NCSL 2019). The most recent congressional districts are shown in Fig. 7a. The selection committee that creates these lines focuses on ensuring the relatively equal sizes of populations across the 53 CDs (Ballotpedia 2020). However, based on this strategy, the congressional district lines are unfair for some protected status variables (PSVs) as shown in Table 9.



(a) Visualization of California's 53 congressional districts. Numbers in the figures indicate the congressional district number of each region



(b) Mapping of the 1769 ZCTAs to the 53 congressional districts. Note how the boundaries of the ZCTAs are shown in black and some ZCTAs are shared with neighboring states.

Fig. 7 California's 53 congressional districts and 1769 ZCTAs

In this paper, we explore the fairness of California's congressional districts with respect to a number of PSVs collected from the Census Bureau's 2018 American Community Survey (Bureau 2020a). After identifying the PSVs with respect to which the CDs are unfair, we attempt to redistribute the population to correct this unfairness.

As previously mentioned, California is divided into 53 CDs. Each CD consists of a subset of the 1769 ZCTAs (Zip Code Tabulation Areas) as shown in Fig. 7b (Bureau 2020b). Each ZCTA consists of multiple census blocks. ZCTAs are similar but not identical to zip codes used the United States Postal Service (USPS). The latter are managed by the USPS and can be arbitrarily changed, while ZCTAs only change every decade. Additionally, ZCTAs are only used by the Census Bureau. Typically, each of the 1769 distinct ZCTAs in California is assigned exclusively to *one* CD, but some ZCTAs on the boundaries of CDs can be shared by multiple CDs.

C.2 Protected status variable creation

The data utilized during the case study is from the Census Bureau's ACS (American Community Survey) 2018 Data (Bureau 2020a), which provides highly detailed socioeconomic, housing, and demographic information of a given population (Bureau 2020a). This data forms the basis of our protected status variable (PSVs). In an effort to protect people's privacy, the PSVs are presented in an aggregated manner via population numbers (e.g., how many females are in a ZCTA). There was difficulty finding this information at the census block level, hence information at a ZCTA level was used instead. Additionally, a relationship file is used to map ZCTA membership in a congressional district. If a ZCTA is shared by multiple congressional districts, its population is distributed equally amongst those CDs to create multiple ZCTA-Instances.

References

Abbasi M, Bhaskara A, Venkatasubramanian S (2021) Fair clustering via equitable group representations. In: Proceedings of FAccT, p 11

Ahmadi S, Galhotra S, Saha B, Schwartz R (2020) Fair correlation clustering. CoRR https://arxiv.org/abs/2002.03508

Ahmadian S, Epasto A, Kumar R, Mahdian M (2020) Fair correlation clustering. In: The 23rd international conference on artificial intelligence and statistics, AISTATS 2020, 26–28 August 2020, Online [Palermo, Sicily, Italy], pp 4195–4205

Backurs A, Indyk P, Onak K, Schieber B, Vakilian A, Wagner, T (2019) Scalable fair clustering. In: Proceedings of 36th ICML, pp 405–413

Ballotpedia: Ballotpedia. (2020). Retrieved 2 September 2020, from (2020). https://ballotpedia.org/Redistricting_in_California

Barocas S, Selbst AD (2016) Big data's disparate impact. California Law Rev 671:671–732

Barocas S, Hardt M, Narayanan A (2017) Fairness in machine learning. NeurIPS tutorial

Basu S, Davidson I, Wagstaff K (2008) Constrained clustering: advances in algorithms, theory and applications. CRC Press, Cambridge

Bera SK, Chakrabarty D, Negahbani M (2019) Fair algorithms for clustering. ArXiv preprint arXiv:1901.02393

Berge C (1972) Balanced matrices. Math Program 2:19–31

Bureau C (2020a) Bureau, Census. American Community Survey (ACS). Retrieved 2 September 2020, from (2020). https://www.census.gov/programs-surveys/acs

Bureau C (2020b) Bureau, Census. ZIP Code Tabulation Areas (ZCTAs). Retrieved 2 September 2020, from (2020). https://www.census.gov/programs-surveys/geography/guidance/geo-areas/zctas.html
Chen X, Fain B, Lyu L, Munagala K (2019) Proportionally fair clustering. In: Proceedings of the 36th international conference on machine learning, ICML 2019, 9–15 June 2019, Long Beach, California, USA, Proceedings of Machine Learning Research, pp 1032–1041
Chhabra A, Mohapatra P (2020) Fair algorithms for hierarchical agglomerative clustering. CoRR https://arxiv.org/abs/2005.03197
Chhabra A, Masalkovaite K, Mohapatra P (2021) An overview of fairness in clustering. IEEE Access 9:130698–130720
Chierichetti F, Kumar R, Lattanzi S, Vassilvitskii S (2017) Fair clustering through fairlets. In: Proceedings of NeurIPS, pp 5036–5044
CNMP: Center for New Media & Promotion (CNMP), U (2020) My Congressional District. Retrieved 2 September 2020, from (2020). https://www.census.gov/mycd/?st=06
Commission UEEO (2007) Employment tests and selection procedures. Retrieved 2 September 2020, from. https://www.ncsl.org/research/redistricting/election-dates-for-legislators-governors-who-will-do-redistricting.aspx
Cormen TH, Leiserson CE, Rivest RL, Stein C (2009) Introduction to algorithms, 2nd edn. MIT Press and McGraw-Hill, Cambridge
Davidson I, Ravi SS (2007) The complexity of non-hierarchical clustering with instance and cluster level constraints. Data Min Knowl Discov 14(1):25–61
Davidson I, Ravi SS (2020) Making existing clusterings fairer: algorithms, complexity results and insights. In: The thirty-fourth AAAI conference on artificial intelligence. AAAI New York, NY, USA. AAAI Press, pp 3733–3740
Dua D, Graff C (2017) UCI machine learning repository. http://archive.ics.uci.edu/ml
Dwork C, Hardt M, Pitassi T, Reingold O, Zemel RS (2012) Fairness through awareness. In: Innovations in theoretical computer science 2012, Cambridge, MA, USA, January 8–10, 2012, pp 214–226
Feldman M, Friedler SA, Moeller J, Scheidegger C, 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, pp 259–268
Flores NJ (2019) Fair algorithms for clustering. Dartmouth Computer Science Technical Report TR2019-867
Friedler SA, Scheidegger C, Venkatasubramanian S (2016) On the (im)possibility of fairness. CoRR http://arxiv.org/abs/1609.07236
Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman & Co., San Francisco
Gurobi optimizer reference manual. Available from https://www.gurobi.com/documentation/9.1/refman/index.html (2020)
Kleindessner M, Awasthi P, Morgenstern J (2019) Fair k-center clustering for data summarization. In: Proceedings of ICML, pp 3448–3457
Kleindessner M, Samadi S, Awasthi P, Morgenstern J (2019) Guarantees for spectral clustering with fairness constraints. In: Proceedings of ICML, pp 3458–3467
Kuo CT, Ravi SS, Dao TBH, Vrain C, Davidson I (2017) A framework for minimal clustering modification via constraint programming. In: AAAI, pp 1389–1395
Mahabadi S, Vakilian A (2020) Individual fairness for k-clustering. In: Proceedings of the 37th international conference on machine learning, ICML 2020, 13–18 July 2020, Virtual Event, pp 6586–6596
NCSL: NCSL (2019) Election Dates for Legislators and Governors Who Will Do Redistricting. Retrieved 2 September 2020, from (2019). https://www.ncsl.org/research/redistricting/election-dates-for-legislators-governors-who-will-do-redistricting.aspx
Rösner C, Schmidt M (2018) Privacy preserving clustering with constraints. ArXiv preprint arXiv:1802.02497
Schaffer C (1994) A conservation law for generalization performance. In: Proceedings of ICML, pp 259–265. Elsevier, New York
Schrijver A (1998) Theory of linear and integer programming. Wiley, New York
Thanh BL, Ruggieri S, 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, pp 502–510

Vaidya PM (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, pp 332–337

von Luxburg U (2006) A Tutorial on spectral clustering. Tech. Rep. TR-149, Max Planck Institute for Biological Cybernetics, Germany

Vazirani VV (2001) Approximation algorithms. Springer, New York

Wagstaff K, 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, pp 1097–1192

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evolut Comput 1(1):67–82

Xu R, Wunsch DC (2005) Survey of clustering algorithms. IEEE Trans Neural Netw 16(3):645–678

Ziko IM, Yuan J, Granger E, Ayed IB (2021) Variational fair clustering. In: Proceedings of thirty-fifth conference on artificial intelligence (AAAI). AAAI Press, pp 11202–11209