OPT2023: 15th Annual Workshop on Optimization for Machine Learning

Fair Minimum Representation Clustering

Connor Lawless CAL379@CORNELL.EDU
Oktay Günlük ONG5@CORNELLL.EDU
Operations Research and Information Engineering, Cornell University, Ithaca NY

Abstract

Clustering is an unsupervised learning task that aims to partition data into a set of clusters. In many applications, these clusters correspond to real-world constructs (e.g. electoral districts) whose benefit can only be attained by groups when they reach a minimum level of representation (e.g. 50% to elect their desired candidate). This paper considers the problem of performing k-means clustering while ensuring groups (e.g. demographic groups) have that minimum level of representation in a specified number of clusters. While incorporating this fairness criteria leads to an NP-Hard problem, we present an alternating minimization algorithm, called MiniReL, and describe computational approaches that make the algorithm practical even for large datasets.

1. Introduction

Clustering is an unsupervised learning task that aims to partition data points into sets of similar data points [17] and it is widely used in various domains [6, 10, 16]. However the wide-spread application of clustering, and machine learning broadly, to human-centric applications has raised concerns about its disparate impact on minority groups and other vulnerable demographics. Motivated by a flurry of recent results highlighting bias in many automated decision making tasks such as facial recognition[3] and criminal justice [12], researchers have begun focusing on mechanisms to ensure machine learning algorithms are *fair* to all those affected. One of the challenges of fairness in an unsupervised learning context, compared to the supervised setting, is the lack of ground truth labels. Consequently, instead of enforcing approximately equal error rates across groups, fair clustering generally aims to ensure that composition of the clusters or their centers (for settings like k-means and k-medians clustering) represent all groups fairly.

A common approach to fair clustering is to require each cluster to have a fair proportion of itself represented by each group (i.e., via balance [5] or bounded representation [2]). However, this approach does not have the desired effect in settings where a group only gains a significant benefit from the cluster when they reach a minimum level of representation in that cluster. Consider a voting system where there are no constraints on the contiguity of how districts are designed and the goal is to design districts where voters are close together. Here, a proportionally fair clustering would assign a minority group that represents 30% of the vote equally among each cluster. However, the minority group only gets a benefit (i.e., the ability to elect a candidate of their choice) if they have at least 50% representation in the cluster. To address this issue we introduce a new notion called *minimum representation fairness*, denoted MR-fairness, which requires each group to have a certain number of clusters where they have large representation (i.e., 50% in the voting example). We introduce a modified version of Lloyd's algorithm for k-means clustering that ensures MR-fairness, henceforth referred to as MINIimum REpresentation fair Lloyd's algorithm (MiniReL for short). The key modification behind our approach is to replace the original greedy assignment step of assigning data points to its closest cluster center in Lloyd's algorithm with an integer program (IP) that finds the minimum cost assignment while ensuring fairness. In contrast to the standard clustering setting, we show that finding a minimum cost clustering that respects MR-fairness is NP-Hard even when the cluster centers are already fixed. Through numerical experiments, we show that the IP approach can find locally optimal solutions even for large datasets.

1.1. Minimum Representation Fair Clustering Problem

The input to the standard clustering problem is a set of n m-dimensional data points $\mathcal{X} = \{x^i \in \mathbb{R}^m\}_{i=1}^n$. Note that assuming the data points to have real-valued features is not a restrictive assumption as categorical features can be converted to real-valued features through the one-hot encoding scheme. The goal of the k-means clustering problem is to partition the data points into a set of K clusters $\mathcal{C} = \{C_1, \dots, C_K\}$ with associated centers $c_1, \dots, c_K$ such that it minimizes the sum of the squared distances between data points and their assigned centers.

In the fair clustering setting, each data point belongs to one or more groups $g \in \mathcal{G}$ (i.e., gender, race). Let X_g be the set of data points belonging to group g. Note that unlike other fair machine learning work, we do not assume that the groups form a partition of the data points. For instance, if the groups correspond to race and gender then a data point can belong to more than one group. The key intuition behind MR-fair clustering is that individuals belonging to a group only gain material benefit if they have a minimum level of representation in their cluster. We denote this minimum representation threshold $\alpha \in (0, 1]$, and define the associated notion of an α-represented cluster as follows:

Definition 1 (α-represented Cluster) *A group $g \in \mathcal{G}$ is said to be α-represented in a cluster C_k if*

$$|C_k \cap X_g| \geq \alpha |C_k|$$

Note that α represents the minimum threshold needed for a given group to receive benefit from a cluster and thus depends on the application. For instance, most voting systems require majority representation (i.e., $\alpha = 0.51$). Our framework also allows for α to be group-dependent (i.e., α_g for each group g), however in most applications of interest α is a fixed threshold regardless of group. For a given clustering $\mathcal{C}$, group g, and α, let $\Lambda(\mathcal{C}, g, \alpha)$ be the number of clusters α-represented by group g. In MR-fairness, each group g has a parameter β_g that specifies a minimum number of clusters that should be α-represented by that group.

Definition 2 (Minimum representation fairness) *A given clustering $\mathcal{C} = \{C_1, \dots, C_K\}$ is said to be an $(\alpha, \boldsymbol{\beta})$-minimum representation fair clustering if for every group $g \in \mathcal{G}$:*

$$\Lambda(\mathcal{C}, g, \alpha) \geq \beta_g$$

for a given $\boldsymbol{\beta} = \{\beta_g \in \mathbb{Z}^+\}_{g \in \mathcal{G}}$.

The definition of MR-fairness is flexible enough that the choice of $\boldsymbol{\beta}$ can and should be specialized to each application as well as the choice of α. In the remainder of the paper we explore two different natural choices for $\boldsymbol{\beta}$ that mirror fairness definitions in the fair classification literature. The first sets β_g to be equal for all groups (i.e., $\beta_g = \left\lfloor \frac{1}{|\mathcal{G}|} \left\lfloor \alpha^{-1} \right\rfloor K \right\rfloor \quad \forall g \in \mathcal{G}$), which we denote

cluster statistical parity. The second set choice sets β_g to be proportional to the size of the group (i.e., $\beta_g = \left\lfloor \frac{|X_g|}{n} \lfloor \alpha^{-1} \rfloor K \right\rfloor \quad \forall g \in \mathcal{G}$), which we denote **cluster equality of opportunity**. We next formally define the optimization problem that we study:

Definition 3 (Minimum representation fair k-means problem) *For a given $\alpha \in (0,1]$ and $\boldsymbol{\beta} = \{\beta_g \in \mathbb{Z}^+\}_{g\in\mathcal{G}}$, the minimum representation fair k-means problem is:*

$$\min_{\mathcal{C}, c_1,\ldots,c_K \in \mathbb{R}^m} \sum_{k\in\mathcal{K}} \sum_{x^i \in C_k} \|x^i - c_k\|_2^2 \quad \textbf{\textit{s.t.}} \quad \Lambda(\mathcal{C}, g, \alpha) \geq \beta_g \ \ \forall g \in \mathcal{G}$$

The main difference between the fair and the standard versions of the k-means clustering problem is that greedily assigning data points to their closest cluster center may no longer be feasible for the fair version (i.e., assigning a data point to a farther cluster center may be necessary to meet the fairness criteria). Thus the problem can no longer be viewed simply as an optimization problem over cluster centers. A more detailed comparison to related work is included in the Appendix.

2. Mixed Integer Optimization Framework

We start by formulating the MR-fair clustering problem as a mixed-integer program with a non-linear objective. We use binary variable z_{ik} to denote if data point x^i is assigned to cluster k, and variable $c_k \in \mathbb{R}^d$ to denote the center of cluster k. Let y_{gk} be the binary variable indicating whether group g is α-represented in cluster k. Let $\mathcal{W} \subseteq \mathcal{G} \times \mathcal{K}$ be the set of allowable clusters that can be α-represented by each group. In most applications $\mathcal{W}$ will be equal to $\mathcal{G} \times \mathcal{K}$, however in some applications it can be beneficial to restrict this set. For instance, we show in Appendix C that pre-fixing groups to specific clusters (i.e., specifying cluster k must be α-represented by group g) can help break symmetry in the IP model and dramatically speedup the runtime of the algorithm.

We can now formulate the MR-fair clustering problem as follows:

$$\begin{aligned}
\textbf{min} \quad & \sum_{x^i\in\mathcal{X}} \sum_{k\in\mathcal{K}} \|x^i - c_k\|_2^2 z_{ik} && && (1)\\
\textbf{s.t.} \quad & \sum_{k\in\mathcal{K}} z_{ik} = 1 && \forall x^i \in \mathcal{X} && (2)\\
& \sum_{x^i\in X_g} z_{ik} + M(1 - y_{gk}) \geq \alpha \sum_{x^i\in\mathcal{X}} z_{ik} && \forall (g,k) \in \mathcal{W} && (3)\\
& \sum_{k\in\mathcal{K}:(g,k)\in\mathcal{W}} y_{gk} \geq \beta_g && \forall g \in \mathcal{G} && (4)\\
& u \geq \sum_{x^i\in\mathcal{X}} z_{ik} \geq l && \forall k \in \mathcal{K} && (5)\\
& z_{ik}, y_{gk} \in \{0,1\} && \forall x^i \in \mathcal{X}, (g,k) \in \mathcal{W} && (6)\\
& c_k \in \mathbb{R}^m && \forall k \in \mathcal{K} && (7)
\end{aligned}$$

The objective (1) is to minimize the cost of the clustering. Constraint (2) ensures that each data point is assigned to exactly one cluster. Constraint (3) tracks whether a cluster k is α-represented by a group g, and includes a big-M which can be set to αn. Finally, constraint (4) tracks that each group g is α-represented in at least β_g clusters. In many applications of interest, it might also be worthwhile to add a constraint on the size of the clusters to ensure that each cluster has a minimum/maximum number of data points. Constraint (5) captures this notion of a cardinality constraint where l and u represent the lower and upper bound for the cardinality of each cluster, respectively. Note that in cases where the cardinality constraint is used, the big-M in constraint (3) can be reduced to αu. For all our experiments we set $l = 1$ to ensure that exactly k clusters are returned by the algorithm. Adding a lower bound also ensures that each group is α-represented in non-trivial clusters. Note that every group would be trivially α-represented in an empty cluster according to Definition 1 but would provide little practical use.

2.1. MiniReL Algorithm

Solving the optimization problem outlined in the preceding section to optimality is computationally challenging as it is an integer optimization problem with a non-linear objective. Instead of solving this problem directly, we take an alternating minimization (AM) approach similar to Lloyd's algorithm and alternate between adjusting cluster centers and assigning data points to clusters to converge to a local optimum. Given a fixed set of cluster assignments (i.e. when variables z are fixed in (1)-(6)) the optimal choice of c_k is the mean value of data points assigned to C_k. Even when the cluster centers are given, the assignment problem in our fair setting is non-trivial due to constraint (4). For a given (fixed) set of cluster centers c_k we denote the problem (1)-(6) the *fair minimum representation assignment problem*. Note that unlike the full formulation, this is a linear integer program. In Lloyd's algorithm the assignment stage can be done greedily in polynomial time. However, the introduction of the fairness and cardinality constraints makes this no longer possible. The following result shows that even finding a feasible solution to the fair assignment problem is NP-Complete (proof can be found in the Appendix).

Theorem 4 *Finding a feasible solution to the fair minimum representation assignment problem is NP-Complete.*

In our algorithm we solve the fair assignment problem to optimality using integer programming. While integer programming tends not to scale well to large problems, in practice we observed the fair assignment problem to be computationally tractable for even datasets with thousands of data points. In Appendix C we also discuss some computational approaches to improve the run time for larger datasets. A natural question is whether MiniReL maintains the convergence guarantees of Lloyd's algorithm which is guaranteed to converge in finite time to a local optimum. The following result shows that the MiniReL also converges to a local optimum in finite time (proof can be found in the Appendix).

Theorem 5 *MiniReL converges to a local optimum in finite time.*

3. Numerical Results

To benchmark our approach, we evaluate it on three datasets that have been used in recent work in fair clustering: adult ($n = 48842$, $m = 14$) [7], default ($n = 30000$, $m = 24$) [7], and Brunswick County voting data ($n = 49190$, $m = 2$) [13]. For each dataset we use one sensitive feature to represent group membership - namely gender for both adult and default, and race for voting. We

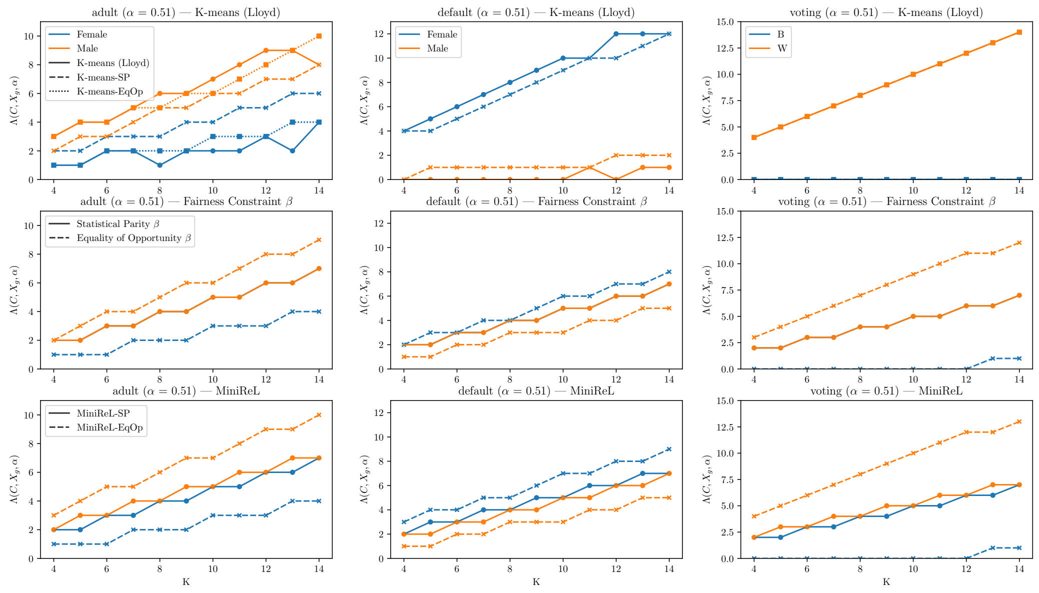

Figure 1: Number of α-represented clusters for k-means and MiniReL algorithms. k-means-SP and k-means-EqOp are the fairest results over 100 random seeds. Row 2 indicates the β values for cluster *statistical parity* and cluster *equality of opportunity*, respectively. Note that in all three plots in Row 2 both lines for Statistical Parity lie on top of each other.

normalize all real-valued features to be between $[0, 1]$ and convert all categorical features to be real-valued via the one-hot encoding scheme. For datasets that were originally used for supervised learning, we remove the target variable and do not use thise attribute as a feature for clustering.

We compare MiniReL against the standard Lloyd's algorithm for k-means using the implementation available in scikit-learn [14] with a k-means++ initialization. For all k-means results we re-run the algorithm with 100 different random seeds and report the result with the best k-means cost. We also report the fairest clustering found with respect to both fairness metrics over the 100 seeds, denoted *k-means-SP* and *k-means-EqOp* for the statistical parity and equality of opportunity critera, respectively. We implemented MiniReL in Python with Gurobi 10.0 [9] for solving all IPs. To initialize the cluster centers, we use a warm start scheme outlined in Appendix C. We ran MiniReL with $\boldsymbol{\beta}$ set for both cluster statistical parity (MiniReL-SP) and cluster equality of opportunity (MiniReL-EqOp). For the following experiments we set $\alpha = 0.51$ to represent majority representation in a cluster.

Figure 1 shows the number of α-represented clusters for each group using both k-means and MiniReL. Across all three datasets we can see that k-means can lead to outcomes that violate MR-fairness constraints, and that selecting the fairest clustering over 100 seeds has only marginal improvement. This is most stark in the default dataset where there is as much as an 11 cluster gap between the two groups despite having similar proportions in the dataset (60% and 40% for females and males, respectively). In contrast, the MiniReL algorithm is able to generate fair clusters under both notions of fairness and for all three datasets. Appendix D shows the average computation time in seconds for both algorithms. As expected, the harder assignment problem in MiniReL leads to higher overall computation times than the standard Lloyd's algorithm. However, it is still able to solve large problems in under 40 seconds demonstrating that the approach is still of practical use. A remaining question is whether fairness comes at cost to the quality of the clustering. Figure 2 shows

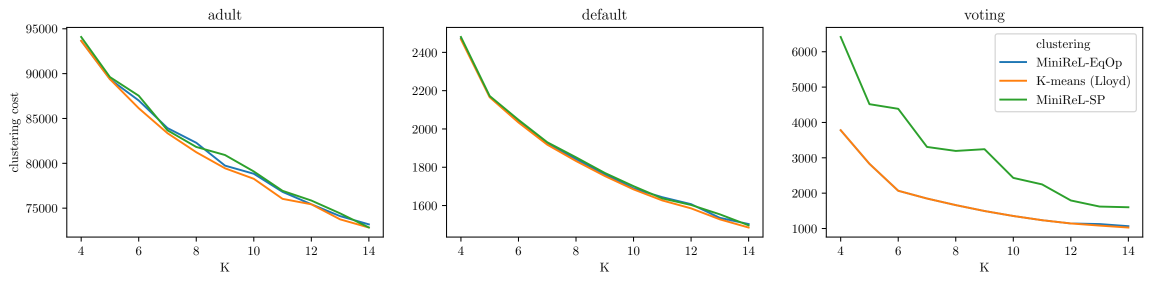

Figure 2: k-means clustering cost of Lloyd's algorithm and MiniReL.

the k-means clustering cost for Lloyd's algorithm and MiniReL under both definitions of fairness. Although there is a small increase in the cost when using MiniReL the overall cost closely matches that of the standard k-means algorithm in 5 out of 6 instances, showing that we can gain fairness at practically no additional cost to cluster quality. The only exception is the voting dataset under statistical parity, where we see a noticeable increase in clustering cost.

4. Conclusion

In this this paper we introduce a novel definition of group fairness for clustering that ensures each group achieves a minimum level of representation in a specified number of clusters. This definition is a natural fit for a number of real world examples, such as voting and entertainment segmentation. To create fair clusters we introduce a modified version of Lloyd's algorithm called MiniReL that includes an NP-hard fair assignment problem. To solve the fair assignment problem we use integer programming, and present some computational tools to improve the run-time of the approach. Unfortunately, solving the integer program remains a computational bottleneck of the approach and an important area of future research is to design more efficient algorithms for the problem or heuristics that can guarantee approximate fairness in polynomial time. Nevertheless we note that our approach is able to solve problems of practical interest, including datasets with thousands of data points, and provides a mechanism to design fair clusters when Lloyd's algorithm fails.

References

[1] Mohsen Abbasi, Aditya Bhaskara, and Suresh Venkatasubramanian. Fair clustering via equitable group representations. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 504–514, New York, 2021. ACM.

[2] Sara Ahmadian, Alessandro Epasto, Ravi Kumar, and Mohammad Mahdian. Clustering without over-representation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 267–275, New York, 2019. ACM.

[3] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91, New York City, USA, 2018. PMLR, PMLR.

[4] Anshuman Chhabra, Vidushi Vashishth, and Prasant Mohapatra. Fair algorithms for hierarchical agglomerative clustering, 2020. arXiv:2005.03197.

[5] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, Long Beach, USA, 2017. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper/2017/file/978fce5bcc4eccc88ad48ce3914124a2-Paper.pdf.

[6] Sher Muhammad Daudpota, Atta Muhammad, and Junaid Baber. Video genre identification using clustering-based shot detection algorithm. *Signal, Image and Video Processing*, 13(7): 1413–1420, 2019.

[7] Dheeru Dua, Casey Graff, et al. Uci machine learning repository, 2017.

[8] Mehrdad Ghadiri, Samira Samadi, and Santosh Vempala. Socially fair k-means clustering. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 438–448, Online, 2021. ACM.

[9] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022. URL https://www.gurobi.com.

[10] Tushar Kansal, Suraj Bahuguna, Vishal Singh, and Tanupriya Choudhury. Customer segmentation using k-means clustering. In *2018 international conference on computational techniques, electronics and mechanical systems (CTEMS)*, pages 135–139, Belgaum, India, 2018. IEEE, IEEE.

[11] Matthäus Kleindessner, Pranjal Awasthi, and Jamie Morgenstern. Fair k-center clustering for data summarization. In *International Conference on Machine Learning*, pages 3448–3457, Long Beach, USA, 2019. PMLR, PMLR.

[12] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning, 2019. arXiv:1908.09635.

[13] NCSBE. North carolina voter registration data, 2020. https://www.ncsbe.gov/results-data/voter-registration-data.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[15] Suhas Thejaswi, Bruno Ordozgoiti, and Aristides Gionis. Diversity-aware k-median: Clustering with fair center representation, 2021. arXiv:2106.11696.

[16] Yanshan Wang, Yiqing Zhao, Terry M Therneau, Elizabeth J Atkinson, Ahmad P Tafti, Nan Zhang, Shreyasee Amin, Andrew H Limper, Sundeep Khosla, and Hongfang Liu. Unsupervised machine learning for the discovery of latent disease clusters and patient subgroups using electronic health records. *Journal of biomedical informatics*, 102:103364, 2020.

[17] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

Appendix A. Related Work

A recent flurry of work in fair clustering has given rise to a number of different notions of fairness. One broad line of research, started by the seminal work of [5], puts constraints on the *proportion* of each cluster that comes from different groups. This can be in the form of balance [5] which ensures each group has relatively equal representation, or a group specific proportion such as the bounded representation criteria [2] or maximum fairness cost [4]. MR-fairness bares a resemblance to this line of work as it puts a constraint on the proportion of a group in a cluster, however instead of constraining a fixed proportion across all clusters it looks holistically across all clusters and ensures that threshold is met in a baseline number of clusters.

Another line of work tries to minimize the *worst case average clustering cost* (i.e. k-means cost) over all the groups, called social fairness [1]. Most similar to our algorithmic approach is the Fair Lloyd algorithm introduced in [8]. They also present a modified version of Lloyd's algorithm that converges to a local optimum. This approach requires a modified center computation step that can be done in polynomial time. The problem we study in this paper, however, requires a modified cluster assignment step that is NP-hard which we solve via integer programming.

Most similar to MR-fairness is diversity-aware fairness introduced in [15] and the related notion of fair summarization [11]. These notions of fairness require that amongst all the cluster centers selected, a minimum number comes from each group. Minimum fairness representation differs in that our criteria is not tied to the group membership of the cluster center selected but the proportion of each group in a given cluster. Our notion of fairness makes more sense in settings where the center cannot be prescribed directly, but is only a function of its composition (i.e. in voting where members of a 'cluster' elect an official).

Appendix B. Proof of Theorems

Theorem 6 *Finding a feasible solution to the fair minimum representation assignment problem is NP-Complete.*

Proof Given an instance of a 3-SAT problem, one of Karp's 21 NP-Complete problems, we construct an instance of the fair assignment problem as follows. We start with a 3-SAT problem with n variables and m clauses $K_1, \ldots, K_m$. Each clause K_i takes the form $v_{i1} \vee v_{i2} \vee v_{i3}$ where v_{ij} is either one of the original variables or its negation.

To construct the instance of the fair assignment problem, start by creating two new data points $x_{v_i}, x_{\bar{v}_i}$, corresponding to each original variable v_i and its negation, respectively. We construct two clusters C_1 and C_2 that each variable can be assigned to. For each original variable v_i we create one group $g_i = \{x_{v_i}, x_{\bar{v}_i}\}$ that can be α-represented in either cluster (i.e. $(g_i, 1), (g_i, 2) \in \mathcal{W}$). For each group g_i we set $\beta_{g_i} = 2$ - ensuring that both clusters must be α-represented by the group. We also create one group for each clause K_i corresponding to its three conditions $g_{K_i} = \{x_{v_{i1}}, x_{v_{i2}}, x_{v_{i3}}\}$ that must be α-represented in cluster 1 (i.e. $(g_{K_i}, 1) \in \mathcal{W}, (g_{K_i}, 2) \notin \mathcal{W}$). For these groups we set $\beta_{K_i} = 1$. Finally, we set $\alpha = \frac{1}{2n}$ - this ensures that any assignment of a group's data point to a cluster will satisfy the α-representation constraint (as there are $2n$ data points and thus at most $2n$ data points in a cluster). We also add a cardinality lower bound on both clusters of 1. Clearly the above scheme can be set-up in polynomial time.

We now claim that a feasible solution to the aforementioned fair assignment problem corresponds to a solution to the original 3-SAT instance. We start by taking the variable settings by looking at C_1. We start by claiming that for each variable exactly one of $x_{v_i}, x_{\bar{v}_i}$ are included in C_1. Suppose this weren't true, either both variables were included in C_1 or C_2 - however whichever cluster has neither of the variables would not be α-represented by group g_i contradicting the constraints. Since C_1 contains either x_{v_i} or $x_{\bar{v}_i}$ we set $v_i = T$ if x_{v_i} is included and $v_i = F$ otherwise. We now claim that such a setting of the variables satisfies all the clauses. Assume it did not, then there exists a clause K_i such that none of $x_{v_{i1}}, x_{v_{i2}}, x_{v_{i3}}$ are included in C_1. However, this violates the α-representation constraint for g_{K_i} providing a contradiction to the feasibility of the fair assignment problem. Note that since the feasibility problem does not consider the objective or the location of the centers the proof applies for both the k-means and k-medians setting. ■

Theorem 7 *MiniReL converges to a local optimum in finite time.*

Proof When discussing a local optimum it is important to formally define a local neighborhood for a solution. Traditionally, a clustering via Lloyd's algorithm is defined by the location of the centers (i.e. a perturbation of the centers can change the cluster assignment). This leads to settings where multiple partitions for a given set of centers need to be tested (see [8] for a discussion) to ensure local optimality. However in the MR-fairness setting a data point is not necessarily assigned to the closest center and therefore a local change to a cluster center does not change the cluster assignment. In this setting, we define a local change as any perturbation to a cluster center, and any individual change to cluster assignment (i.e. moving a data point from one cluster to another).

We start by showing that Algorithm 1 returns a local optimum. Suppose this were not the case, then there must be a local move that could improve the objective - specifically either a perturbation of a cluster center, or changing a single data point's cluster assignment. Consider the first case - a perturbation to a cluster center improves the objective. For the k-means case, it is easy to verify that for a given cluster the optimal center is the mean of every data point in the cluster by looking at first and second order optimality conditions. Consider the derivative of the loss function with respect to a cluster center c_k:

$$\frac{d}{dc_k} \sum_{x^i \in C_k} \|x^i - c_k\|_2^2 = \sum_{x^i \in C_k} 2(c_k - x^i)$$

By setting the derivative of the loss to 0, we see that $c_k = \frac{1}{n} \sum_{x^i \in C_k} x^i$ is a critical point. Furthermore, the loss function is convex confirming that this point is in fact a minimizer. Thus a local perturbation of any cluster center cannot result in a decrease to the objective. This is also true for the k-medians objective because we choose the optimal center by looking at all candidates. However, changing any single data point's cluster assignment will also not decrease the objective since we solve the fair assignment problem to optimality via integer programming. Thus the solution must be a local optimum.

It remains to show that the algorithm will converge in a finite amount of time. Similar to the proof for Lloyd's algorithm we leverage the fact that there exists a finite number of partitions of the data points. By construction at each iteration of the algorithm we decrease the objective value, and thus can never cycle through any partition multiple times as for a given partition of the data set we use the optimal cluster centers (as proven above). Thus the algorithm in the worst case can visit each partition once and thus must terminate in finite time.

■

Appendix C. Scaling MiniReL

In MiniReL, the computational bottleneck is solving the fair assignment problem which is an IP. To improve the run-time of MiniReL, we introduce two key computational approaches that reduce the number of fair assignment problems that need to be solved, and improve the speed at which they can be solved.

C.1. Warm-Starting MiniReL

To reduce the number of iterations needed to converge in MiniReL, we warm-start the initial cluster centers with the final centers of the unfair variants of Lloyd's algorithm. The key intuition behind this approach is that it allows us to leverage the polynomial time assignment problem for the majority of iterations, and only requires solving the fair assignment problem to adjust the locally optimal unfair solution to a fair one. To incorporate warm-starting into MiniReL, we replace the initial seeding of centers with centers generated from running Lloyd's algorithm.

Figure 3 shows the impact of these initialization schemes on the total computation time including time to perform the initialization. Each initialization scheme was tested on three datasets from the UCI Machine learning repository: iris, adult, and default. For each dataset we re-run MiniReL with 10 random seeds. The results show that using Lloyd's algorithm to warm-start MiniReL can lead to a large reduction in computation time, even taking into account the cost of running the initialization. However, there are diminishing returns. Namely running 100 different initialization for k-means and selecting the best leads to slightly larger overall run-times.

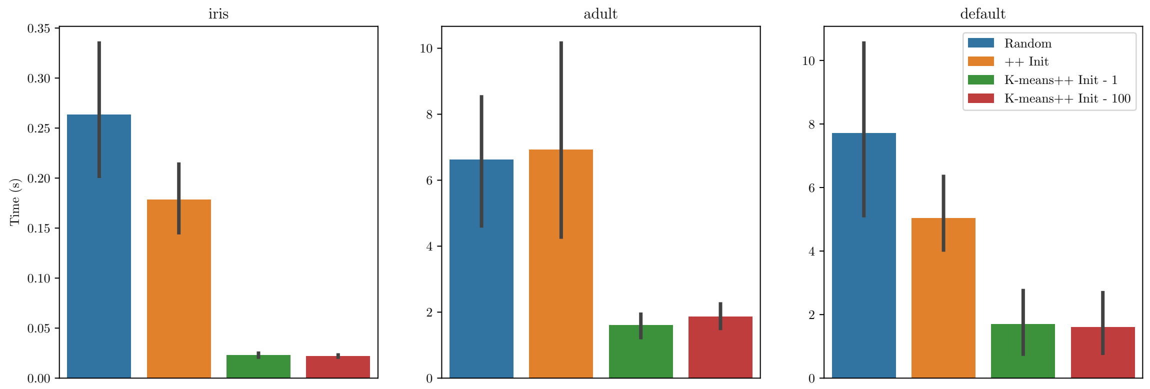

Figure 3: Average computation time in seconds over 10 random seeds for MiniReL with different initialization schemes. Bars indicate standard error.

C.2. Pre-fixing Group Assignment

One computational shortcoming of the fair assignment IP model is the use of big-M constraints which are well known to lead to weak continuous relaxations and by extension longer computation times. These constraints are needed in the IP model to track which groups are α-represented in which clusters. One approach to avoid the need for this tracking is to simply pre-fix which groups need to be represented by which clusters (i.e. akin to fixing the y_{gk} variable). This removes the need for the y variables and the associated big-M constraints and breaks some symmetry in the IP (i.e.

removes permutations of feasible cluster assignments), dramatically improving the computation time. In problems where only a single group can be α-represented by a cluster (i.e. a data point can only be part of one group and $\alpha > 0.5$), pre-fixing preserves the optimal solution to the full problem. However, in more complicated settings (i.e. multiple intersecting groups, $\alpha \leq 0.5$) pre-fixing may remove the optimal solution and thus simply represents a heuristic for improving run-time. It is worth noting that the MiniReL algorithm is itself a heuristic, and thus the pre-fixing scheme has ambiguous effects on the quality of the solution as it may cause the algorithm to converge to a better local optimum.

Given a warm-started initialization to MiniReL we formulate the problem of finding the best pre-fixing as a small integer program. Let $x_{g,k}$ be a binary variable indicating whether group g will be α-represented in cluster k. To find a good pre-fixing, we consider the myopic (i.e. one-shot) increase in cost of moving additional data points to cluster k to meet the fairness constraints. For a cluster k, let q be the additional number of points from group g needed for g to be α-represented in this cluster (i.e. $q + p_{gk}|C_k| \geq \alpha(q + |C_k|)$). Let $c_{(x)} = \text{argmin}_{c \in \{c_1,\ldots,c_K\}}\{D(x,c)\}$ be the closest center for $x \in \mathcal{X}$. The myopic cost c_{gk} associated with α-representing group g in cluster k then becomes:

$$c_{gk} = \min_{X \subset X_g \setminus C_K : |X| = q} \sum_{x \in X} \big(D(x, c_k) - D(x, c_{(x)})\big)$$

where $q = \lceil \frac{\max(0, \alpha - p_{gk})}{1-\alpha} |C_k| \rceil$. We can now formulate an IP to perform the pre-fix assignment as follows:

$$\textbf{min} \quad \sum_{(g,k) \in \mathcal{W}} c_{gk} x_{gk} \tag{8}$$

$$\textbf{s.t.} \quad \sum_{k \in \mathcal{K} : (g,k) \in \mathcal{W}} x_{gk} \geq \beta_g \qquad \forall g \in \mathcal{G} \tag{9}$$

$$\sum_{g \in \mathcal{G}} x_{gk} \leq \left\lfloor \frac{1}{\alpha} \right\rfloor \qquad \forall k \in \mathcal{K} \tag{10}$$

$$x_{gk} \in \{0, 1\} \qquad \forall (g,k) \in \mathcal{W} \tag{11}$$

The objective (8) is simply to minimize the cost of pre-fixing. Constraint (9) ensures enough clusters are allocated to each group to meet the MR-fairness constraint. Finally constraint (10) ensures no cluster is assigned more groups than can simultaneously be α-represented by it. Note that there are only $\mathcal{W} \leq |\mathcal{K}| \times |\mathcal{G}|$ variables in this IP. In practice we found that solving this IP took under 1 second for all instances tested in this paper. The resulting speedup is very significant when compared to running MiniReL with the full IP model for fair assignment, at times leading to over a 400x speedup in some instances as shown in Figure 4.

Appendix D. Computation Time

Table 1 shows the mean computation time (and standard deviation) for both MiniReL and K-means over 10 random seeds. As expected, MiniReL has a higher computation time than k-means. However, it still solves all instances in under 40 seconds, showing that the approach is still feasible for practical instances.

Table 1: Computation time in seconds over 10 random seeds (standard deviation in parentheses). Final row includes average computation time over all random seeds and setting of K for each dataset. K-means columns correspond to Lloyd's algorithm.

	adult		default		voting	
K	K-means	MiniReL	K-means	MiniReL	K-means	MiniReL
4	0.6 (0.03)	13.4 (0.25)	0.3 (0.02)	10.8 (0.18)	0.8 (0.04)	2.1 (0.03)
5	0.7 (0.04)	14.3 (0.37)	0.4 (0.03)	11.2 (0.14)	0.7 (0.02)	2.4 (0.04)
6	0.8 (0.04)	15.8 (1.08)	0.5 (0.05)	12.0 (0.33)	0.7 (0.03)	3.4 (0.04)
7	0.9 (0.04)	15.8 (1.03)	0.6 (0.05)	12.3 (0.35)	1.1 (0.07)	3.5 (0.08)
8	1.0 (0.05)	17.2 (0.54)	0.8 (0.09)	13.3 (0.29)	1.0 (0.09)	3.8 (0.06)
9	1.1 (0.06)	17.8 (0.91)	0.8 (0.08)	13.5 (0.37)	1.1 (0.08)	4.5 (0.18)
10	1.3 (0.04)	19.1 (0.9)	0.9 (0.07)	14.8 (0.29)	1.4 (0.07)	5.0 (0.12)
11	1.4 (0.04)	19.6 (1.16)	1.0 (0.06)	14.4 (0.3)	1.5 (0.12)	5.4 (0.21)
12	1.4 (0.04)	21.4 (3.77)	1.1 (0.11)	15.6 (0.55)	1.7 (0.18)	6.1 (0.3)
13	1.5 (0.05)	20.9 (3.01)	1.3 (0.18)	16.0 (0.36)	2.2 (0.19)	24.3 (0.32)
14	1.7 (0.07)	22.8 (3.02)	1.5 (0.21)	16.7 (0.35)	2.4 (0.18)	34.7 (0.3)
15	1.8 (0.08)	21.4 (1.7)	1.5 (0.11)	17.5 (1.68)	2.5 (0.20)	38.1 (0.41)
mean	1.2 (0.39)	18.3 (3.43)	0.9 (0.41)	14.0 (2.17)	1.3 (0.58)	8.7 (10.64)

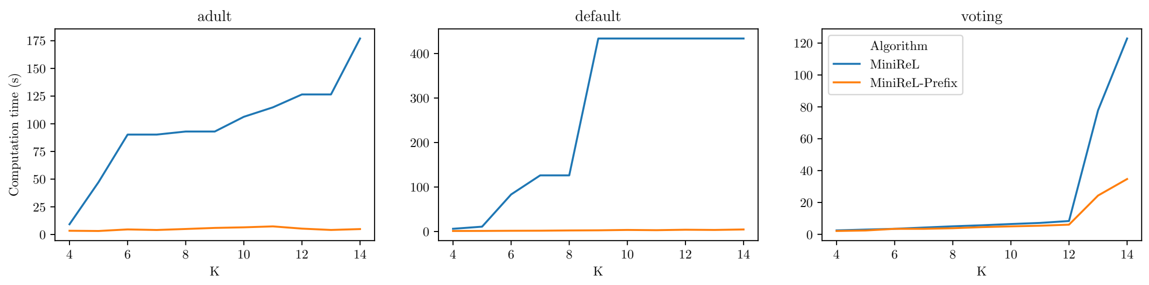

Figure 4: Impact of pre-fixing on runtime of the MiniReL Algorithm.