5-29-2019

Fair Algorithms for Clustering

Nicolas J. Flores
Dartmouth College

FAIR ALGORITHMS FOR CLUSTERING

DARTMOUTH COMPUTER SCIENCE TECHNICAL REPORT TR2019-867

A PREPRINT

Nicolas J. Flores
Department of Computer Science
Dartmouth College
Hanover, NH 03755
`nicolasflores.19@dartmouth.edu`
Advisor: Deeparnab Chakrabarty

May 29, 2019

1 Introduction

In recent years, machine learning algorithms have grown in popularity and found new applications in everyday life. However, one major concern with these algorithms is their usage as "black boxes", which some say is problematic because the results they produce are not always fair. For example, COMPAS, an algorithm produced by Northpointe Inc., has been used to advise judges on the recidivism risk of potential parolees, but has recently been alleged to be prejudicial [1]. Specifically, the algorithm was more likely to misclassify black inmates as being likely to commit a violent crime if released. This disparity in treatment has obvious consequences for the livelihood of these citizens and has led researchers (and the public in general) to demand fairer algorithms. This example begs two questions: what does it mean for an algorithm to be fair and how do we design algorithms to achieve this notion of fairness?

There are two major schools of thought on fairness: fairness of process and fairness of outcome [2]. As the names would suggest, fairness of process would try to make the processes as fair as possible, while fairness of outcome would try to make the outcomes as egalitarian as possible. For example, in the case of COMPAS, fairness of process could be achieved by simply omitting details about race and ethnicity when giving the candidate profile to the algorithm. However, the COMPAS algorithm actually followed this procedure and had no direct information about the inmate's race. However, it is still possible to make biased decisions because other attributes in the data might be used as proxies for sensitive ones. We might find that ZIP code of a candidate may be able to tell us about the likely demographic of our candidate. This motivates us to look at fairness of outcome.

The *disparate impact* (DI) doctrine makes it illegal for protected groups to suffer adverse impact because of unfair hiring practices in the United States[3]. This legal doctrine began with the Supreme Court case Griggs v. Duke Power Co [3], which mandated that Duke Power Co.'s usage certain of certain applicant qualities was unconstitutional because it led to racial discrimination in their hiring practices. Although the applicant qualities were not explicitly race-based, they were correlated to race and led to a disparate impact in their hiring. The definition of adverse impact is intentionally nebulous, but is sometimes quantified in employment as "less than 80% of the group with the highest [hiring] rate" [4], and is known informally as the 80% rule. This *disparate impact* notion of fairness is the current interpretation of fairness in the machine learning community. Feldman et al. [5] and Zafar et al. [6] use this lens to study designing fair classification, while Celis et al. [7][8] use disparate impact to study the fair ranking and multiwinner voting problem respectively. Chierichetti et al. [9] was the first one to apply the disparate impact notion of fairness to the clustering problem.

In this work, we apply this notion of fairness to the classic unsupervised learning algorithm: clustering. Intuitively the clustering problem can be described as:

Given a set of data points, partition them into a set of groups which are as similar as possible. [10]

This algorithm is one of the most popular unsupervised learning algorithms and has a host of applications. For example, clustering has many real-world applications in analyzing customer data, such as for market segmentation and collaborative filtering, where it is used to group customers based on their similarity [10]. Another application of clustering is as a preprocessing step for classification problems, where the cluster that a data point belongs to can be used as an additional feature. One of the reasons for its popularity is that there are a multitude of ways to cluster data according to different notions of distance. Popular distance metrics for clustering include the l^1, l^2 and l^∞ norms considered here (*k-median*, *k-means* and *k-center* objectives, respectively).

Fairness in Clustering To apply the notion of fairness to clustering, we now allow every point to belong to multiple protected color groups. For example, in a dataset, a point might represent a person, where the two protected attributes might be race and sex. In this example, the color would be the category of the protected attribute that the point belonged to (ie. male or female for sex). It should be obvious that clustering does not inherently guarantee fair results because it only aims to partition the data such that the distance metric it is given is reduced. Here we modify the problem to find a fair solution, while still attempting to find minimal values for that metric. We define a fair clustering as a clustering which aims to balance the amount of each color in each of the clusters it produces. The amount that we determine as fair for each color would depend on the latent distribution of that color in the entire dataset. For example, if the distribution of male-female in a hypothetical dataset was 80-20, we would define fair clusters as those where the distribution of male-female in that cluster was similarly 80-20. Because achieving this perfect notion of balance can in some cases be extremely difficult, we introduce a looser notion of fairness with two additional parameters: α and β. For any color i, we define two quantities β_i and α_i, which define the minimum and maximum level of representation we deem fair for that color. Notice that this allows us to extend our notion of fairness to include the disparate impact doctrine, if we set the $\beta_i = 0.8 * r_i$ for every color where r_i is defined as the ratio of that color in the dataset. Additionally, we note that disparate impact is itself not illegal and that Supreme Court found that disparate impact is illegal if it cannot be classified as a "business necessity" [3]. Our algorithm aims to not only design a fair solution to the clustering problem, but we also allow comparison between the unfair and fair solutions to show how the implementation of fairness constraints affects the quality of cluster solutions.

2 Previous Work and Summary of Contributions

In this section, we discuss the previous work in this space, and summarize how our own work extends and generalizes it. Before we discuss the work directly related to ours, we present those that are in the fair clustering but study a slightly different problem. Chen et al. [11] study a slightly different notion of fair clustering than the one looked at here where protected group greater than a certain size are allowed to pick a cluster center. In addition, Kleindessner et al. [12] instead enforces fairness on the points chosen as cluster centers rather than the points assigned to these centers.

Chierichetti et al. [9] was the seminal work in this space to explore the notion of fairness for the clustering problem. The notion of fairness they applied was the disparate impact notion explored earlier. However, one of the major limitations of this work is that they only allow the points in the dataset to belong to two color groups. To quantify fairness in a clustering, they define a notion of balance based on the ratio of the quantities of the two colors in each cluster. For example, using RED and BLUE as our two colors, they define balance as:

$$\text{balance} = \min\left(\frac{\#of\text{RED}}{\#of\text{BLUE}}, \frac{\#of\text{BLUE}}{\#of\text{RED}}\right)$$

Their approach focuses on a preprocessing step where they perform a *fairlet decomposition* of the set of points, X. A fairlet is defined as a clustering where the fairness is preserved within the cluster. The fairlet decomposition step of their algorithm involves finding m many fairlets to break the dataset X into and then performing a vanilla clustering of the centers of these m fairlets to produce k clusters [9]. This approach is currently the most popular in the fair clustering space and the two other works we consider in this section also utilize it.

Schmidt et al. [13] builds on Chierichetti et al. [9] by considering the fairness problem with the k-means objective. This works similarly assumes that there is a single protected attribute with only two color classes. They show how to extend the fairlet decomposition approach to the k-means objective and provide experimental validation of their results as well. Backurs et al. [14] uses the two color notion of fairness defined above to look at designing scalable fair algorithms for clustering using the k-median objective.

The other major work to consider is that of Rösner and Schmidt [15], who extend the notion of fairness in clustering to multiple colors for the k-center and k-median objectives. Instead of considering balanced clustering, as in Chierichetti et al. [9], they define fair clustering as *exactly balanced clustering*. Under *exactly balanced clustering*, a cluster is fair if and only if the ratio between colors in that cluster exactly matches the ratio in the dataset. Thus, this notion of fairness is extremely brittle and highly dependent on the latent distribution of colors in the dataset. For example, if the quantity of two colors in the dataset are coprime then the only "fair" solution under *exactly balanced clustering* is to have exactly one cluster.

Our own work generalizes prior work first because we allow our points to belong to multiple protected colors, while not being limited by the extremely stringent *exactly balanced clustering* notion. Second, we crucially allow these color classes to overlap (one point can belong to multiple protected attributes). This allows our algorithm to find a fair solution to the clustering problem along multiple dimensions. Rösner and Schmidt technically could capture this by defining a new color class as the intersection between any two overlapping color classes. For example, if two overlapping color classes are "female" and "Hispanic", a new color could be define instead as "female and Hispanic". However, it is easy to see that such an approach would lead to an explosion in the number of color classes. Furthermore, this would likely further hamstring their approach because of the stringent notion of fairness defined by *exactly balanced clustering*. Third, our algorithm allows for every color class i to have its own β_i and α_i specifying the ratio of its representation within a given cluster, meaning that our algorithm allows for practitioners to trade-off between objective cost and fairness. One caveat of our algorithm is that in theory it incurs a small additive violation of $4\Delta + 3$, where Δ is defined as the maximum amount of overlap between color classes. However, we show in Section 4.4 that in our experiments this additive violation is never greater than 3.02, much smaller than our theoretical guarantee. Finally, our own algorithm is objective neutral, meaning that it can work with any practitioner-defined metric easily. Furthermore, our own algorithm may be more practical in deployment because it performs fairness as a post-processing step rather than a pre-processing step and does not require knowledge of the clustering step, only the centers found by it. This means that currently unfair clusterings being used can be converted to fair solutions with only knowledge of the unfair cluster centers. For our experiments, we use the k-median, k-means and k-center objectives because they are historically popular and because they allow for comparison to prior work.

3 Theoretical Introduction

3.1 Preliminaries

In this section I discuss the major theoretical contributions made by Suman K. Bera, Maryam Neghabani and Deeparnab Chakrabarty that occurred before I joined the project [16].

First, we formally define a VANILLA (k, p)-CLUSTERING. This is the traditional clustering without any fairness conditions. The input for this problem is a triple (X, d, k) where (X, d) is a metric space and k is a positive integer telling us the *"number of clusters"*. The clustering algorithm proceeds by partitioning the points X into $F \cup C$. Note that these are not necessarily disjoint and in most interpretations of the problem, $X = F = C$. Finally, the solution returned is a set of centers (S, ϕ), such that $S \subseteq F$ and $|S| \leq k$. ϕ is the assignment function ($\phi : C \to S$) that maps the points to a given center, such that the *clustering objective* is minimized:

$$L_p(S, \phi) := (\sum_{v \in C} d(v, \phi(v))^p)^{\frac{1}{p}}$$

Finally we note that p is implicit in the problem definition and is the l_p norm for the distance. In our work we looked at the popular settings of $p \in \{1, 2, \infty\}$, which correspond to the k-median, k-means, and k-center objectives respectively. The notion of an assignment function (ϕ) for VANILLA (k, p)-CLUSTERING is odd because each point is necessarily assigned to its closest center. However, in the FAIR (k, p)-CLUSTERING problem we see that this is not necessarily the case. Thus, we introduce the notation here to draw parallels between the two problems.

We now consider converting this VANILLA (k, p)-CLUSTERING problem to a fair problem. To do this, we introduce the concept of "color classes" that partition the data into ℓ not necessarily disjoint subsets: $C = C_1 \cup C_2 \cup C_3 ... \cup C_l$. To be concrete, Chierichetti et. al[9] consider the case of $\ell = 2$. Furthermore, Rösner and Schmidt [15] consider the case of general ℓ, but both these works necessitate that the color classes do not overlap. To quantify this notion we introduce the notation Δ, which is the maximum number of color classes that any $v \in C$ can belong to. To introduce the notion of fairness between these classes we define two parameters $\alpha, \beta \in [0, 1]^\ell$. Our notion of fairness says that we would like the representation of any two classes should be within some desired range. This can be expressed as the following:

$$\beta_i \cdot \big|\{v \in C : \phi(v) = f\}\big| \leq \big|\{v \in C_i : \phi(v) = f\}\big| \qquad \forall i \in [\ell], \forall f \in S \tag{MP}$$

$$\big|\{v \in C_i : \phi(v) = f\}\big| \leq \alpha_i \cdot \big|\{v \in C : \phi(v) = f\}\big| \qquad \forall i \in [\ell], \forall f \in S \tag{RD}$$

In plain English, this means that the representation of a given color i in a certain cluster f must be at least β_i and at most α_i. We call these conditions, minority protection (MP) and restricted dominance (RD) respectively. To intuit this fractional interpretation of β_i and α_i, realize that we can divide on both sides of the inequality by the total number of people in the cluster:

$$\beta_i \leq \frac{\left|\{v \in C_i : \phi(v) = f\}\right|}{\left|\{v \in C : \phi(v) = f\}\right|} \leq \alpha_i \qquad \forall i \in [\ell], \forall f \in S$$

We define a FAIR (k, p)-CLUSTERING as a VANILLA (k, p)-CLUSTERING that additionally accepts parameters α, β and that all clusters satisfy the inequalities laid out above. To distinguish unfair clustering from fair clustering, we will denote the tuple returned from fair clustering as $(S, \hat{\phi})$ where $\hat{\phi}$ denotes an assignment that we deem fair (satisfies inequalities). Note that in the fair clustering problem, we are still minimizing the *clustering objective* laid out above.

Finally, we because the algorithm we present is an *almost* fair algorithm for clustering, we define what it means for our clustering to make additive violations. An *almost* fair clustering that makes λ additive violations is defined as:

$$-\lambda + \beta_i \cdot \left|\{v \in C : \phi(v) = f\}\right| \leq \left|\{v \in C_i : \phi(v) = f\}\right| \leq \alpha_i \cdot \left|\{v \in C : \phi(v) = f\}\right| + \lambda \qquad \forall i \in [\ell] \quad \forall f \in S$$

Intuitively, we can think of λ as the amount of people that we would need to add or remove from this class to make the constraint fair. However, this intuition is a little simplistic because it doesn't take into account the fact that adding/removing from one color class in a certain cluster f affects the ratio of other color classes in cluster f.

3.2 Reduction

The general technique for the algorithm is reducing the FAIR (k, p)-CLUSTERING to a fair assignment problem. This is done by using VANILLA (k, p)-CLUSTERING to generate a set of k centers and from there performing a fair assignment from the points to centers (leaving the centers fixed). This method differs heavily from the fairlet decomposition approach previously used by Chierichetti [9] and Rösner and Schdmit [15]. Their approach first considers satisfying fairness and then attempts to performing clustering on top of this to minimize the clustering objective. Our approach leverages VANILLA (k, p)-CLUSTERING to find the centers for us, which is advantageous in the case where the assignment produced by VANILLA (k, p)-CLUSTERING is close to being fair. In the empirical results, we show that although the balance in many clusters is extremely low, these clusters may actually be very close to fair in terms of the number of people that need to be reassigned.

3.3 Algorithm

Given as input (X, d, k), our algorithm proceeds in three steps. First, we use VANILLA (k, p)-CLUSTERING as a black box to obtain (S, ϕ). Next, we use the set of centers S to perform a partial assignment between our points and these centers. Finally, we use an iterative rounding procedure to convert our partial solution to an integral one.

$$\mathsf{LP} := \min \sum_{v \in C, f \in S} d(v, f)^p x_{v,f} \qquad x_{v,f} \in [0, 1], \ \forall v \in C, f \in S \tag{LP}$$

$$\sum_{v \in C_i} x_{v,f} \leq \alpha_i \sum_{v \in C} x_{v,f} \qquad \forall f \in S, \forall i \in [\ell] \tag{1a}$$

$$\sum_{v \in C_i} x_{v,f} \geq \beta_i \sum_{v \in C} x_{v,f} \qquad \forall f \in S, \forall i \in [\ell] \tag{1b}$$

$$\sum_{f \in S} x_{v,f} = 1 \qquad \forall v \in C \tag{1c}$$

The linear program detailed in LP shows the implementation of the fair partial assignment algorithm. The algorithm can be described as setting up a connection between every point v and every center S with the assignment having weight $x_{v,f}$. We specify that every point that sum of a point's partial assignments must sum to 1 (Equation 2c). Furthermore, we specify in Equations 2a and 2b that the partial assignments to every cluster must satisfy the β_i, α_i bounds that we specified above.

After running the linear program, we obtain x^*, which is a vector of partial assignments to centers. We then run Algorithm 1, which uses LP2, to convert the fair partial solution to a fair integral one, with additive violations. In

Algorithm 1 Algorithm for the FAIR p-ASSIGNMENT problem

```
1: procedure FAIRASSIGNMENT($(\mathcal{X}, d)$, $S$, $C = \cup_{i=1}^{\ell} C_i$, $\vec{\alpha}, \vec{\beta} \in [0,1]^{\ell}$)
2:     $\hat{\phi}(v) = \emptyset$ for all $v \in C$
3:     solve the LP given in eq. (1), let $x^\star$ be an optimal solution
4:     for each $x^\star_{v,f} = 1$, set $\hat{\phi}(v) = f$ and remove $v$ from $C$ (and also relevant $C_i$s).
5:     let $T_f := \sum_{v \in C} x^\star_{v,f}$ for all $f \in S$
6:     let $T_{f,i} := \sum_{v \in C_i} x^\star_{v,f}$ for all $i \in [\ell]$ and $f \in S$
7:     construct LP2 as given in eq. (2), only with variables $x^\star_{v,f} > 0$
8:     while there exists a $v \in C$ such that $\hat{\phi}(v) = \emptyset$ do
9:         solve LP2, let $x^\star$ be an optimal solution
10:        for each $x^\star_{v,f} = 0$, delete the variable $x^\star_{v,f}$
11:        for each $x^\star_{v,f} = 1$, set $\hat{\phi}(v) = f$ and remove $v$ from $C$ (and also relevant $C_i$s). Reduce $T_f$ and relevant $T_{f,i}$s by 1.
12:        for every $i \in [\ell]$ and $f \in S$, if $|x^\star_{v,f} : 0 < x^\star_{v,f} < 1,\ v \in C_i| \le 2(\Delta + 1)$ remove the respective constraint in eq. (2a)
13:        for every $f \in S$, if $|x^\star_{v,f} : 0 < x^\star_{v,f} < 1,\ v \in C| \le 2(\Delta + 1)$ remove the respective constraint in eq. (2b)
```

Algorithm 1, the idea is to fix the points who are already integrally assigned to a center. Then, the algorithm iteratively rounds the points if the assignments are "close enough" to being fair. Close enough is determined if the number of points left to assign in a color in a cluster or in a cluster are smaller than $2\Delta + 2$. The final assignment generated by this iterative rounding procedure is our *almost* fair clustering, $\hat{\phi}$. In this step, we framed the rounding procedure as an instance of the *minimum degree-bounded matroid basis* problem and use the result from Király et al. [17] to guarantee our theoretical bounds. Thus, we chose the bounds for "close enough" in our algorithm based on the result from this work.

$$\mathsf{LP2} := \min \sum_{v \in C, f \in S} d(v,f)^p x_{v,f} \qquad x_{v,f} \in [0,1],\ \forall v \in C, f \in S \tag{LP2}$$

$$\lfloor T_{f,i} \rfloor \le \sum_{v \in C_i} x_{v,f} \le \lceil T_{f,i} \rceil \qquad \forall f \in S, \forall i \in [\ell] \tag{2a}$$

$$\lfloor T_f \rfloor \le \sum_{v \in C} x_{v,f} \le \lceil T_f \rceil \qquad \forall f \in S, \forall i \in [\ell] \tag{2b}$$

$$\sum_{f \in S} x_{v,f} = 1 \qquad \forall v \in C \tag{2c}$$

3.4 Theoretical Guarantees

Based on the algorithm given above we now show that the maximum additive violation due to our algorithm is $4\Delta + 3$. Let $\overline{T}_f, \overline{T}_{f,i}$ be the final number of points assigned to cluster f and assigned to cluster f in color i respectively. Then, we know that $\overline{T}_{f,i} \le \lceil T_{f,i} \rceil + 2\Delta + 1$ because this is most we allowed our algorithm to violate. Then,

$$\overline{T}_{f,i} \le \lceil \alpha_i T_f \rceil + 2\Delta + 1 \le \alpha_i \lfloor T_f \rfloor + 2\Delta + 2 \le \alpha_i(\overline{T}_f + 2\Delta + 1) + 2\Delta + 2 \le \alpha_i \overline{T}_f + (4\Delta + 3)$$

The first inequality follows from the fact that $\lceil T_{f,i} \rceil \le \lceil \alpha_i T_f \rceil$ based on the first partial LP. The second and fourth inequality based on the definition of α as ≤ 1. The third inequality follows from $\overline{T}_f \le \lceil T_f \rceil + 2\Delta + 1$, which is again how much we allowed our algorithm to violate by.

3.5 Implementation Notes

One remark we should make here is that the LP we suggest doesn't make sense for the $p = \infty$ or k-center objective. Thus, we use a binary search to find the lowest value of the LP that is feasible. We begin with a guess G of the optimal LP value. Next, for all $x_{v,f}$ such that $d(v,f) > G$ we set $x_{v,f} = 0$ because according to our guess this connection does not occur. We attempt to solve the LP and if it returns feasible we know that our guess G is an upper bound on the cost of the LP. If our LP is infeasible than we know the cost of a fair solution must be greater than G. Thus, we continue until we find a minimum value for the LP.

4 Experimental Results

In this section we cover the experimental results obtained by implementing our algorithm on datasets from the UCI Machine Learning repository [18]. To judge our algorithm, we examined the increase in the cost associated with achieving fairness (the *cost of fairness*) and the *balance* of clusters to determine the fairness of our clustering. Finally, we also look at the additive violations associated with our fair clustering because as noted our algorithm finds *almost* fair clusterings. In brief, our results show that the *cost of fairness* associated with our algorithm is very low and that the additive violations associated with our algorithm was much lower than our theoretical guarantee of $4\Delta + 3$.

4.1 Setup

All experiments were written in Python 3.6 and we run all our experiments on a Macbook Air with a 1.8 GHz Intel Core i5 Processor and 8 GB 1600 MHz DDR3 memory. We chose Python 3.6 to use the CPLEX Studio [19] to solve our linear programs. Because we were focused more on correctness than speed, our implementation in Python is not optimized for large datasets. Nevertheless, we ran trials exploring our algorithm's performance on larger datasets (number of points = $500,000$) to get a sense of its ability to scale.

Datasets. We use four datasets from the UCI repository [18]: [1] **(1)** `bank` [20] with 4,521 points, corresponding to phone calls from a marketing campaign by a Portuguese banking institution. **(2)** `census` [21] with 32,561 points, representing information about individuals extracted from the 1994 US `census` . **(3)** `creditcard` [22] with 30,000 points, related to information on credit card holders from a certain credit card in Taiwan. **(4)** `census1990` [23] with 2,458,285 points, taken from the 1990 US Census, which we use for run time analysis.

Table 1: For each dataset, the coordinates are the numeric attributes used to determined the position of each record in the Euclidean space. We chose the numeric attributes in `bank` and `census` to match those used by Chierichetti et al. [9]. The sensitive attributes determines protected groups.

Dataset	**Coordinates**	**Sensitive attributes**	**Protected groups**
`bank`	age, balance, duration	marital	married, single, divorced
		default	yes, no
`census`	age, education-num, final-weight, capital-gain, hours-per-week	sex	female, male
		race	Amer-ind, asian-pac-isl, black, other, white
`creditcard`	age, bill-amt 1 — 6, limit-bal, pay-amt 1 — 6	marriage	married, single, other, null
		education	7 groups
`census1990`	dAncstry1, dAncstry2, iAvail, iCitizen, iClass, dDepart, iFertil, iDisabl1, iDisabl2, iEnglish, iFeb55, dHispanic, dHour89	dAge	8 groups
		iSex	female, male

The datasets `bank` and `census` have become benchmark datasets for fair clustering after their usage by Chierichetti et al. [9]. In addition, Chierichetti et al. [9] used a third dataset "diabetes", which our team could not find in the UCI repository and seems to have been deleted. To make up for this, we include `creditcard` . Furthermore, because it was of interest to see how our algorithm scaled we use the `census1990` [23] dataset.

4.2 Metrics

We now formally define the two metrics we use in our experiments: *cost of fairness* and balance. Cost of fairness is the cost associated with converting a vanilla clustering to a fair clustering. Because the vanilla clustering objective

[1] https://archive.ics.uci.edu/ml/datasets/

is different per objective and per dataset, to standardize this notion, we define *cost of fairness* as the ratio of the fair clustering objective over the vanilla clustering objective. To define $\mathrm{balance}$ we extend the two color definition presented by Chierichetti et al. [9]. To be analogous to their measure, our measure should capture the fact that a clustering is unfair if any color is too under represented or over represented in a given clustering, relative to its representation in the dataset. To quantify this notion we introduce some notation. First, remember that C_i is the set of points who belong to color i, then define $C(f)$ as the set of points assigned to cluster f, $C(f) = \big|\{v \in C : \phi(v) = f\}\big|$. We further define $C_i(f)$ as $C_i \cap C(f)$ or the points that belong to color i and are assigned to cluster f. We then define $\mathrm{balance}(f)$ or the balance of cluster f as:

$$r_i = \frac{|C_i|}{|C|} \qquad r_i(f) = \frac{|C_i(f)|}{|C(f)|}$$

$$\mathrm{balance}(f) = \min\left(\frac{r_i(f)}{r_i}, \frac{r_i}{r_i(f)}\right) \quad \forall i \in [\ell]$$

The quantities r_i and $r_i(f)$ are the ratios of color i in the dataset and cluster f respectively. Intuitively, we define the balance of a cluster f as a quantity that measures how much the ratio of a color in cluster f deviates from its ratio in the dataset.

Although we could set the values of α_i and β_i for each color arbitrarily, we found that it made more sense to parameterize them by a single value δ:

$$\beta_i = r_i \cdot (1 - \delta) \qquad \alpha_i = r_i \cdot (\frac{1}{1-\delta})$$

This allows us to define the fair representation of a given color i with respect to its representation in the dataset r_i. Furthermore, we also use δ as an expression of how strict our fairness conditions are. Setting $\delta = 0$ in the equations above yields $\beta_i = \alpha_i = r_i$, meaning that the representation of a color in any cluster must be exactly equal to the representation of that color in the entire dataset. Furthermore, setting $\delta = 1$ for β_i and taking the limit as δ approaches 1 for α_i, yields $\beta_i = 0$ and $\alpha_i = +\infty$, suggesting that the representation of a color in a cluster can be anything. [2] This allows us to interpret δ as the looseness of our fairness constraints where a smaller value of δ corresponds to stricter fairness constraints.

4.3 Comparison with Vanilla Clustering

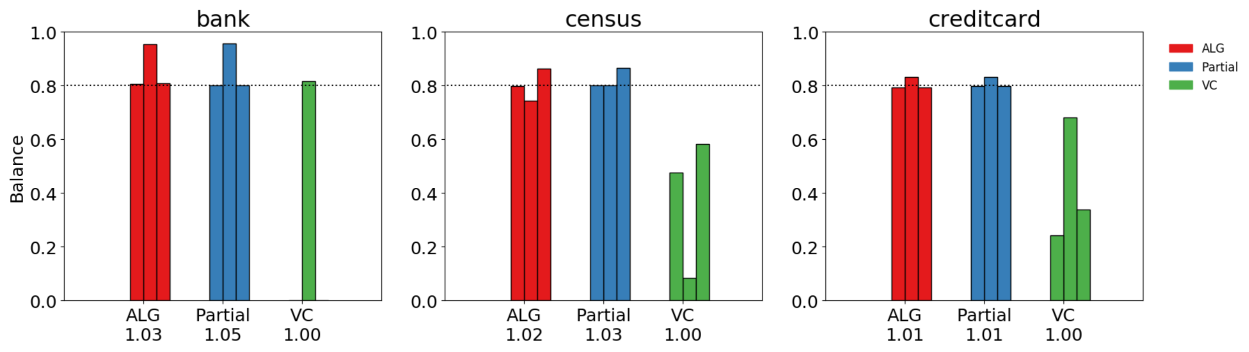

Figure 1: Comparison of our algorithm (ALG) versus vanilla clustering (VC) in terms of $\mathrm{balance}$ for the k-means objective. We use $k = 4$ and report the three largest clusters.

The first metric we examine our algorithm on is its ability to a find fair clustering as measured by balance. We do this on the `bank` , `census` , and `creditcard` datasets in Figure 1. Here we compare our algorithm ("ALG"), the vanilla cluster ("VC") and the partial solution found by LP ("Partial") on the k-means objective using $\Delta = 2$, where the protected

[2]It would be more appropriate here to define $\alpha_i = \min(1, r_i \cdot (\frac{1}{1-\delta}))$ as $\alpha_i > 1$ is nonsensical. However, I keep the definition shown above because it was the one used in the implementation of the algorithm. In addition, practically, setting $\delta = 1$ exactly is not very interesting because it corresponds to completely relaxed fairness conditions, which is simply vanilla clustering.

attributes are specified in 1. Additionally, because we use $\delta = 0.2$ (corresponding to the 80%-rule interpretation of the DI doctrine) in this trial we plot a dotted line at $\text{balance} = 0.8$ to indicate the minimum balance we desire from our fair clustering. As we can see, in every dataset examined, the vanilla clustering fails to have more than one fair cluster. On the other hand, our algorithm consistently finds fair clusters and in only cluster on the `census` dataset falls below our 0.8 goal. Furthermore, we quantified the balance of this cluster as 0.75, and as we show later on this discrepancy can be explained by a small additive violation. Next, we plot balance of our partial solution as a sanity check for our algorithm: because our partial solution is allowed to partially assign points to clusters it *must* be fair. As expected, in every dataset the balance of all clusters was at least 0.8.

Finally, below the x-axis labels we report the associated *cost of fairness*. The vanilla clustering's *cost of fairness* is shown as 1.00 to reiterate the fact that this is our benchmark in each case. We show that to achieve the DI doctrine interpretation of fairness in each of these datasets the largest associated *cost of fairness* was 3% if we allow for additive violations. Interestingly, we see that the associated *cost of fairness* is actually higher for our partial solution than it is for our algorithm. Although, seemingly not intuitive, this result occurs because we allow for our integral assignment to additive violations, while we do not allow this for our partial solution. This indicates that in many cases, points that were partially assigned were actually rounded to the center which minimized cost.

4.4 Analysis of Additive Violations

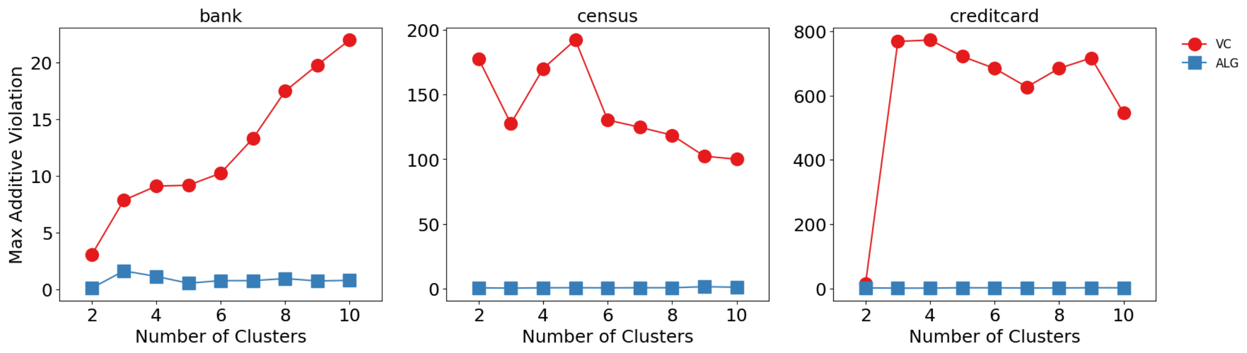

Figure 2: Comparison of the maximum additive violation (for $\delta = 0.2$ and $\Delta = 2$) over all clusters and all groups between our algorithm (ALG) and vanilla (VC), using the k-means objective.

Because our algorithm is *almost* fair it is of interest to examine how close we actually are to our target of fairness. In Figure 2, we examine this for the k-means objective with $\delta = 0.2$. On the y-axis we plot the maximum additive violation or how much any one color in any cluster is from being within the bounds of fairness (defined by α and β). Compared to the vanilla clustering, our algorithm makes a much smaller maximum additive violation. In fact, because we run this trial with $\Delta = 2$, we guarantee theoretically that our maximum additive violation is 11. However, in practice, it is much lower. In Table 2, we examine our maximum additive violation for $k \in [2, 10]$ and for a variety of δ and found that we never saw an additive violation greater than 3.02. In fact, for the `bank` and `census` datasets, we never saw greater additive violations than 2. We hypothesize that this occurred because our partial solutions were so close to being integral (in all cases the partial solutions were more than 99% integral) and so our iterative rounding algorithm was able to perform much better than the theoretical guarantee.

Table 2: The maximum additive violation across a range of δ of our algorithm compared to vanilla k-means. For each δ, we take maximum over k, for $k \in [2, 10]$ on all datasets.

δ	0.01	0.05	0.1	0.2	0.3	0.4	0.5	Vanilla ($\delta = 0.2$)
`bank`	1.45	1.17	1.39	**1.54**	1.19	1.15	1.03	**21.99**
`census`	1.44	1.53	**1.89**	1.08	1.18	0.97	1.03	**773.19**
`creditcard`	**3.02**	2.32	2.11	2.29	2.03	1.63	1.03	**192.01**

4.5 The case of $\Delta > 1$

In this section, we examine one of the unique strengths of our algorithm: the ability to handle multiple overlapping color groups ($\Delta > 1$). This is of particular interest because it represents a significant gain in practicality. In many real-world scenarios there are multiple protected classes which overlap, such as race and sex, and we would like to be fair on not only one of these, but both. We hypothesized that the when clustering with respect to one protected color attribute, the other attribute in a dataset would become unfair and vica versa. To test this, we used the two protected attributes described in Table 1 ran three trials for every dataset. In two trials we used the $\Delta = 1$ setting and attempted to find a fair clustering with respect to only one of the two attributes, and in the third trial we considered both attributes as protected. In Figure 3, we plot the results of these trials. For every dataset, we show two plots: one for the $\mathrm{balance}$ of each protected attribute, which is shown in the top right-hand corner of the plot. Furthermore, for each of these plots we show the results of all three trials. For example in `creditcard` , we considered the two protected attributes as "marriage" and "education", which represent marital status and education level respectively. In the left graph (the one with "marriage" in the top right hand corner and creditcard as the title), we show the results of all three trials. When we balance with respect to "marriage", the three clusters are all fair (one has a small additive violation) with respect to the marriage attribute. However when we balance only with respect to "education", the representation of different marital statuses is unbalanced. Furthermore, we show in the right graph ("education" in the top right), that reverse is true: balancing with respect to only "marriage" leads to a solution that is unfair with respect to education level. This demonstrates the need to consider both attributes if we desire a solution that is fair to both. As shown by the x-axis label "both", when we consider both attributes, our fair clustering is fair with respect to both attributes. Another enviable quality of our algorithm is that it is able to consider both these attributes in its fairness balancing while only modestly increasing the cost compared to only balancing with respect to one of them. The associated *cost of fairness* for these clusterings is reported below them along the x axis.

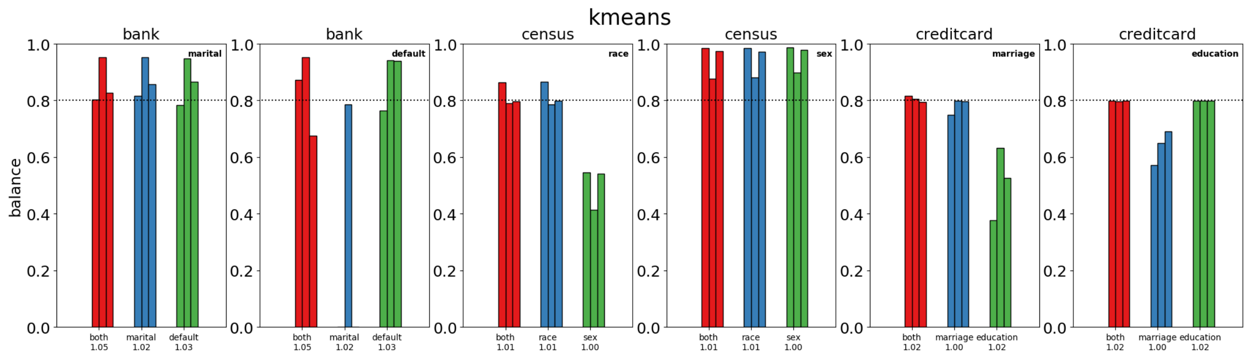

Figure 3: Comparison of clustering with only one protected attribute versus two protected attributes on k-means objective. We use $\delta = 0.2$ and $k = 4$ and report the three largest clusters.

An interesting result we find is that sometimes clustering with respect to one attribute is enough to make it fair for both. For example, when considering the `census` dataset, when we balanced with respect to "race", this solution was found to be fair to both "race" and "sex". However, the reverse is not true: performing a fair clustering with respect to "sex" was not enough to make it fair with respect to "race".

4.6 Tuning δ

In Figure 4, we show how varying the strictness of our algorithm's fairness parameters affects the associated *cost of fairness*. The power of these plots is the ability for machine learning practitioners to select their own value of δ based on the associated *cost of fairness*. This is important because as stated in the introduction, disparate impact is ruled illegal only if the businesses cannot defend it as a business necessity. In the case of clustering, whether or not disparate is "necessary" could be measured in terms of the clustering objective. Thus, the ability to vary the strictness of our fairness parameters allows us to choose an acceptable compromise between fairness and cost.

Observing the trend in Figure 4, we see that as δ increases, the cost of fairness drops to 1.0. Intuitively, this makes sense because as we relax our fairness conditions, we expect fair clustering to leave the vanilla solution untouched if the vanilla solution is deemed "fair enough". However, an interesting quirk of our algorithm is that is that sometimes increasing the value of δ leads to an increase in the *cost of fairness*. This is unexpected because as we increase δ, we are relaxing our fairness constraints. After investigation, we determined that this was due to randomness in our iterative

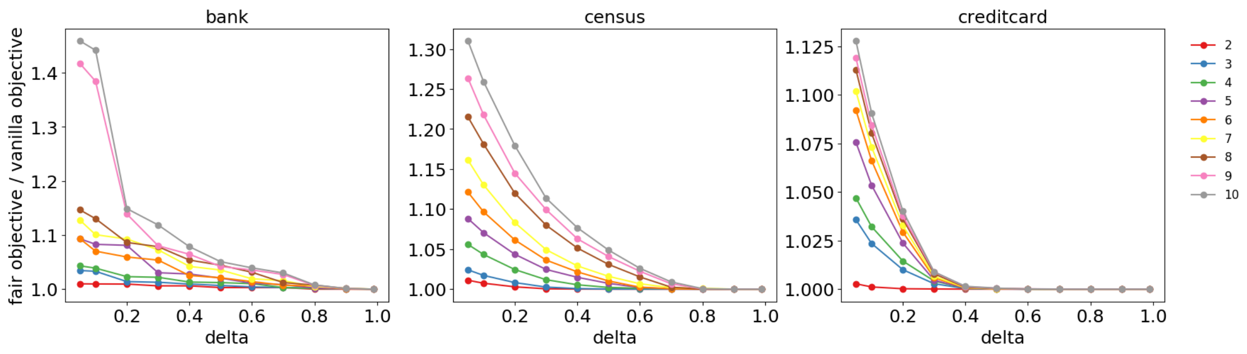

Figure 4: We show the effects of varying δ (x-axis) on our algorithm's fair objective cost over the vanilla cost (y-axis).

rounding algorithm. To further validate that our algorithm was working as intended, we observed the associated *cost of fairness* for our partial LP solutions and found that they monotonically decreased as we increased δ.

4.7 Runtime Analysis

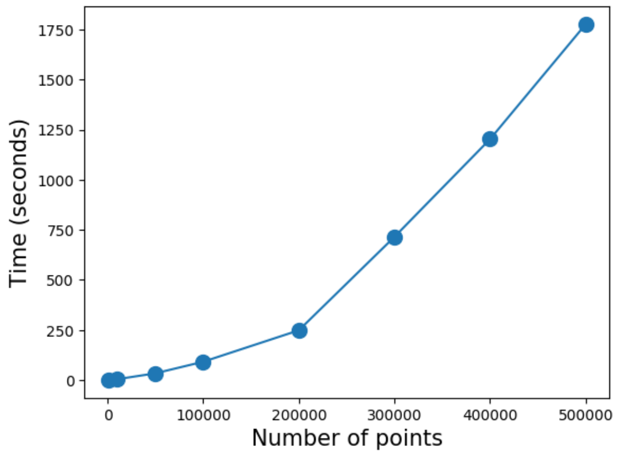

Figure 5: Runtime of our algorithm on subsampled data from `census1990` for k-means ($k = 3$)

Finally, we explore how our algorithm scales by observing its performance on a subsampled version of the `census1990` dataset. Because our algorithm is solving linear programs, its ability to scale to extremely large data is in question, but we see that it is able to comfortably find a solution for $500,000$ points in around half an hour (1776.5 seconds). One major concern of practitioners might be the notion that our algorithm not only has to solve one linear program but multiple! However, in practice, this fear is not realized because as stated earlier, the partial linear program (LP) finds a near integral solution. The iterative rounding linear program (LP2) does not consider points whose assignments are already integral, thus is much smaller than the first. Finally, an interesting thought is doing away completely with the linear program and instead solving an integer program to find fair clusters. This approach was able to find fair solutions for the `bank` (number of points = $4,521$) dataset reliably , but when tested on the `census` dataset (number of points = 32,561), it was unreliable and sometimes failed to terminate after an hour. Further, the integer program's ability to scale to large data is questionable.

5 Conclusions

The algorithm we present here significantly generalizes prior work in three major ways. First it allows for multiple overlapping color classes without the need for the stringency of *exactly balanced clustering*. Second, it allows for practitioners to tune the definition of fairness by adjusting α, β, which is practical because it allows for compromise between fairness and clustering quality. Finally, it generalizes the clustering objective. The one major caveat to our algorithm is that it theoretically allows for $4\Delta + 3$ additive violations to the fairness constraints. In practice, the maximum additive violation is often between 1 and 2 and was never above 3.02 in our trials.

References

[1] Adam Liptak. Sent to prison by a software's secret algorithms. *New York Times*, 2017.

[2] Gijs van de Kuilen Stefan T. Trautmann. Process fairness, outcome fairness, and dynamic consistency: Experimental evidence for risk and ambiguity. *Journal of Risk and Uncertainty*, 2016.

[3] Supreme Court of the United States. Griggs v. Duke Power Co. 401 U.S. 424, March 8 1971.

[4] The U.S. EEOC. Uniform guidelines on employee selection procedures, March 2 1979.

[5] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *Proc. 21st Annual SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 259–268, 2015.

[6] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi. Fairness constraints: Mechanisms for fair classification. In *Proc. 20th Proceedings, International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 962–970, 2017.

[7] L. Elisa Celis, Damian Straszak, and Nisheeth K. Vishnoi. Ranking with Fairness Constraints. In *Proc. 45th International Colloquium on Automata, Languages and Programming*, pages 28:1–28:15, 2018.

[8] L Elisa Celis, Lingxiao Huang, and Nisheeth K Vishnoi. Multiwinner voting with fairness constraints. In *IJCAI*, pages 144–151, 2018.

[9] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. In *Proc. 31st Conference on Neural Information Processing Systems*, pages 5029–5037, 2017.

[10] Charu C Aggarwal and Chandan K Reddy. *Data clustering: algorithms and applications*. CRC press, 2013.

[11] Xingyu Chen, Brandon Fain, Charles Lyu, and Kamesh Munagala. Proportionally fair clustering. In *Proc. 36th Proceedings, International Conference on Machine Learning (ICML)*, June 2019.

[12] Matthäus Kleindessner, Pranjal Awasthi, and Jamie Morgenstern. Fair k-center clustering for data summarization. In *Proc. 36th Proceedings, International Conference on Machine Learning (ICML)*, June 2019.

[13] Melanie Schmidt, Chris Schwiegelshohn, and Christian Sohler. Fair coresets and streaming algorithms for fair k-means clustering. *arXiv preprint arXiv:1812.10854*, 2018.

[14] Krzysztof Onak Baruch Schieber Ali Vakilian Arturs Backurs, Piotr Indyk and Tal Wagner. Scalable fair clustering. In *Proc. 36th Proceedings, International Conference on Machine Learning (ICML)*, June 2019.

[15] Clemens Rösner and Melanie Schmidt. Privacy Preserving Clustering with Constraints. In *Proc. 45th International Colloquium on Automata, Languages and Programming*, pages 96:1–96:14, 2018.

[16] Suman K. Bera, Deeparnab Chakrabarty, and Maryam Negahbani. Fair algorithms for clustering. *CoRR*, abs/1901.02393, 2019.

[17] Tamás Király, Lap Chi Lau, and Mohit Singh. Degree bounded matroids and submodular flows. *Combinatorica*, 32(6):703–720, 2012.

[18] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[19] IBM. Ibm ilog cplex 12.9. 2019.

[20] Paulo Rita Sérgio Moro, Paulo Cortez. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 2014.

[21] Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Annual SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1996.

[22] Che-hui Lien I-Cheng Yeh. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 2009.

[23] Christopher Meek, Bo Thiesson, and David Heckerman. The learning-curve sampling method applied to model-based clustering. *Journal of Machine Learning Research*, 2:397, 2002.