Scalable Fair Clustering

Arturs Backurs [1 2] **Piotr Indyk** [1 3] **Krzysztof Onak** [1 4] **Baruch Schieber** [1 5] **Ali Vakilian** [1 3] **Tal Wagner** [1 3]

Abstract

We study the fair variant of the classic k-median problem introduced by Chierichetti et al. (Chierichetti et al., 2017) in which the points are colored, and the goal is to minimize the same average distance objective as in the standard k-median problem while ensuring that all clusters have an "approximately equal" number of points of each color. Chierichetti et al. proposed a two-phase algorithm for fair k-clustering. In the first step, the pointset is partitioned into subsets called fairlets that satisfy the fairness requirement and approximately preserve the k-median objective. In the second step, fairlets are merged into k clusters by one of the existing k-median algorithms. The running time of this algorithm is dominated by the first step, which takes super-quadratic time. In this paper, we present a practical approximate fairlet decomposition algorithm that runs in *nearly linear* time.

1. Introduction

The success of machine learning led to its widespread adoption in many aspects of our daily lives. Automatic prediction and forecasting methods are now used to approve mortgage applications or estimate the likelihood of recidivism (Chouldechova, 2017). It is thus crucial to design machine learning algorithms that are *fair*, i.e., do not suffer from bias against or towards particular population groups. An extensive amount of research over the last few years has focused on two key questions: how to formalize the notion of fairness in the context of common machine learning tasks, and how to design efficient algorithms that conform to those formalizations. See e.g., the survey by Chouldechova and Roth for an overview (Chouldechova & Roth, 2018).

In this paper we focus on the second aspect. Specifically, we consider the problem of *fair clustering* and propose efficient algorithms for solving this problem. Fair clustering, introduced in (Chierichetti et al., 2017), generalizes the standard notion of clustering by imposing a constraint that all clusters must be *balanced* with respect to specific sensitive attributes, such as gender or religion. In the simplest formulation, each input point is augmented with one of two colors (say, red and blue), and the goal is to cluster the data while ensuring that, in each cluster, the fraction of points with the less frequent color is bounded from below by some parameter strictly greater than 0. Chierichetti et al. proposed polynomial time approximation algorithms for fair variants of classic clustering methods, such as k-center (minimize the *maximum* distance between points and their cluster centers) and k-median (minimize the *average* distance between points and their cluster centers). To this end, they introduced the notion of *fairlet decomposition*: a partitioning of the input pointset into small subsets, called *fairlets*, such that a good balanced clustering can be obtained by merging fairlets into clusters. Unfortunately, their algorithm for computing a fairlet decomposition has running time that is at least *quadratic* in the number of the input points. As a result, the algorithm is applicable only to relatively small data sets.

In this paper we address this drawback and propose an algorithm for computing fairlet decompositions with running time that is *near-linear* in the data size. We focus on the k-median formulation, as k-center clustering is known to be sensitive to outliers. Our algorithms apply to the typical case where the set of input points lie in a d-dimensional space, and the distance is induced by the Euclidean norm.[1]

To state the result formally, we need to introduce some notation. Consider a collection of n points $P \subseteq \mathbb{R}^d$, where each point $p \in P$ is colored either *red* or *blue*. For a subset of points $S \subseteq P$, the *balance* of S is defined as $\mathbf{balance}(S) := \min\{\frac{|S_r|}{|S_b|}, \frac{|S_b|}{|S_r|}\}$ where S_r and S_b respectively denote the subset of red and blue points in S. Assuming $b < r$, a clustering $\mathcal{C} = \{C_1 \dots C_k\}$ of P is *(r, b)-fair* if for every cluster $C \in \mathcal{C}$, $\mathbf{balance}(C) \geq \frac{b}{r}$. In k-median clustering, the goal is to find k centers and partition the

[1]Authors ordered alphabetically. [2]TTIC, Chicago, IL, USA [3]CSAIL, MIT, Cambridge, MA, USA [4]IBM T. J. Watson, Yorktown Heights, NY, USA [5]Department of Computer Science, New Jersey Institute of Technology, Newark, NJ, USA. Correspondence to: Ali Vakilian <vakilian@mit.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

[1]E.g., all data sets used to evaluate the algorithms in (Chierichetti et al., 2017) fall into this category.

pointset P into k clusters centered at the selected centers such that the sum of the distances from each point $p \in P$ point to its assigned center (i.e., the center of the cluster to which p belongs) is minimized. In the (r, b)-fair k-median problem, all clusters are required to have balance at least $\frac{b}{r}$. Our main result is summarized in the following theorem.

Theorem 1.1. *Let $T(n, d, k)$ be the running time of an α-approximation algorithms for the k-median problem over n points in $\mathbb{R}^d$. Then there exists an $O(d \cdot n \cdot \log n + T(n, d, k))$-time algorithm that given a point set $P \subseteq \mathbb{R}^d$ and balance parameters (r, b), computes a (r, b)-fair k-median of P whose cost is within a factor of $O_{r,b}(d \cdot \log n + \alpha)$ from the optimal cost of (r, b)-fair k-median of P.*

The running time can be reduced further by applying dimensionality reduction techniques, see, e.g., (Makarychev et al., 2018) and the references therein.

We complement our theoretical analysis with empirical evaluation. Our experiments show that the quality of the clustering obtained by our algorithm is comparable to that of (Chierichetti et al., 2017). At the same time, the empirical runtime of our algorithm scales almost linearly in the number of points, making it applicable to massive data sets (see Figure 1).

Related work. Since the original paper of (Chierichetti et al., 2017), there has been several followup works studying fair clustering. In particular, (Rösner & Schmidt, 2018) and (Bercea et al., 2018) studied the fair variant of k-center clustering (as opposed to k-median in our case). Furthermore, the latter paper presented a "bi-criteria" approximation algorithm for k-median and k-means under a somewhat different notion of fairness. However, their solution relies on a linear program that is a relaxation of an integer linear program with at least n^2 variables, one for every pair of points. Thus, their algorithm does not scale well with the input size. Another algorithm proposed in (Bera et al., 2019), requires solving a linear program with nk variables. Due to the special structure of the LP it is plausible that it can be solved efficiently, but we are not aware of any empirical evaluation of this approach.

The work most relevant to our paper is a recent manuscript by Schmidt et al. (Schmidt et al., 2018), which proposed efficient streaming algorithms for fair k-means (which is similar to k-median studied here). Specifically, they give a near-linear time streaming algorithm for computing a *core-set*: a small subset $S \subseteq P$ such that solving fair clustering over S yields an approximate solution for the original pointset P. In order to compute the final clustering, however, they still need to apply a fair clustering algorithm to the core-set. Thus, our approach is complementary to the core-set approach, and the two can be combined to yield algorithms which are both fast and space-efficient[2].

We note that the above algorithms guarantee constant approximation factors, as opposed to the logarithmic factor in our paper. As we show in the experimental section, this does not seem to affect the empirical quality of solutions produced by our algorithm. Still, designing a constant factor algorithm with a near-linear running time is an interesting open problem.

Possible settings of (r, b). (Chierichetti et al., 2017) gave (r, b)-fairlet decomposition algorithms only for $b = 1$. This does not allow for computing a full decomposition of the pointset into well-balanced fairlets if the numbers of red and blue points are close but not equal (for instance, if their ratio is 9:10). One way to address this could be to downsample the larger set in order to make them have the same cardinality and then compute a $(1, 1)$-fairlet decomposition. The advantage of our approach is that we do not disregard any, even random, part of the input. This may potentially lead to much better solutions, partially by allowing that the clusters are not ideally balanced. The general settings of r and b are also considered by Bercea et al. and Bera et al. (Bercea et al., 2018; Bera et al., 2019).

Our techniques. Our main contribution is to design a nearly-linear time algorithm for (r, b)-fairlet decomposition for any integer values of r, b. Our algorithm has two steps. First, it embeds the input points into a tree metric called *HST* (intuitively, this is done by computing a quadtree decomposition of the point set, and then using the distances in the quadtree). In the second step it solves the fairlet decomposition problem with respect to the new distance function induced by HST. The distortion of the embedding into the HST accounts for the $\log n$ factor in the approximation guarantee.

Once we have the HST representation of the pointset, the high-level goal is to construct "local" (r, b)-fairlets with respect to the tree. To this end, the algorithm scans the tree in a top-down order. In each node v of the tree, it greedily partitions the points into fairlets so that the number of fairlets whose points belong to subtrees rooted at different children of v is minimized. In particular, we prove that minimizing the number of such fairlets (which we refer to as the *Minimum Heavy Point* problem) leads to an $O(1)$-approximate (r, b)-fairlet decomposition with respect to the distance over the tree.

[2] We note, however, that since core-sets typically require assigning weights to data points, such combination requires extending the clustering algorithm to weighted pointsets. In this paper we do not consider the weighted case.

2. Preliminaries

Definition 2.1 (Fairlet Decomposition). *Suppose that $P \subseteq Y$ is a collection of points such that each is either colored* red *or* blue*. Moreover, suppose that* $\mathbf{balance}(P) \geq \frac{b}{r}$ *for some integers* $1 \leq b \leq r$ *such that* $\gcd(r, b) = 1$*. A clustering* $\mathcal{X} = \{D_1, \cdots, D_m\}$ *of* P *is an* (r, b)-fairlet decomposition *if* (a) *each point* $p \in P$ *belongs to exactly one cluster* $D_i \in \mathcal{X}$*,* (b) *for each* $D_i \in \mathcal{Y}$*,* $|D_i| \leq b + r$*, and* (c) *for each* $D_i \in \mathcal{X}$*,* $\mathbf{balance}(D_i) \geq \frac{b}{r}$*.*

Probabilistic metric embedding. A probabilistic metric $(X, \overline{d})$ is defined as a set of ℓ metrics $(X, d_1), \cdots, (X, d_\ell)$ along with a probability distribution of support size ℓ denoted by $\alpha_1, \cdots, \alpha_\ell$ such that $\overline{d}(p, q) = \sum_{i=1}^{\ell} \alpha_i \cdot d_i(p, q)$. For any finite metric $M = (Y, d)$ and probabilistic metric $(X, \overline{d})$, an *embedding* $f : Y \to X$ has distortion c_f, if:

- for all $p, q \in Y$ and $i \leq k$, $d_i(f(p), f(q)) \geq d(p, q)$,
- $\overline{d}(f(p), f(q)) \leq c_f \cdot d(p, q)$.

Definition 2.2 (γ-HST). *A tree T rooted at vertex r is a* hierarchically well-separated tree (γ-HST) *if all edges of T have non-negative weights and the following two conditions hold:*

1. *The (weighted) distances from any node to all its children are the same.*
2. *For each node $v \in V \setminus \{r\}$, the distance of v to its children is at most $1/\gamma$ times the distance of v to its parent.*

We build on the following result due to Bartal (Bartal, 1996), which gives a probabilistic embedding from $\mathbb{R}^d$ to a γ-HST. Our algorithm explicitly computes this embedding which we describe in more detail in the next section. For a proof of its properties refer to (Indyk, 2001) and the references therein. In this paper, we assume that the given pointset has $\mathrm{poly}(n)$ *aspect ratio* (i.e. the ratio between the maximum and minimum distance is $\mathrm{poly}(n)$).

Theorem 2.3 ((Bartal, 1996)). *Any finite metric space M on points in $\mathbb{R}^d$ can be embedded into probabilistic metric over γ-HST metrics with $O(\gamma \cdot d \cdot \log_\gamma n)$ distortion in $O(d \cdot n \cdot \log_\gamma n)$ time.*

3. High-level Description of Our Algorithm

Our algorithm for (r, b)-fair k-median problem in Euclidean space follows the high-level approach of (Chierichetti et al., 2017): it first computes an approximately optimal (r, b)-fairlet decomposition for the input point set P (see Algorithm 1). Then, in the second phase, it clusters the (r, b)-fairlets produced in the first phase into k clusters (see Algorithm 2). Our main contribution is designing a *scalable* algorithm for the first phase of this approach, namely *(r, b)-fairlet decomposition*.

Preprocessing phase: embedding to γ-HST. An important step in our algorithm is to embed the input pointset P into a γ-HST (see Section 2 for more details on HST metrics). To this end, we exploit the following standard construction of γ-HST using *randomly shifted grids*.

Suppose that all points in P lie in $\{-\Delta, \cdots, \Delta\}^d$. We generate a random tree T (which is a γ-HST embedding of P) recursively. We translate the d-dimensional hypercube $H = [-2\Delta, 2\Delta]^d$ via a uniformly random shift vector $\sigma \in \{-\Delta, \cdots, \Delta\}^d$. It is straightforward to verify that all points in P are enclosed in $H + \sigma$. We then split each dimension of H into γ equal pieces to create a grid with γ^d cells. Then we proceed recursively with each non-empty cell to create a hierarchy of nested d-dimensional grids with $O(\log_\gamma \frac{\Delta}{\varepsilon})$ levels (each cell in the final level of the recursion either contains exactly one point of P or has side length ε). Next, we construct a tree T corresponding to the described hierarchy nested d-dimensional grids as follows. Consider a cell C in the i-th level (level 0 denote the initial hypercube H) of the hierarchy. Let $T_C^1, \cdots, T_C^\ell$ denote the trees constructed recursively for each non-empty cells of C. Denote the root of each tree T_C^j by u_C^j. Then we connect u_C (corresponding to cell C) to each of u_j^C with an edge of length proportional to the diameter of C (i.e., $(\sqrt{d} \cdot \Delta)/\gamma^i$).

Note that the final tree generated by the above construction is a γ-HST: on each path from the root to a leaf, the length of consecutive edges decrease exponentially (by a factor of γ) and the distance from any node to all of its children are the same. Moreover, we assume that $\Delta/\varepsilon = n^{O(1)}$.

Phase 1: computing (r, b)-fairlet decomposition. This phase operates on the probabilistic embedding of the input into a γ-HST T from the preprocessing phase, where $\gamma = \mathrm{poly}(r, b)$. The distortion of the embedding is $O(d \cdot \gamma \cdot \log_\gamma n)$. Additionally, we augment each node $v \in T$ with integers N_r and N_b denoting the number of *red* and *blue* points, respectively, in the subtree $T(v)$ rooted at v.

Step 1. Compute an *approximately minimum* number of points that are required to be removed from the children of v so that (1) the set of points contained by each child becomes (r, b)-balanced, and (2) the union of the set of removed points is also (r, b)-balanced. More formally, we solve Question 3.2 approximately (recall that for each child v_i, N_r^i and N_b^i respectively denotes the number of red and blue points in $T(v_i)$).

Definition 3.1 (Heavy Point). *A point $p \in T(v)$ is* heavy *with respect to v if it belongs to a fairlet D such that $\mathrm{lca}(D) = v$. For each fairlet $D \in \mathcal{X}$, $\mathrm{lca}(D)$ denotes*

Algorithm 1 FAIRLETDECOMPOSITION(v, r, b): returns an (r, b)-fairlet decomposition of the points in $T(v)$

if v is a leaf node of T **then**
  return an arbitrary (r, b)-fairlet decomposition of the points in $T(v)$
end if
{**Step 1:** approximately minimize the total number of heavy points with respect to v}
$\{x_r^i, x_b^i\}_i \leftarrow$ MINHEAVYPOINTS$(\{N_r^i, N_b^i\}_{i\in[\gamma^d]}, r, b)$ {for non-empty children $i \in [\gamma^d]$ of v}
{**Step 2:** find an (r, b)-fairlet decomposition of heavy points with respect to v}
$P_v \leftarrow \emptyset$
for all non-empty children $i \in [\gamma^d]$ of v **do**
  remove an arbitrary set of x_r^i red and x_b^i blue points from $T(v_i)$ and **add** them to P_v
end for
output an (r, b)-fairlet decomposition of P_v
{**Step 3:** proceed to the children of v}
for all non-empty children $i \in [\gamma^d]$ of v **do**
  FAIRLETDECOMPOSITION(v_i, r, b)
end for

the least common ancestor (lca) *of the points contained in D in T.*

Question 3.2 (Minimum Heavy Points Problem). *Suppose that v is a node in T. For each $i \in [\gamma^d]$ corresponding to non-empty children of v, let x_r^i, x_b^i be respectively the number of red and blue points that are removed from $T(v_i)$. The goal is to minimize $\sum_{i=1}^{\gamma^d} x_r^i + x_b^i$ such that the following conditions hold:*

1. *for each $i \in [\gamma^d]$, $(N_r^i - x_r^i, N_b^i - x_b^i)$ is (r, b)-balanced.*
2. *$(\sum_{i\in[\gamma^d]} x_r^i, \sum_{i\in[\gamma^d]} x_b^i)$ is (r, b)-balanced.*

Step 2. After computing $\{x_r^i, x_b^i\}_{i\in[\gamma^d]}$, for each $i \in [\gamma^d]$, remove an *arbitrary* set of x_r^i red and x_b^i blue points from $T(v_i)$ and add them to P_v. Then, output an arbitrary (r, b)-fairlet decomposition of points P_v which is guaranteed to be (r, b)-balanced by Step 1.

Step 3. For each $i \in [\gamma^d]$ corresponding to a non-empty child of v, run FAIRLETDECOMPOSITION(v_i, r, b) which is guaranteed to be (r, b)-balanced by Step 1.

Here is the main guarantee of our approach in the first step (i.e., (r, b)-fairlet decomposition).

Theorem 3.3. *There exists an $O(d \cdot n \cdot \log_\gamma n)$ time algorithm that given a point set $P \subseteq \mathbb{R}^d$ and balance parameters (r, b), computes an (r, b)-fairlet decomposition of P with respect to $\mathrm{cost}_{\mathrm{median}}$ whose expected cost is within $O(d \cdot (r^8 + b^8) \cdot \log n)$ factor of the optimal (r, b)-fairlet decomposition of P in expectation.*

Phase 2: merging (r, b)-fairlets into k clusters. In this phase, we essentially follow the same approach as (Chierichetti et al., 2017).

Algorithm 2 CLUSTERFAIRLET(Q): the algorithm returns an (r, b)-fair k-median of P given an (r, b)-fairlet decomposition Q of P

for all fairlet $q_i \in Q$ **do**
  let an arbitrary point $c_i \in q_i$ be the center of q_i
  add $|q_i|$ copies of c_i to $\overline{P}$
end for
$\mathcal{C} \leftarrow \beta$-approximate k-median clustering of $\overline{P}$
$\mathcal{C}^* \leftarrow \{\bigcup_{j:c_j\in C_i} q_j\}_{i=1}^k$ {each fairlet joins the cluster of its center in $\mathcal{C}$}
return $\mathcal{C}^*$

Theorem 3.4 (Fairlet to Fair Clustering). *Suppose that Q is an α-approximate (r, b)-fairlet decomposition of P. Then,* CLUSTERFAIRLET*(Q) returns an $(\alpha + (r + b) \cdot \beta)$-approximate (r, b)-fair k-median clustering of P where β denotes the approximation guarantee of the k-median algorithm invoked in* CLUSTERFAIRLET.

Finally, Theorem 3.3 and 3.4 together imply Theorem 1.1.

4. Fairlet Decomposition: a Top-down Approach on γ-HST

In this section, we provide a complete description of the first phase in our (r, b)-fair k-median algorithm (described in Section 3), namely our scalable (r, b)-fairlet decomposition algorithm.

The first step in our algorithm is to embed the input point set into a γ-HST (for a value of γ to be determined later in this section). Once we build a γ-HST embedding T of the input points $P \subseteq \mathbb{R}^d$, the question is how to partition the points into (r, b)-fairlets. We assume that each node $v \in T$ is augmented with extra information N_r and N_b respectively denoting the number of *red* and *blue* points in the subtree $T(v)$ rooted at v.

To compute the total cost of a fairlet decomposition, it is important to specify the clustering cost model (e.g., k-median, k-center). Here, we define $\mathrm{cost}_{\mathrm{median}}$ to denote the cost of a fairlet (or cluster) with respect to the cost function of k-median clustering: for any subset of points $S \subset \mathbb{R}^d$, $\mathrm{cost}_{\mathrm{median}}(S) := \min_{p\in S} \sum_{q\in S} d(p, q)$ where $d(p, q)$ denotes the distance of p, q in T.

In this section, we design a fast fairlet decomposition algorithm with respect to $\mathrm{cost}_{\mathrm{median}}$.

Theorem 4.1. *There exists an $\widetilde{O}(n)$ time algorithm that given an $O(r^5+b^5)$-HST embedding T of the point set P and balance parameters (r,b), computes an $O(r^3+b^3)$-approximate (r,b)-fairlet decomposition of P with respect to $\mathrm{cost}_{\mathrm{median}}$ on T.*

Thus, by Theorem 4.1 and the bound on the expected distortion of embeddings into HST metrics (Theorem 2.3), we can prove Theorem 3.3.

Proof of **Theorem 3.3.** We first embed the points into an $O(r^5+b^5)$-HST T and then perform the algorithm guaranteed in Theorem 4.1. By Theorem 2.3, the expected distortion of our embedding to T is $O(d\cdot\gamma\cdot\log_\gamma n)$ and by Theorem 4.1, there exists an algorithm that computes an $O(r^3+b^3)$-approximate fairlet-decomposition of P with respect to distances in T. Hence, the overall algorithm achieves $O(d\cdot(r^8+b^8)\cdot\log n)$-approximation.

Since the embedding T can be constructed in time $O(d\cdot n\cdot\log n)$ and the fairlet-decomposition algorithm of Theorem 4.1 runs in near-linear time, the overall algorithm also runs in $\widetilde{O}(n)$. □

Before describing the algorithm promised in Theorem 4.1, we define a modified cost function $\mathrm{cost}_{\mathrm{med}}$ which is a simplified variant of $\mathrm{cost}_{\mathrm{median}}$ and is particularly useful for computing the cost over trees. Consider a γ-HST embedding of P denoted by T and assume that $\mathcal{X}$ is a fairlet decomposition of P. Moreover, $h(D)$ denotes the height of $\mathrm{lca}(D)$ in T. For each fairlet D, $\mathrm{cost}_{\mathrm{med}}(D)$ is defined as

$$\mathrm{cost}_{\mathrm{med}}(D) := \sum_{q\in D} d(\mathrm{lca}(D),q) = \Theta(|D|\cdot\gamma^{h(\mathrm{lca}(D))}). \tag{1}$$

In particular, $\mathrm{cost}_{\mathrm{med}}(D)$ relaxes the task of finding "the best center in fairlets" and at the same time, its value is within a small factor of $\mathrm{cost}_{\mathrm{median}}(D)$.

Claim 4.2. *Suppose that T is a γ-HST embedding of the set of points P. For any (r,b)-fairlet S of P, $\mathrm{cost}_{\mathrm{median}}(S)\le\mathrm{cost}_{\mathrm{med}}(S)\le(r+b)\cdot\mathrm{cost}_{\mathrm{median}}(S)$ where the distance function is defined w.r.t. T.*

In the rest of this section we prove the following result which together with Claim 4.2 imply Theorem 4.1.

Lemma 4.3. *There exists a near-linear time algorithm that given an $O(r^5+b^5)$-HST embedding T of the point set P and balance parameters (r,b), computes an $O(r^2+b^2)$-approximate (r,b)-fairlet decomposition of P with respect to $\mathrm{cost}_{\mathrm{med}}$ on T.*

Lemma 4.4. *For any tree T with the root vertex v, the number of heavy points with respect to v in the (r,b)-fairlet decomposition constructed by* MINHEAVYPOINTS(v,r,b) *is at most $O(r^2+b^2)$ times the minimum number of heavy points in any valid (r,b)-fairlet decomposition of T.*

4.1. Description of Step 1: Minimizing the Number of Heavy Points

In this section, we show that MINHEAVYPOINTS algorithm invoked by FAIRLETDECOMPOSITION finds an $O(r^2+b^2)$-approximate solution of *Minimum Heavy Points* problem.

The high-level overview of MINHEAVYPOINTS is as follows. For any subset of points $D\subseteq P$, we can compute in $O(1)$ what the maximal size (r,b)-balanced subset of D is: w.l.o.g. suppose that $N_r\ge N_b$ and $r\ge b$. If $N_r\le\frac{r}{b}\cdot N_b$, the collection is (r,b)-balanced. Otherwise, it suffices to greedily pick maximal size (r,b)-fairlets (see procedure UNBALANCEDPOINTS for the formal algorithm). This simple observation implies a lower bound on the size of any optimal solution of *Heavy Points Minimization* with respect to v and we use this value to bound the approximation guarantee of MINHEAVYPOINTS algorithm.

Claim 4.5. UNBALANCEDPOINTS(N_r,N_b,r,b) *correctly computes the minimum number of points that is required to be removed from $N_r\cup N_b$ so that the remaining points become (r,b)-balanced. Moreover, the solution returned by the procedure only removes points form a single color class.*

For each $i\in[\gamma^d]$, let $(\tilde{x}_r^i,\tilde{x}_b^i)$ be the output of UNBALANCEDPOINTS(N_r^i,N_b^i,r,b).

Corollary 4.6. *Any (r,b)-fairlet decomposition of the points in $T(v)$ has at least $\sum_{i\in[\gamma^d]}\tilde{x}_r^i+\tilde{x}_b^i$ heavy points.*

Stage 1: minimum number of heavy points. If $\sum_{i\in[\gamma^d]}\tilde{x}_r^i$ red points together with $\sum_{i\in[\gamma^d]}\tilde{x}_b^i$ blue points form an (r,b)-balanced collection, then MINHEAVYPOINTS technically terminates at the end of stage 1 and the solution returned by MINHEAVYPOINTS achieves the minimum possible number of heavy points. However, in general, the collection with $\sum_{i\in[\gamma^d]}\tilde{x}_r^i$ red points and $\sum_{i\in[\gamma^d]}\tilde{x}_b^i$ may not form an (r,b)-balanced collection. Next, we show that we can always pick at most $rb(\sum_{i\in[\gamma^d]}\tilde{x}_r^i+\tilde{x}_b^i)$ additional heavy points and keep both all subtrees rooted at children of v and the set of heavy points (r,b)-balanced.

Another structure we will refer to in the rest of this section is *saturated (r,b)-fairlets*. A fairlet D is a saturated (r,b)-fairlet if it has *exactly* $r+b$ points; r points from color c and b points from color $\bar{c}$.

Stage 2: Adding free points. If the "must-have" heavy points are not (r,b)-balanced, then one color is *dominant*. For a color class $c\in\{r,b\}$, a collection of points S is c-dominant if $|S_c|\ge\frac{r}{b}\cdot|S_{\bar{c}}|$. Moreover, the collection is *minimally-balanced* c-dominant if S is (r,b)-balanced but it will be no longer (r,b)-balanced even if we remove a single point of color $\bar{c}$.

Let c be the dominant color in the heavy points. Then, we

Algorithm 3 MINHEAVYPOINTS($\{N_r^i, N_b^i\}_{i\in[\gamma^d]}, r, b, \gamma$)

{Stage 1: lower bound on the number of heavy points}
for all non-empty children $i \in [\gamma^d]$ of v **do**
  $(x_r^i, x_b^i) \leftarrow$ UNBALANCEDPOINTS(N_r^i, N_b^i, r, b)
end for
$y_r \leftarrow \sum_{i\in[\gamma^d]} x_r^i, y_b \leftarrow \sum_{i\in[\gamma^d]} x_b^i$
$c_{\text{dom}} \leftarrow \text{argmax}_{c\in\{r,b\}} y_c, \overline{c_{\text{dom}}} \leftarrow \{r, b\} \setminus c_{\text{dom}}$
{Stage 2: add free points with color $\overline{c_{\text{dom}}}$}
if $(\sum_{i\in[\gamma^d]} x_r^i, \sum_{i\in[\gamma^d]} x_b^i)$ is (r, b)-balanced **then**
  break
end if
for all non-empty children $i \in [\gamma^d]$ of v **do**
  $x^i_{\overline{c_{\text{dom}}}} \leftarrow x^i_{\overline{c_{\text{dom}}}} +$
  $\max($EXTRAPOINT$(\overline{c_{\text{dom}}}, N_r^i - x_r^i, N_b^i - x_b^i, r, b),$
  $y_{c_{\text{dom}}} - y_{\overline{c_{\text{dom}}}})$
end for
{Stage 3: add points of non-saturated (r, b)-fairlets}
if $(\sum_{i\in[\gamma^d]} x_r^i, \sum_{i\in[\gamma^d]} x_b^i)$ is (r, b)-balanced **then**
  break
end if
for all non-empty children $i \in [\gamma^d]$ of v **do**
  $(n_r, n_b) \leftarrow$
  NONSATFAIRLET$(N_r^i - x_r^i, N_b^i - x_b^i, r, b)$
  $x_r^i \leftarrow x_r^i + n_r, x_b^i \leftarrow x_b^i + n_b$
end for
return $(\{x_r^i, x_b^i\}_{i\in[\gamma^d]})$

Algorithm 4 UNBALANCEDPOINTS(N_r, N_b, r, b): returns the minimum number of points that are required to be removed so that (N_r, N_b) become (r, b)-balanced.

if $N_r \geq N_b$ **then**
  return $(N_r - \lfloor N_b \cdot \frac{r}{b} \rfloor, 0)$
end if
return $(0, N_b - \lfloor N_r \cdot \frac{r}{b} \rfloor)$

inspect all children of v and if there exits a child in which $\overline{c}$ is dominant, we borrow as many points of color $\overline{c}$ as we can (we need to keep the subtree (r, b)-balanced, see EXTRAPOINT procedure) till either the set of heavy points becomes (r, b)-balanced or all subtrees rooted at children of v become minimally-balanced c-dominant. It is straightforward to show that at most $\frac{b}{r} \cdot |S_c|$ points of color $\overline{c}$ will be borrowed from the children of v in this phase.

Lemma 4.7. *Suppose that the set of heavy points is c-dominant. If the set of heavy points is not (r, b)-balanced at the end of stage 2, then for each $i \in [\gamma^d]$, the set of points in the subtree rooted at v_i is minimally-balanced c-dominant.*

Corollary 4.8. *Suppose that the set of heavy points is c-dominant. If the set of heavy points is not (r, b)-balanced at the end of stage 2, then for each $i \in [\gamma^d]$, the set of points in the subtree rooted at v_i have an (r, b)-fairlet decomposition with at most one non-saturated (r, b)-fairlet.*

Algorithm 5 EXTRAPOINT(c, N_r, N_b, r, b): returns the maximum number of points of color c that can be removed from the set (N_r, N_b) such that they remain (r, b)-balanced.

if $N_c \leq N_{\overline{c}}$ **then**
  return 0
end if
return $\lceil N_{\overline{c}} \cdot \frac{b}{r} \rceil$

Stage 3: Non-saturated fairlets. Here, we show that we can increase the number of heavy points by at most a factor of $O(rb)$ and make both the set of heavy points and the set of points in all subtrees rooted at children of v (r, b)-balanced. Let ns denote the total number of non-saturated fairlets in the subtree rooted at v. We consider two cases depending on the value of ns and the total number of heavy points N_{H} that do not belong to any saturated fairlets (in particular, $|N_{\text{H}}| \leq \sum_{i\in[\gamma^d]} \tilde{x}_r^i + \tilde{x}_b^i$):

Case 1: $ns \leq b \cdot N_{\text{H}}$. If we add all non-saturated fairlets, since the rest of fairlets in the subtree rooted at children of v are saturated (r, b)-balanced, then this "extended" collection of heavy points has to be (r, b)-balanced. Otherwise, the whole data set itself is not (r, b)-balanced which is a contradiction. Moreover, the total number of heavy points is at most $ns \times (r + b) = O(rb \cdot N_{\text{H}})$

Case 2: $ns > b \cdot N_{\text{H}}$. Here we show that after adding at most $b \cdot N_{\text{H}}$ non-saturated (r, b)-fairlets, the set of heavy points becomes (r, b)-balanced. Let r_i and b_i $(r_i \geq b_i)$ specify the size of the non-saturated fairlet that belongs to the i-th child of v. Note that since all fairlets are c-dominant, r_i denotes the number of points of color c.

Moreover, in any non-saturated (r, b)-fairlet, $\frac{r_i}{b_i} < \frac{r}{b}$, which implies that $r_i b \leq r b_i - 1$. Let Q denote the set of children of v whose non-saturated fairlets are picked. After adding all points in these non-saturated fairlets,

$$\begin{aligned}
\#\text{points of color } c &\leq N_{\text{H}} + \sum_{j\in Q} r_j \\
&\leq N_{\text{H}} + \sum_{j\in Q} (\frac{r}{b} \cdot b_j - \frac{1}{b}) \\
&\leq \frac{r}{b} \sum_{j\in Q} b_j \triangleright \text{since } |Q| = b \cdot N_{\text{H}}, \quad (2) \\
\#\text{points of color } \overline{c} &\geq \frac{r}{b} \sum_{j\in Q} b_j. \quad (3)
\end{aligned}$$

Moreover, since in the beginning of the process the number of points of color c is more than the number of points of color $\overline{c}$ and also in each non-saturated fairlest the number of

points of color c is more than the number of points of color $\bar{c}$, at the end of the process, in heavy points, the size of color c is larger than the size of color $\bar{c}$. Thus, by (2) and (3), at the end of stage 3, the extended heavy points has size $O(rb \cdot N_{\mathrm{H}})$ and is (r, b)-balanced as promised in Lemma 4.4.

Runtime analysis of MinHeavyPoints. Here we analyze the runtime of MINHEAVYPOINTS which corresponds to step 1 in FAIRLETDECOMPOSITION. Note that stage 1 only requires $O(1)$ operations on the number of red and blue points in $T(v)$. Each of stage 2 and stage 3 requires $O(1)$ operations on the number of red and blue points in all non-empty children of $T(v)$. Although the number of children of $T(v)$ can be as large as γ^d, for each node v in T, MINHEAVYPOINTS performs $O(1)$ operations on the number of red and blue points in $T(v)$ exactly twice: when it is called on v and the parent of v. Hence, in total MINHEAVYPOINTS performs $O(1)$ time on each node in T which in total is $O(n)$.

Algorithm 6 NONSATFAIRLET(N_r, N_b, r, b): returns the non-saturated fairlet in a set with (N_r, N_b) points.

if $N_r \leq N_b$ **then**
  $z_r \leftarrow N_r - \lfloor \frac{N_r}{b} \rfloor \cdot b$, $z_b \leftarrow N_b - \lfloor \frac{N_r}{b} \rfloor \cdot r$
else
  $z_b \leftarrow N_b - \lfloor \frac{N_b}{b} \rfloor \cdot b$, $z_r \leftarrow N_r - \lfloor \frac{N_b}{b} \rfloor \cdot r$
end if
return (z_r, z_b)

5. Experiments

In this section we show the performance of our proposed algorithm for (r, b)-fair k-median problem on three different standard data sets considered in (Chierichetti et al., 2017) which are from UCI Machine Learning Repository (Dheeru & Karra Taniskidou, 2017)[3]. Furthermore, to exhibit the performance of our algorithms on large and high-dimensional scale datasets, we consider an additional data set.

- **Diabetes.** The dataset[4] represents 10 years of clinical care at 130 US hospitals and in particular represent the information and outcome of patients pertaining to diabetes (Strack et al., 2014). Points are in $\mathbb{R}^2$ and dimensions correspond to two attributes ("age", "time-in-hospital").
- **Bank.** The dataset[5] is extracted from marketing campaigns of a Portuguese banking institution (Moro et al., 2014). Among the information about the clients, we selected ("age", "balance", "duration-of-account") as attributes to represent the dimensions of the points in the space.
- **Census.** The dataset[6] contains the records extracted from 1994 US Census (Kohavi, 1996). We picked attributes ("age" , "fnlwgt", "education-num", "capital-gain", "hours-per-week") to represent the points in the space.
- **Census II.** The dataset[7] contains the records extracted from 1990 US Census. We picked 25 numeric attributes to represent points in the space.

Algorithm. We essentially implement the algorithm described in Section 4.[8] However, instead of building $\mathrm{poly}(r, b)$-HST, in our implementation, we embed the points into a 2-HST. After computing a fairlet-decomposition of the points with given balance parameters, we run an existing K-medoids clustering subroutine[9].

Results. Comparing the cost of the solution returned by our fairlet decomposition algorithm with the result of (Chierichetti et al., 2017) (as in Table 1) shows that we achieve empirical improvements on all instances. The main reason is that our algorithm is particularly efficient when the input pointset lies in a low dimensional space which is the case in all three datasets "Diabetes", "Bank" and "Census". Moreover, unlike (Chierichetti et al., 2017), for each dataset, we can afford running our algorithm on the whole dataset (see Table 3). Empirically, the running time of our algorithm scales almost linearly in the number points in the input pointset (see Figure 1).

In Figure 1 and both Table 1 and 3, the reported runtime for each sample size S is the median runtime of our algorithm on 10 different sample sets from the given pointset each of size S.

6. Conclusion and Future Direction

We presented a scalable algorithm for the fair k-median problem by applying the techniques from embedding to tree metrics and locally constructing fairlets in a near linear time. We have demonstrated the runtime analysis of our algorithm both theoretically and empirically.

Designing a constant factor algorithm for the fair k-median problem with a near-linear time is an interesting open problem.

[3] https://archive.ics.uci.edu/ml/datasets/diabetes
[4] https://archive.ics.uci.edu/ml/datasets/diabetes+130-us+hospitals+for+years+1999-2008
[5] https://archive.ics.uci.edu/ml/datasets/Bank+Marketing
[6] https://archive.ics.uci.edu/ml/datasets/adult
[7] https://archive.ics.uci.edu/ml/datasets/US+Census+Data+(1990)
[8] Our code is publicly available at `https://github.com/talwagner/fair_clustering`.
[9] https://www.mathworks.com/help/stats/kmedoids.html

Dataset	Balance	Fairlet Decomposition Cost		Fair Clustering Cost ($k = 20$)	
		(Chierichetti et al., 2017)	**Ours**	(Chierichetti et al., 2017)	**Ours**
Diabetes (1000 points)	0.8^a	~ 9836	2971	~ 9909	4149
Bank (1000 points)	0.5	$\sim 5.46 \times 10^5$	5.24×10^5	$\sim 5.55 \times 10^5$	6.03×10^5
Census (600 points)	0.5	$\sim 3.59 \times 10^7$	2.31×10^7	$\sim 3.65 \times 10^7$	2.41×10^7

Table 1. The table compares the performance of our fairlet-decomposition algorithm and the algorithm of (Chierichetti et al., 2017). We remark that the number for (Chierichetti et al., 2017) mentioned in this table are not explicitly stated in their paper and we have extracted them from Figure 3 in their paper. Note that the cost denotes the total distances of the points to their fairlet/cluster centroids.

[a]In (Chierichetti et al., 2017), based on the description of the experiment setup, the desired balance in all three datasets (including Diabetes) are 0.5. However, for Diabetes dataset, they have achieved the higher balance of value 0.8.

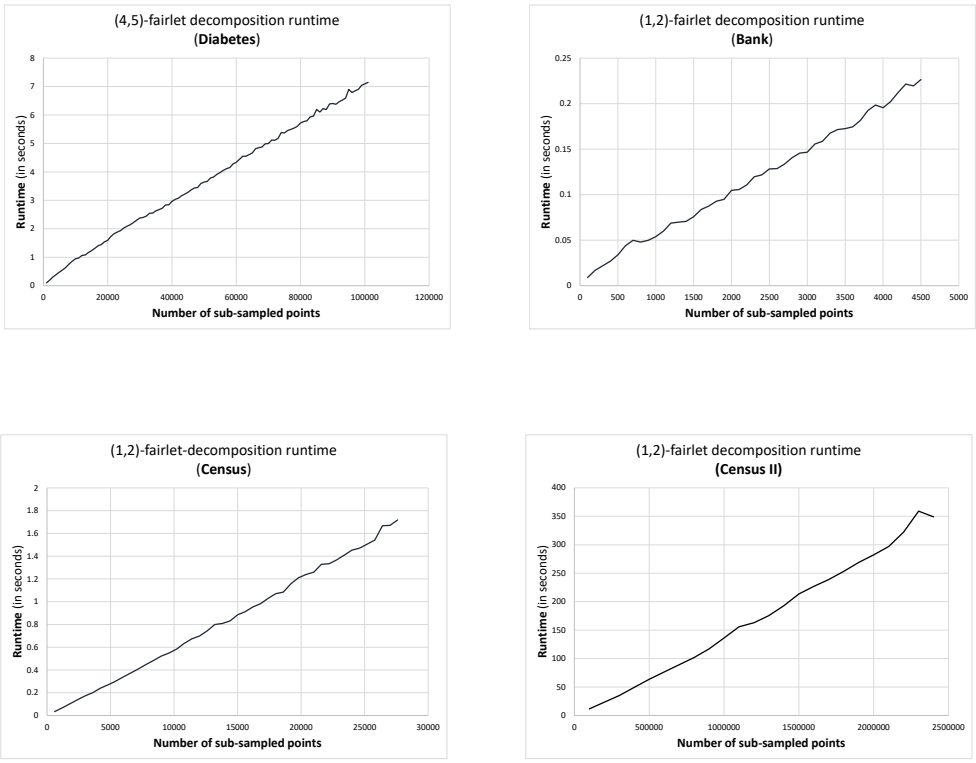

Figure 1. Each figure captures the running time of our fairlet decomposition algorithms with the specified balance parameter on different number of sample points from one of the four datasets: Diabetes, Bank, Census and Census II.

Dataset	Dimension	Number of points	Sensitive attribute
Diabetes	2	$101,765$	gender
Bank	3	$4,520$	marital-status
Census	5	$32,560$	gender
Census II	25	$2,458,285$	gender

Table 2. The description of the three datasets used in our empirical evaluation. In each dataset, the goal is find a fair k-median with respect to the sensitive attribute.

Dataset	Target Balance	Runtime for $k = 20$ (in sec)	
		Fairlet dec.	**Total**
Diabetes	0.8	7.42	14
Bank	0.5	0.23	7.63
Census	0.45	5.18	14.19
Census II	0.5	349.08	750.09

Table 3. The performance of our algorithm on all points in each dataset. We provide the runtime of both fairlet decomposition and the whole clustering process. Since Census dataset is not $(1, 2)$-balanced, we picked a lower balance-threshold for this dataset.

References

Bartal, Y. Probabilistic approximation of metric spaces and its algorithmic applications. In *Foundations of Computer*

Science, 1996. Proceedings., 37th Annual Symposium on, pp. 184–193. IEEE, 1996.

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

Bercea, I. O., Groß, M., Khuller, S., Kumar, A., Rösner, C., Schmidt, D. R., and Schmidt, M. On the cost of essentially fair clusterings. *CoRR*, abs/1811.10319, 2018.

Chierichetti, F., Kumar, R., Lattanzi, S., and Vassilvitskii, S. Fair clustering through fairlets. In *Advances in Neural Information Processing Systems*, pp. 5036–5044, 2017.

Chouldechova, A. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163, 2017.

Chouldechova, A. and Roth, A. The frontiers of fairness in machine learning. *arXiv preprint arXiv:1810.08810*, 2018.

Dheeru, D. and Karra Taniskidou, E. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Indyk, P. Algorithmic applications of low-distortion geometric embeddings. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pp. 10–33, 2001.

Kohavi, R. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *KDD*, volume 96, pp. 202–207, 1996.

Makarychev, K., Makarychev, Y., and Razenshteyn, I. Performance of johnson-lindenstrauss transform for k-means and k-medians clustering. *arXiv preprint arXiv:1811.03195*, 2018.

Moro, S., Cortez, P., and Rita, P. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

Rösner, C. and Schmidt, M. Privacy preserving clustering with constraints. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pp. 96:1–96:14, 2018.

Schmidt, M., Schwiegelshohn, C., and Sohler, C. Fair coresets and streaming algorithms for fair k-means clustering. *arXiv preprint arXiv:1812.10854*, 2018.

Strack, B., DeShazo, J. P., Gennings, C., Olmo, J. L., Ventura, S., Cios, K. J., and Clore, J. N. Impact of hba1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records. *BioMed research international*, 2014, 2014.