Clustering without Over-Representation

Sara Ahmadian
Google Research
New York, NY, US
sahmadian@google.com

Alessandro Epasto
Google Research
New York, NY, US
aepasto@google.com

Ravi Kumar
Google Research
Mountain View, CA, US
ravi.k53@gmail.com

Mohammad Mahdian
Google Research
New York, NY, US
mahdian@google.com

ABSTRACT

In this paper we consider clustering problems in which each point is endowed with a color. The goal is to cluster the points to minimize the classical clustering cost but with the additional constraint that no color is over-represented in any cluster. This problem is motivated by practical clustering settings, e.g., in clustering news articles where the color of an article is its source, it is preferable that no single news source dominates any cluster.

For the most general version of this problem, we obtain an algorithm that has provable guarantees of performance; our algorithm is based on finding a fractional solution using a linear program and rounding the solution subsequently. For the special case of the problem where no color has an absolute majority in any cluster, we obtain a simpler combinatorial algorithm also with provable guarantees. Experiments on real-world data shows that our algorithms are effective in finding good clustering without over-representation.

CCS CONCEPTS

• **Information systems → Clustering**; *Data mining*; • **Theory of computation → Facility location and clustering**; **Unsupervised learning and clustering**.

ACM Reference Format:
Sara Ahmadian, Alessandro Epasto, Ravi Kumar, and Mohammad Mahdian. 2019. Clustering without Over-Representation. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330987

1 INTRODUCTION

Clustering is a fundamental problem in data mining and unsupervised machine learning. Many variants of this problem have been studied in the literature. In a number of applications, clustering needs to be performed in the presence of additional constraints, such as those associated with fairness or diversity. Chierichetti et al. [9] study one such clustering problem, where the constraint is that the distribution of a particular feature (say, gender) in each cluster is identical to that of the general population. This is a highly constraining requirement, particularly in cases where the protected feature can take many values, and in many cases such a clustering does not exist. Furthermore, in many applications, such as the ones explained below, proportional representation is not really required: a clustering that ensures no particular feature value is highly over-represented in any cluster suffices.

A motivating application for our work is the following: every day, online advertising systems sell billions of advertising opportunities, specified by keywords the advertisers provide, through auctions. This is a highly heterogeneous set of auctions, and to optimize any of the auction parameters, one needs to cluster this set into smaller, more homogeneous, clusters. However, to ensure that no advertiser can manipulate this process, it is crucial that no advertiser has a large market share in any cluster (see [12] for a theoretical justification of this statement). Hence, keywords must be clustered such that no advertiser is over-represented in any cluster.

In addition to the above, there are other settings where an upper bound on the representation of each group in each cluster can capture real-world requirements. For example, in clustering news articles, requiring that no cluster is dominated by a certain view point or a certain news source is a good way to ensure balance and diversity in each cluster. Another example is clustering a number of agents into committees, where it is desirable that no committee is dominated by agents of a certain background. See Celis et al. [6] for an example where a similar constraint is applied to the problem of selecting a single committee maximizing a certain scoring function.

Our contributions. In this paper we formulate the problem of clustering without over-representation and study its algorithmic properties. For the clustering part, we focus on the k-center formulation. While there are many different well-studied models for clustering (such as k-median, k-means, k-center, or correlation clustering), we have picked the k-center model because of its theoretical simplicity (which allows us to prove good theoretical bounds) as well as the strong guarantees that are useful in many applications (that every point in a cluster is close to the center of that cluster).

Our formulation of the problem is in terms of a parameter α that specifies the maximum fraction of nodes in a cluster that have a specific value for the protected feature. Our main results are the following. First, for the case of $\alpha = 1/2$, we obtain a combinatorial approximation algorithm. Note that $\alpha = 1/2$ is a canonical case as it corresponds to ensuring that no cluster is dominated by a group with an absolute majority. Second, for the case of general α, we

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08.
https://doi.org/10.1145/3292500.3330987

give an approximation algorithm based on linear programming (LP) that achieves a bicriteria approximation guarantee. We also prove that the problem is NP-hard to approximate. Finally, we evaluate our LP-based algorithm on a number of real data sets, showing that its performance is even better than the theoretical guarantees.

Related work. Clustering is a classical problem in unsupervised learning and finds application in a variety of settings (see, e.g., Jain [19]); examples include information retrieval, image segmentation, and targeted marketing. The most popular clustering formulation studies the problem under an optimization objective that minimizes the ℓ_p norm for $p \in \{1, 2, \infty\}$ corresponding to k-median, k-means, and k-center, respectively. In this work, our focus is on the k-center case, which admits a 2-approximation [15, 17] and is NP-hard to approximate within a factor better than 2 [18].

Fairness in machine learning is relatively new but has received a significant amount of attention. This includes research on defining notions of fairness [5, 10, 13, 21] and on designing algorithms that respect fairness [1, 6, 7, 9, 20, 21, 27]. A recent line of work considers batch classification algorithms that achieve group fairness or equality of outcomes and avoid disparate impact [5, 13, 14, 22].

Chierichetti et al. [9] extended the notion of disparate impact to clustering problems and studied the fair k-center problem in the case there are only two groups (also called colors). This was later generalized by Rösner and Schmidt [26] to multiple groups, achieving a 14-approximation algorithm in the general case. Even with two colors, the problem is challenging, and the optimum solution can violate common conventions, e.g., a point may not necessarily be assigned to the closest open center. The main difference between our work and that of [9, 26] is that the latter focuses on the problem of finding a clustering where the distribution of colors is in each cluster is exactly the same as the distribution of colors over all given data points, whereas we only require that in each cluster, the fraction of nodes of each color is at most a given threshold. Note that requiring exact proportional representation in each cluster is often prohibitively restrictive. For example, if the number of times different colors appear in the graph are relatively prime, there is no non-trivial feasible clustering in the setting of Chierichetti et al. [9], whereas our formulation often admits non-trivial solutions.

Concurrently and independently, Bera et al. [3] and Bercea et al. [4] obtained algorithms to convert an arbitrary clustering solution to a fair one, sacrificing both approximation and fairness. They provide bicriteria approximations for a more general problem (with upper and lower bound on the representation of a color). Our algorithm, however, is simpler and we prove (at most) an additive 2 violation for the fairness constraint (improved to 1 for a special case) in contrast to Bera et al. [3] who prove an additive 4 violation and Bercea et al. [4] who do not bound the additive violation.

There has been some work on clustering with diversity [24], where the objective is to ensure each cluster has at least a certain number of colors; our objective is clearly different from this. The large body of work on clustering with constraints [2], to the best of our knowledge, does not address the over-representation constraint.

Outline of the paper. In Section 2 we formalize the problem of finding an α-capped k-center clustering. In Section 3 we present our main theoretical result, an LP-based algorithm for the general α case. Later, in Section 4 we provide a purely combinatorial algorithm for the $\alpha = \frac{1}{2}$ case. Then in Section 5 we report the results of our empirical studies. In Section 6 we show that obtaining a decomposition in α-capped clusters of minimum cost is hard for $\alpha \leq 1/2$ irrespective of the constraint on the number of clusters. Finally, in Section 7 we discuss future avenues of research.

2 MODEL AND PRELIMINARIES

In the *k-clustering* problem, we are given a set D of points in a metric space with the distance function $d(\cdot, \cdot)$ and an integer bound k, and the goal is to cluster the points into at most k clusters $C_1, C_2, \ldots, C_k$. Various clustering problems have been studied and in this paper, we focus on k-center clustering. We define the problem in terms of *facility location* terminology where points are referred to as *clients* and clusters are defined by the assignment of clients to *centers* (also called *facilities*). An instance $\mathcal{I} = (D, F, d, k)$ of *k-center* consists of a client set D, a facility set $F = D$, a metric space $d : D \times D \to \mathbb{R}_{\geq 0}$, and a positive integer bound k. A feasible k-center solution is a pair (F', σ), where $F' \subseteq F$ is a set of at most k facilities and $\sigma : D \to F'$ is a mapping that assigns each client j to a facility $\sigma(j) \in F'$. The goal is to find a feasible solution that minimizes the *maximum radius* or *clustering cost* defined as $\lambda(F', \sigma) = \max_{j \in D} d(j, \sigma(j))$. Of course, in the classic k-center problem, once the set F' is determined, assigning each client to the closest facility in F' yields the assignment with minimum objective. With additional constraints, however, the closest assignment might be infeasible.

Even though the standard k-center problem is computationally hard, it admits an elegant 2-approximation algorithm [17]: first select an arbitrary point as center, then, iteratively pick the next center to be the point that is farthest from all currently chosen centers, until k centers are chosen. For completeness, we present it below (Algorithm 1). In this paper, we consider the *α-capped k-center* problem where points have colors and we have a constraint on the representation of each color in each cluster. More precisely, in an α-capped k-center instance $\mathcal{I} = (D, F, d, k, \alpha, c)$, in addition to the input of classical k-center, we are given a fractional bound $\alpha \in (0, 1]$ and a color $c(j)$ for each point $j \in D$. We use D_c to denote the set of clients of color c. A feasible solution $(F', \sigma : D \to F')$ is a feasible k-center solution that satisfies the *representation constraint*, which states that for each color c and each facility i, the total number of clients of color c assigned to i should be no more than α fraction of all clients assigned to i. This constraint can be written as

$$\forall i \in F', c : \ |\sigma^{-1}(i) \cap D_c| \leq \alpha |\sigma^{-1}(i)|.$$

The goal in α-capped k-center problem is to find a feasible solution (F', σ) that minimizes

Algorithm 1 Greedy-k-center($\mathcal{I} = (D, D \subseteq F, d, k \geq 1)$).

1: $i_0 \leftarrow$ an arbitrary client in D.
2: $F' = \{i_0\}$
3: **for** $l \in \{1, 2, \ldots, k-1\}$ **do**
4: $\quad i_l \leftarrow \arg\max_{j \in D} \min_{i \in F'} d(j, i)$, the furthest client from F'
5: $\quad F' \leftarrow F' \cup \{i_l\}$
6: $\forall j \in D : \sigma(j) \leftarrow i = \arg\min_{i \in F'} d(i, j)$
7: $\lambda \leftarrow \lambda(F', \sigma)$
8: **return** $((F', \sigma), \lambda)$

$$\lambda(F', \sigma) = \max_{j \in D} d(j, \sigma(j)).$$

Let (F^*, σ^*) be the optimal clustering, and let $\lambda^* = \lambda(F^*, \sigma^*)$ be the optimal clustering cost. A *ρ-approximation algorithm*, for $\rho \geq 1$, outputs a clustering (F', σ) such that $\lambda(F', \sigma) \leq \rho \cdot \lambda(\sigma^*, C^*)$.

3 A GENERAL ALGORITHM

We present a general algorithm to solve the α-capped k-center clustering problem. The main idea is to first solve a linear program (LP) relaxation of the problem to obtain a fractional solution and then modify the fractional solution—sacrificing a little both in the approximation factor and in the representation constraint—to get an integral solution. In the course of doing this, we will get what is called a *bicriteria* algorithm, i.e., while we get a constant-factor approximation to α-capped k-center, our solution will violate the α upper bound mildly. In fact, we can show that for each color and each facility, there are at most two extra clients in addition to the allowed number of clients, so the cap is violated additively by at most two additional nodes—a negligible quantity for a large cluster.

3.1 An LP formulation

For a given distance $\lambda \in \mathbb{R}_{\geq 0}$, consider the problem of finding a feasible assignment of clients to facilities in such a way that the clustering cost of the solution is at most λ. This problem can be formulated using the following integer program (IP).

$$\begin{aligned}
\textstyle\sum_{i \in F} x_{ij} &\geq 1 && \forall j \in D, && (1)\\
x_{ij} &\leq y_i && \forall i \in F, j \in D, && (2)\\
\textstyle\sum_{j \in D_c} x_{ij} &\leq \alpha \cdot \textstyle\sum_{j \in D} x_{ij} && \forall c \in [t], i \in F, && (3)\\
\textstyle\sum_{i \in F} y_i &\leq k, && && (4)\\
x_{ij}, y_i &\in \{0, 1\} && \forall i \in F, j \in D, && (5)\\
x_{ij} &= 0 && \forall i \in F, j \in D, d(i, j) > \lambda. && (6)
\end{aligned}$$

Here, the indicator variable y_i denotes if facility $i \in F$ is open or not and the indicator variable x_{ij} denotes if client j is assigned to facility i. Note that by constraint (6), x_{ij} can take non-zero value only if facility $i \in F$ is at distance at most λ from client $j \in D$. Constraint (2) captures that a facility must be open if it has a client assigned to it, (3) captures the representation constraint, and (4) captures that the total number of open facilities is at most k.

Before relaxing the integrality constraint of the above IP, we strengthen it by adding the following constraint: if a facility i is open, it has to serve at least $\lceil \frac{1}{\alpha} \rceil$ clients to satisfy the representation constraint. Therefore, every integral solution of the above program must satisfy the inequality $\sum_{j \in D} x_{ij} \geq \lceil \frac{1}{\alpha} \rceil \cdot y_i$.

We consider the following LP obtained by adding this constraint and relaxing the integrality constraint (5). We use $\mathcal{P}(\lambda, \alpha)$ to denote the polytope defined by this LP.

$$\begin{aligned}
\textstyle\sum_{i \in F} x_{ij} &\geq 1 && \forall j \in D,\\
x_{ij} &\leq y_i && \forall i \in F, j \in D,\\
\textstyle\sum_{j \in D_c} x_{ij} &\leq \alpha \cdot \textstyle\sum_{j \in D} x_{ij} && \forall c \in [t], i \in F, && (7)\\
\textstyle\sum_{j \in D} x_{ij} &\geq \lceil \tfrac{1}{\alpha} \rceil \cdot y_i && \forall i \in F,\\
\textstyle\sum_{i \in F} y_i &\leq k, &&\\
0 \leq y_i &\leq 1 && \forall i \in F,\\
0 \leq x_{ij} &\leq 1 && \forall i \in F, j \in D,\\
x_{ij} &= 0 && \forall i \in F, j \in D, d(i, j) > \lambda.
\end{aligned}$$

As mentioned above, we present a bicriteria algorithm that finds a solution that might violate the representation constraint, i.e., constraint (7). We use the notation $\mathcal{P}(\lambda, \alpha, \Delta)$, for $\Delta \in \mathbb{R}_{\geq 0}$, to denote the set of points that satisfy all the constraint for $\mathcal{P}(\lambda, \alpha)$ except constraint (7) and only violate that constraint with an additive error of Δ, i.e., $\sum_{j \in D_c} x_{ij} \leq \alpha \cdot \sum_{j \in D} x_{ij} + \Delta$. Note that $\mathcal{P}(\lambda, \alpha) = \mathcal{P}(\lambda, \alpha, 0)$.

3.2 Outline

Recall that λ^* is the value of the optimal solution to the problem. The main idea in our algorithm is that, since the polytope $\mathcal{P}(\lambda^*, \alpha)$ is non-empty, by binary search, we can first find the smallest value λ' such that $\mathcal{P}(\lambda', \alpha)$ is non-empty (since the set of distances between pairs of points is finite). Note that the non-emptiness check via solving the LP also yields a point $(x, y) \in \mathcal{P}(\lambda', \alpha)$, which is a fractional solution to the LP. The plan then is to use (x, y) to construct a feasible integral solution in a slightly larger polytope, namely, $(x'', y'') \in \mathcal{P}(3\lambda', \alpha, 2)$, where x'', y'' are integral and hence will correspond to a valid solution to the k-center problem.

Theorem 3.1. *Given an instance $\mathcal{I}$ of α-capped k-center clustering, there is a polynomial time algorithm that finds a solution (F', σ) of cost at most $3\lambda^*$ such that*

$$|\sigma^{-1}(i) \cap D_c| \leq \alpha \cdot |\sigma^{-1}(i)| + 2.$$

In the case of $1/\alpha \in \mathbb{Z}$, we can actually improve the additive term to 1 and in term of multiplicative bound we get $|\sigma^{-1}(i) \cap D_c| \leq 2\alpha |\sigma^{-1}(i)|$.

To prove Theorem 3.1, the integral solution (x'', y'') is constructed from (x, y) in two steps. In the first step, we construct a solution $(x', y') \in \mathcal{P}(3\lambda', \alpha)$ using (x, y), where y' is integral. This step can be thought of as determining which facilities to open based on the fractional solution. In the second step, we construct an integral solution $(x'', y'') \in \mathcal{P}(3\lambda', \alpha, 2)$. This step uses the open facilities to define a suitable maximum flow problem to obtain an assignment of clients to facilities. We describe these two steps.

3.3 Finding facilities to open

The goal in this step is to find $(x', y') \in \mathcal{P}(3\lambda', \alpha)$ where y' is integral. Let $F' \subseteq F$ be a maximal subset of facilities such that any two facilities $i, i' \in F'$ are at least distance $2\lambda'$ from each other, i.e., $d(i, i') > 2\lambda'$. We open all facilities in F', i.e., set $y'_i = 1$ for $i \in F'$ and $y'_i = 0$ for $i \notin F'$. Note that if λ' is a correct guess of the optimum, none pair of clients at locations in F' can be served by the same center and so the size of F' is smaller than or equal to k. Next, we show how to define x'. We essentially transfer the fractional assignment of clients from F to F'. First we define a mapping $\theta : \{i \in F \mid y'_i > 0\} \to F'$ as

- If $i \in F'$, then $\theta(i) = i$.
- If $i \notin F'$, then $\theta(i) = i'$ where $i' \in F'$ with $d(i, i') < 2\lambda'$. (Such an i' exists by the maximality of F'.)

Now for each client $j \in D$, we can define

$$x'_{ij} = \begin{cases} \sum_{i' \in \theta^{-1}(i)} x_{i'j} & i \in F' \\ 0 & \text{otherwise.} \end{cases}$$

We now show that (x', y') has the desired properties.

Lemma 3.2. *$(x', y') \in \mathcal{P}(3\lambda', \alpha)$ and y' is integral.*

Proof. Let us first show that x'_{ij} can only take non-zero value if facility i is at distance $3\lambda'$ from it. If x'_{ij} is non-zero, then there exists a facility i' where $\theta(i') = i$ and $x_{ij} > 0$. Since $x_{ij} > 0$, we get that $d(i', j) < \lambda'$ and since $\theta(i') = i$, $d(i, i') < 2\lambda'$, so by the triangle inequality, we have $d(j, i) \leq 3\lambda'$. Since x' is just rerouting the assignment of clients from facilities in F to F', $y_i = 1$ for all facilities in F', and F' has at most k facilities, (x', y') satisfy Constraints (1), (2), and (4). Constraint (3) is satisfied since for each $i \in F', c \in [t]$,

$$\sum_{j \in D_c} x'_{ij} = \sum_{i' \in \theta^{-1}(i)} \sum_{j \in D_c} x_{i'j} \leq \alpha \cdot \sum_{i' \in \theta^{-1}(i)} \sum_{j \in D} x_{ij} = \alpha \cdot \sum_{j \in D} x'_{ij},$$

where the inequality follows from the definition of θ. □

3.4 Assigning clients to facilities

The goal in this step is to construct a solution $(x'', y'') \in \mathcal{P}(3\lambda', \alpha, 2)$ such that x'', y'' are integral. In fact, $x''_{ij} > 0$ only if $x'_{ij} > 0$. We let (x'', y'') be the solution to the following maximum flow problem and use the fact that a network with integral bound on edges and integral demands, if feasible, always has an integral solution.

Construct a flow network (V, A) as follows:

- $V = \{s, t\} \cup D \cup \{(i, c) \mid i \in F', c \in [t]\}$.
- $A = A_1 \cup A_2 \cup A_3 \cup A_4$ where $A_1 = \{(s, j) \mid j \in D\}$ with capacity 1, $A_2 = \{(j, (i, c)) \mid j \in D_c, x'_{ij} > 0\}$ with capacity 1, $A_3 = \{((i, c), i)\}$ with lower bound $\lfloor \sum_{j \in D_c} x'_{ij} \rfloor$ and capacity $\lceil \sum_{j \in D_c} x'_{ij} \rceil$, and $A_4 = \{(i, t)\}$ with lower bound $\lfloor \sum_{j \in D} x'_{ij} \rfloor$ and capacity $\lceil \sum_{j \in D} x'_{ij} \rceil$.

Note that (x', y') is a feasible flow of value $|D|$, so there is an integral flow of value $|D|$ such that a client j sends a flow to a facility i if $x'_{ij} > 0$. Thus $x''_{ij} > 0$ only if client j is at distance $3\lambda'$ from facility i. This concludes the steps of our algorithm (Algorithm 2). It remains

Algorithm 2 Fair-k-center($\mathcal{I} = (D, F, d, k), \alpha, \lambda$).

1: $(x, y) \leftarrow$ a feasible solution of $\mathcal{P}(\lambda, \alpha)$
2: **if** $\mathcal{P}(\lambda, \alpha)$ is empty **then**
3: **return** $(\emptyset, \emptyset)$
4: $F' \leftarrow$ a maximal subset of F where $\forall i \neq i' \in F' : d(i, i') > 2\lambda$
5: $(x', y') \leftarrow$ client reassignment based on F' (Section 3.3)
6: $(x'', y'') \leftarrow$ client assignment based on max flow in network (V, A) (Section 3.4)
7: $F^s \leftarrow \{i \mid y''_i > 0\}$
8: $\forall j \in D$: $\sigma^s(j) \leftarrow i$ where $x''_{ij} > 0$
9: **return** (F^s, σ^s)

to bound the violation of the representation constraint.

Lemma 3.3. *For any color c and any facility i, $\sum_{j \in D_c} x''_{ij} \leq \alpha \cdot \sum_{j \in D} x''_{ij} + 2$ where the additive term can be improved to $+1$ for $1/\alpha \in \mathbb{Z}^+$.*

Proof. Let $T' = \sum_{j \in D_c} x'_{ij}$, $B' = \sum_{j \in D} x'_{ij}$, $T'' = \sum_{j \in D_c} x''_{ij}$, and $B'' = \sum_{j \in D} x''_{ij}$. Since (x', y') is a feasible solution of $\mathcal{P}(\lambda', \alpha)$, we have $T' \leq \alpha \cdot B'$. Using the lower bounds and upper bounds on the edge $((i, c), i) \in A_3$, we know that $\lfloor T' \rfloor \leq T'' \leq \lceil T' \rceil$ and $\lfloor B' \rfloor \leq B'' \leq \lceil B' \rceil$. Since $\lceil T' \rceil < T' + 1$, we can bound T'' in terms of B'' as follows:

$$T'' < T' + 1 \leq \alpha B' + 1 \leq \alpha B'' + \alpha + 1 \leq \alpha B'' + 2.$$

Now suppose $\alpha = 1/m$ for some $m \in \mathbb{Z}^+$ and suppose $B'' = p \cdot m + r$ for $r < m$. Then, $\alpha B'' + \alpha = p + \frac{r+1}{m}$. If $r < m - 1$, then the largest integer smaller than $\alpha B'' + \alpha + 1$ is $p + 1 \leq \alpha B'' + 1$. If $r = m - 1$, then $\alpha B'' + \alpha + 1 = p + 2$, now since $T'' < p + 2$, it follows that $T'' \leq p + 1 \leq \alpha B'' + 1$. □

We can bound the cost of the solution, in terms of violating the representation constraint multiplicatively as follows.

Corollary 3.3.1. *For any color c and facility i, $\frac{\sum_{j \in D_c} x''_{ij}}{\sum_{j \in D} x''_{ij}} \leq 2\alpha$ for $1/\alpha \in \mathbb{Z}^+$.*

Proof. Since $B'' \geq \lfloor B' \rfloor \geq \lfloor \frac{1}{1/m} \rfloor = m$, the $+1$ term in the last line of the proof of Lemma 3.3, can be bounded by $\alpha B''$. □

4 AN ALGORITHM FOR $\alpha = 1/2$

In this section, we present a simple, combinatorial approximation algorithm for the important special case of $\alpha = 1/2$. This case corresponds to finding a clustering of the points such that no color is the absolute majority in any cluster, i.e., every color in a cluster occurs at most half of the times as the cluster size. To proceed, we need two notions, namely, caplets and threshold graphs.

Caplets. Let G be any graph whose set of nodes is D. A *caplet* in G is a subset $K \subseteq D, 2 \leq |K| \leq 3$ with distinct colors, i.e., $c(j) \neq c(j')$ for $j \neq j' \in K$. Since caplets can have either size two or three, we call the former case an *edge caplet* and the latter a *triangle caplet*. For two caplets K_1 and K_2, let $\text{dist}(K_1, K_2)$ be defined as the minimum distance between pair of points of the two caplets, i.e., $\text{dist}(K_1, K_2) = \min_{j_1 \in K_1, j_2 \in K_2} d(j_1, j_2)$. Note that the distance function defined on caplets is not necessarily a metric but will be useful to bound the distance between points belonging to different caplets. The *diameter* of a caplet K is $\text{diam}(K) = \max_{j, j' \in K} d(j, j')$. The diameter of a set $\mathcal{K}$ of caplets is $\text{diam}(\mathcal{K}) = \max_{K \in \mathcal{K}} \text{diam}(K)$.

A *caplet decomposition* $\kappa(G)$ of a connected graph G, if it exists, is a set of edge caplets and at most one triangle caplet such that each node in G is present in exactly one caplet. Note that the only time when a caplet decomposition uses a triangle caplet is when the number of nodes in G is odd. The caplet decomposition can be found in polynomial time by guessing the triangle if the size of graph is odd, and then finding the perfect matching on the remaining vertices.

Threshold graph. Given D, a threshold $\tau > 0$, we define a *threshold graph* $G(\tau) = (D, E)$ to be an undirected graph on the points in D, where $(j, j') \in E$ iff they have different colors and they are at distance at most τ from each other, i.e., $c(j) \neq c(j')$ and $d(j, j') \leq \tau$.

4.1 Algorithm

First of all, we assume that we know the optimal value $\lambda^* = \lambda(\sigma^*)$. This is without loss of generality since by definition of k-center, $\lambda^* \in \{d(i, j) \mid i \in F, j \in D\}$. Hence an algorithm can enumerate over the set of all possible values for λ^*; at worst, this enumeration only

costs an additional factor $|F| \cdot |D|$ in the running time.[1] Assuming we know λ^*, the idea is to create the threshold graph with $2\lambda^*$ as the threshold, and then to decompose it into caplets. Finally, the caplets can be clustered using the greedy algorithm for k-center. The steps are presented in Algorithm 3.

Algorithm 3 Non-dominant-k-center($\mathcal{I} = (D, F, d, k), \alpha = 1/2$).

1: **for** $\lambda \in \{d(i,j) \mid i \in F, j \in D\}$ in non-decreasing order **do**
2: $D' \leftarrow \emptyset$
3: **for** Connected component C of $G(2\lambda)$ **do**
4: $G_C \leftarrow (C, E')$ where $E' = \{(j,j') \mid c(j) \neq c(j'),\ d(j,j') \leq 10\lambda\}$
5: **if** no caplet decomp. for G_C **then**
6: reject λ and continue to next λ
7: $D' \leftarrow D' \cup \{j_K \mid$ arbitrary client $j_K \in K \in \kappa(G_C)\}$
8: $((F^g, \sigma^g), \lambda^g) \leftarrow$ Greedy-k-center($\mathcal{I}' = (D', F, d, k)$)
9: **if** $\lambda^g > 2\lambda$ **then**
10: reject λ and continue to next λ
11: $\forall j : \sigma^s(j) \leftarrow \sigma^g(j_K)$ where $j, j_K \in K$.
12: **return** (F^g, σ^s)

Note that our approach is similar in spirit to the fairlet decomposition approach proposed in [9]. However, since our representation constraint is less stringent than the fair clustering constraint, as we will see, the reasoning becomes more delicate and involved.

To show that Algorithm 3 obtains a provably good approximation, we show a key characterization: there is a caplet decomposition of each connected component of $G(2\lambda^*)$ with small diameter.

Lemma 4.1. *For each connected component C of $G(2\lambda^*)$, there is a caplet decomposition $\kappa(C)$ such that* $\mathrm{diam}(\kappa(C)) \leq 10\lambda^*$.

Before proving the lemma, we use it to show that Algorithm 3 gives a good approximation.

Theorem 4.2. *Algorithm 3 finds a $(1/2)$-capped k-clustering solution of cost at most $12\lambda^*$.*

Proof. Using Lemma 4.1, we know that the **if** statement (line 3 in Algorithm 3) fails for λ^*. Furthermore, since the optimal capped clustering yields a feasible solution for the k-center instance I and there is a 2-approximation algorithm for k-center, a feasible solution can be found for $\lambda = \lambda^*$ (line 7). Therefore the loop terminates successfully (line 8) for some $\lambda \leq \lambda^*$.

We next show we get a valid $(1/2)$-capped clustering. For each color c, note that the number of points of color c assigned to facility $i \in F$ is at most the number of caplets assigned to i. However, by definition, each caplet is of size at least two and has distinct colors. Therefore, no color can be the absolute majority for each $i \in F$; this proves the $(1/2)$-capped property. The cost of clustering is a 12-approximation since each point j in a caplet K assigned to a facility i is at most at distance 2λ from j_K and $d(j_K, j) \leq 10\lambda$ since $\mathrm{diam}(K) \leq 10\lambda$ by Lemma 4.1. The proof is complete as $\lambda \leq \lambda^*$. □

[1] One can also get an $1+\epsilon$ approximation of the optimum λ^* in logarithmic many tries with standard techniques.

4.2 Analysis

We now prove Lemma 4.1. Let C be a connected component of $G(2\lambda^*)$. There are two steps in the proof. In the first step, we find a set $\mathcal{K}_i$ of caplets with respect to each facility i such that $\mathrm{diam}(\kappa(\mathcal{K}_i)) \leq 2\lambda^*$. In the second step, we collect the caplets $\kappa(\mathcal{K}_i)$ for each $i \in F$ from the first step and appropriately modify them to obtain a caplet decomposition $\kappa(\mathcal{K})$ of C such that $\mathrm{diam}(\kappa(\mathcal{K})) \leq 10\lambda^*$. (If we naively take the union of the caplets for $i \in F$ we may not get a valid caplet decomposition of C since we might have more than one triangle caplet, violating the definition.)

The first step is relatively straightforward. Indeed, consider the optimal solution with open facilities F^* and an assignment $\sigma : D \to F^*$. Since for each open facility $i \in F^*$, the number of points with the same color is less than half of the points assigned to i, if $|\sigma^{-1}(i)|$ has even size, we can define a matching between points of different colors in $\sigma^{-1}(i)$. If $|\sigma^{-1}(i)|$ is odd, then there are at least three colors present in $\sigma^{-1}(i)$. Define the triangle to include three points of different colors and the rest of points in $\sigma^{-1}(i)$ can be matched to points of different colors. This yields $\mathcal{K}_i$ with the property that it has at most one triangle caplet. Furthermore that since all the points in $\sigma^{-1}(i)$ are at distance at most λ^* from i, by the triangle inequality, any two points in $\sigma^{-1}(i)$ are at distance at most $2\lambda^*$ from each other. Therefore, these points will belong to the same connected component of $G(2\lambda^*)$. Let $\tilde{\mathcal{K}}_C = \cup_{i \in F^*} \mathcal{K}_i \cap C$.

Next, we consider the second step. For this, it is helpful to work with the graph $G' = (\tilde{\mathcal{K}}_C, E)$ such that for $K, K' \in \tilde{\mathcal{K}}_C$, we have $(K, K') \in E$ if $\mathrm{dist}(K, K') \leq 2\lambda^*$. Notice that G' is connected since it is constructed from C.

The goal is to transform the caplets obtained in the first step into a valid caplet decomposition of C. This is done by finding a path between two triangle caplets and "shifting" points to get a new set of edge caplets, sacrificing some in the distance between caplets. Fix C henceforth.

From C, we construct a set P of disjoint paths with the following properties: each path in P is of the form $K_0, \ldots, K_\ell$ where (i) K_0 and K_ℓ are triangle caplets and $K_i, 1 \leq i < \ell$ are edge caplets, (ii) $\mathrm{dist}(K_i, K_{i+1}) \leq 6\lambda^*$, and (iii) $\mathrm{diam}(K_i) \leq 2\lambda^*$. Let T be a minimal rooted tree spanning the nodes corresponding to triangle caplets in C. Note that all the leaves in T correspond to triangle caplets and the internal nodes in T may be edge or triangle caplets. We perform a bottom-up procedure on T, removing paths from T and adding them P in an iterative manner; the procedure ends when T has at most one triangle caplet. Let T_f denote the rooted subtree of T rooted at a node f. In the bottom-up procedure, we maintain the property that for each scanned node f there is at most one triangle caplet in T_f. Note this property is already satisfied at the leaves. Let K be the deepest node in the current tree that does not satisfy this property. If K has more than one child, let $p_1 = (K, K_1, \ldots, K_r)$ and $p_2 = (K, K'_1, \ldots, K'_s)$ be two paths starting at K and ending at triangle caplets K_r and K_s. Note that the degree of internal nodes on p_1 and p_2 is exactly two by the choice of K. We add the path $p = (K_r, K_{r-1}, \ldots, K_1, K'_1, K'_2, \ldots, K'_s)$ to P and remove the edges of $p_1 \cup p_2$ from T. Since K_1 and K'_1 are at distance $2\lambda^*$ from K and points inside K are at distance at most $2\lambda^*$ from each other, K_1 and K'_1 are at distance at most $6\lambda^*$ from each other. We continue this procedure until K has at most one child. If K is a leaf, then we

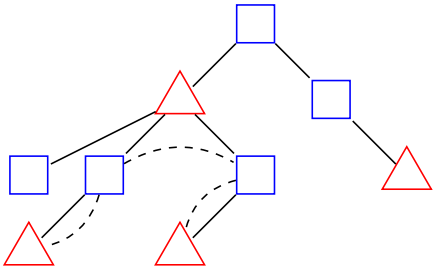


Figure 1: Construction of a path (dashed line) in P. The triangle nodes are triangle caplets and the square nodes are edge caplets.

remove if it is an edge caplet and leave it in T if otherwise. Else, let K' be the sole child of K. If K' is an edge caplet, we remove K' from T. If both K and K' are triangle caplets, we add them to P and remove both of them from T. We continue the procedure until we reach the root and at the end of this, there exists at most one triangle caplet that is not covered by a path in P. It is also easy to see that each path in P satisfies the desired properties. (See Figure 1.)

Now consider each $p = (K_0, K_1, \ldots, K_\ell) \in P$. Recall from property (i) above that K_0 and K_ℓ are triangle caplets and the rest are edge caplets. We define a new set of edge caplets $K'_0, \ldots, K'_{\ell+1}$ as follows. We pick an arbitrary point i_0 from K_0 and shift it to the next caplet K_1 and then shift some point from K_1 to the next caplet, and so on. More precisely, let i_0 be an arbitrary point in K_0, define $K'_0 = K_0 \setminus \{i_0\}$ and let $K'_1 = \{i_0, i'_1\}$ where i'_1 is point in K_1 with different color than i_0. We continue the process iteratively, where at each step r, we define the edge caplet K'_{r+1} to contain point i_r in K_r not covered by $K'_0, K'_1, \ldots, K'_r$ for $(r < \ell)$, and point i'_{r+1} in K_{r+1} with different color than i_r. In the last step, a point $i'_{\ell-1} \in K_{\ell-1}$ is shifted and matched to a point $i_\ell \in K_\ell$ and we define $K'_{\ell+1} = K_\ell \setminus \{i_\ell\}$. Note this process is possible since each caplet K_r contains at least two points of different colors, there always exists a point that has a different color than the shifted point. (See Figure 2.) By properties (ii) and (iii) above, the diameter of each caplet is at most $2\lambda^*$ and two consecutive caplets are at distance at most $6\lambda^*$ from each other. Applying the triangle inequality, we get that the diameter of the caplets in $K'_0, \ldots, K'_{\ell+1}$ is at most $10\lambda^*$.

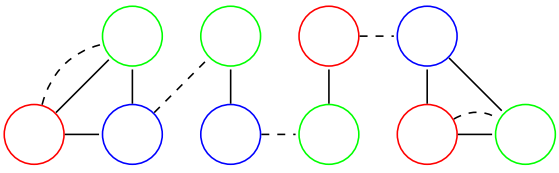


Figure 2: The shifting operation in action on a path of four caplets, beginning and ending with a triangle caplet. The solid lines denote the original caplets and the dotted lines denote the new caplets after the shifting operation.

Dataset	# Points	# Dim.	# Colors	Max ratio
4area	25,853	8	4	40.2
query	> 29,000	20	> 12,000	< 7.0%
reuters	2500	10	50	2.0%
victorian	4500	10	45	2.2%

Table 1: Datasets used. Column # Dim. reports the number of dimensions of the space used and column max ratio represents the maximum ratio of a color in the dataset.

5 EMPIRICAL EVALUATION

In this section we empirically evaluate our algorithms on several publicly-available datasets from the UCI Repository[2]and DBLP[3], as well as on a proprietary dataset related to online auctions. In our empirical analysis we focus on the LP-based algorithm (Section 3). We describe the datasets used, the baselines we consider, the quality measures we compute, and finally the results.

5.1 Datasets

The datasets reported in Table 1 come from different domains and represent Euclidean spaces with dimensions ranging from 8 to 20 as well as a wide range of colors (between 4 and > 12,000). The datasets report different levels of balance of color distribution, from complete balance (each color is equally represented in the whole dataset) to high imbalance (> 40% of points of one color).

We now describe more in detail the datasets used. We obtained two datasets (`reuters`, `victorian`) from text embeddings of multi-author datasets, one from a co-authorship graph embedding (`4area`), and one from online auctions (`query`). All datasets represent points in the Euclidean space and we always use the ℓ_2 distance.

(i) `reuters`[4]. It contains 50 English language texts from each of 50 authors (for a total of 2,500 texts). We transformed each text into a 10-dimensional vector using Gensim's Doc2Vec with standard parameter settings. Here, the colors represent the author of the text. We observe that clustering doc2vec embeddings has been used extensively in language analysis (see, e.g., [8]).

(ii) `victorian`[5]. It consists of texts from 45 English language authors from the Victorian era. Each text consists of 1,000-word sequences obtained from a book of the author (we use the training dataset). The data has been extracted and processed in [16]. From each document, we extract a 10-dimensional vector using again Gensim's doc2vec with standard parameter settings and we use the author as color. We use 100 texts from each author.

(iii) `4area` [3]. It contains 25,853 points in 8 dimensions representing each a researcher in one of four areas of CS: data mining, machine learning, databases, and information retrieval. The color is the main area of research of the author. The points are obtained by using the graph embedding method DeepWalk [25] on the undirected co-authorship graph of `4area`, using default settings.

(iv) `query`. It is a representative subset of an anonymized proprietary dataset. Each point in this dataset represents a bag of queries

[2]http://archive.ics.uci.edu/ml
[3]http://dblp.uni-trier.de/xml/
[4]Available at archive.ics.uci.edu/ml/datasets/Reuter_50_50
[5]Available at archive.ics.uci.edu/ml/datasets/Victorian+Era+Authorship+Attribution

in an online auction environment. The points have 20 dimensions and are obtained with a proprietary embedding method that encodes semantic similarity. The color of the point is the anonymous id of the main advertiser of the submarket represented by the bag.

5.2 Experimental setup

5.2.1 Baselines. We use the following two baselines.

(i) Greedy. Because the k-center problem is NP-hard, even without the additional constraint of being α-capped, we use the well-known k-center greedy method, which ignores the representation constraint, as a gold standard. Notice that this algorithm returns a 2-approximation of the cost of the optimum (without representation constraint) which is always lower than the optimum cost of our problem. To further strengthen the baseline, we post-process the output apply a round of the standard Lloyd iterative algorithm, with k-center cost. This step can only improve the results. We use this method as a gold standard baseline to evaluate the increased cost incurred by our algorithm to enforce the representation constraint and we measure how much our algorithm improves the representation constraint bound of the clusters.

(ii) Random. We also compare against the baseline of sampling k random points as centers and assigning all points to the nearest center selected. Because this method depends on randomness (while all other algorithms are deterministic), we rerun the algorithm ten times and report the average results. Notice that this algorithm as well does not (necessarily) respect the capped constraints.

5.2.2 Measures of quality. We evaluate the following measures of quality for a clustering.

Cost. We measure the maximum distance of a point to the nearest center in the solution. In particular, we compare the cost of the solution output by our α-capped k-clustering algorithm, (for a certain α), and the solution of the baselines for the same k.

Additive violation of representation constraint. Recall that our algorithm in Section 3 can output a solution mildly violating the representation constraint. We wish to study how big is this violation in practice. To this end, let C be a cluster in the solution output of an α-capped clustering instance. The maximum allowed number of points of a certain color in the cluster C is $\lfloor |C|\alpha \rfloor$. We let $\Delta = \max_{C,c} \max(|C \cap D_c| - \lfloor |C|\alpha \rfloor, 0)$ be the maximum additive violation of the α-capped constraint, over any cluster C and any color c. Our algorithm, provably, has an additive violation Δ of at most 2 point. We also evaluate the additive violation of the output of the greedy algorithm and random.

5.2.3 Implementation details and parameters of the algorithm. We now describe the main parameters of the algorithm in Section 3. The algorithm takes in input k, α, representing the number of centers allowed and parameter of the α-capped constraint. To find a small λ for which the polytope $P(\lambda, \alpha)$ gives a feasible solution, instead of binary search, we use following method. We obtain a lower bound on the cost the clustering by running the greedy k-center algorithm and using $\frac{\lambda'}{2}$ as a lower bound, where λ' is the cost of the solution found (this is provably a lower bound of the cost for our problem). We also bound the maximum distance of two points by λ'' (e.g., by using 2 times the maximum distance of a fixed point to any other point) and iterate over a grid Λ that is exponentially increasing by a $(1+\epsilon)$ multiplicative factor between these two extremes,

$$\Lambda = \left\{ \frac{\lambda'}{2}, \frac{\lambda'}{2}(1+\epsilon), \frac{\lambda'}{2}(1+\epsilon)^2, \ldots, \lambda'' \right\},$$

to find the smallest feasible λ. Notice that a solution is found unless the problem is infeasible (i.e., α is lower than the maximum fraction of points of a color). This allows us to check the LP feasibility with lower λ's first, which is better since checking feasibility becomes computationally more expensive as λ increases.

Algorithm 4 FasterAlgorithm($\mathcal{I} = (D, F, d, k), \epsilon, m$).

$\lambda'' \leftarrow \max_{j \in D} d(i_0, j)$ for arbitrary $i_0 \in D$
$((F', \sigma'), \lambda') \leftarrow$ Greedy-k-center($\mathcal{I} = (D, F, d, k)$)
$((F^c, \sigma^c), \lambda^c) \leftarrow$ Greedy-k-center($\mathcal{I}^c = (D, F, d, m * k)$)
for $\lambda \in \{\frac{\lambda'}{2}, \frac{\lambda'}{2}(1+\epsilon), \frac{\lambda'}{2}(1+\epsilon)^2, \ldots, 2 * \lambda''\}$ **do**
    $(F^s, \sigma^s) \leftarrow$ Fair-k-center($\mathcal{I}' = (D, F^c, d, k), \lambda$).
    if (F^s, σ^s) is non-empty **then return** (F^s, σ^s)

Finally, to speed-up the computation, we restrict the variables y_i,x_{ij} that we create to be non-zero only for $i \in F' \subseteq F$ where F' is a core-set of the dataset, obtained by running the greedy algorithm to select $m \times k$ facilities. Notice that using $m \geq 1$ results, provably, in a constant factor approximation algorithm. We evaluate the effect of $\epsilon = 0.1, 0.5$, and experiment with $m \geq 2$.

All our computations are run, independently, each on a single machine, from a proprietary Cloud, using Google's Linear Optimization Package (GLOP) as our LP solver, and a maximum flow solver in C++. Both packages are available in Google's OR tools.[6]

5.3 Experimental results

Comparison with the baselines. In Table 2, we report, for various α factors, a comparison of the quality of the output of our algorithm with that of the baselines. In this table, we fix the parameters: $k = 25$, $\epsilon = 0.1$, $m = 2$ and show results for all datasets and representative α's that are close to the maximum color ratio of a color in each dataset (there is no feasible solution for α's lower than this ratio).

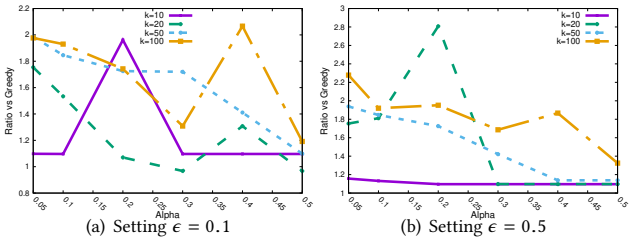
First, we evaluate the ratio of the cost (i.e., the maximum distance of a point to its center) of the solution obtained by our algorithm to that of the greedy algorithm. Notice that in all datasets our algorithm reports a cost that is relatively close to the unconstrained greedy algorithm and is usually between +10% worse and up to 2x worse. Interestingly, despite the fact that the unconstrained problem can have a much better optimum cost, we can sometimes obtain costs that are at most 10–50% larger than of the unconstrained solution (which in turn is lower than the actual optimum value for our problem). This result is better than that predicted by the worst-case theoretical analysis (where we show a 3x factor). This improvement occurs even for α very close to the strongest possible representation constraint for which there is a solution.

In Table 2, we also evaluate the maximum additive violation of the color cap constraint for our algorithms as well as the baselines. As proved formally, the maximum additive violation for our algorithm (Δ) is at most 2 for general α's (and 1 for the case of integer $1/\alpha$). We observe interestingly that it is always 1 in our

[6] https://developers.google.com/optimization/

Dataset	α	Cost vs Greedy	Cost vs Random	Δ	Δ_G	Δ_{Rand}
4area	0.45	+62%	+50%	1	32	660
	0.50	+67%	+55%	1	19	552
	0.60	+62%	+50%	1	6	338
	0.70	+64%	+52%	0	2	124
	0.80	+64%	+52%	0	0	0
query	0.07	+6%	+7%	1	132	66
	0.08	+6%	+7%	1	9	46
	0.09	+6%	+7%	0	7	26
	0.10	+6%	+7%	0	4	6
reuters	0.02	+80%	+44%	1	35	38
	0.05	+75%	+40%	1	29	35
	0.10	+53%	+22%	1	24	29
	0.20	+7%	-15%	1	17	18
	0.30	-3%	-23%	1	15	10
	0.40	+31%	+4%	0	12	8
	0.50	-3%	-23%	0	9	6
victorian	0.05	+109%	+26%	1	62	57
	0.10	+45%	-13%	1	56	38
	0.20	+39%	-17%	1	43	9
	0.30	+63%	-2%	1	30	0
	0.40	+45%	-13%	1	17	0
	0.50	+45%	-13%	0	10	0

Table 2: Comparison of the cost and maximum additive violation of representation constraint for our algorithm, as well as the baselines, over various datasets and α factors, for $k = 25$, $\epsilon = 0.1$, $m = 2$. We report the ratio of the cost of our algorithm's solution with respect to both the greedy algorithm (Cost vs Greedy) and the random baseline (Cost vs Random); the maximum additive violation for our algorithm (Δ), the maximum additive violation of the greedy algorithm (Δ_G), and of the random baseline (Δ_{Rand}).



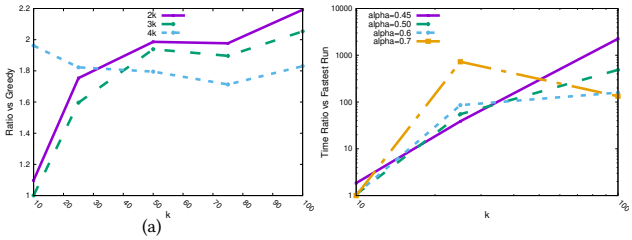
(a) Setting $\epsilon = 0.1$ (b) Setting $\epsilon = 0.5$

Figure 3: Cost of the solution vs Greedy baseline for various α, k over the reuters dataset, using $\epsilon = 0.1, 0.5$, $m = 2$.

experiments. Note instead that the baselines, which do not take into account the constraint, can incur very large additive violations of up to hundreds of points. This result confirms the importance of using algorithms specifically designed for this problem.

Effect of the parameters. We now study more in detail the effect of the main parameters k, α, ϵ, m on the quality of the clustering.

Figures 3(a) and 3(b) show the ratio of the cost of the solution over the cost of the greedy baseline, for various α ranges, and distinct k's, in the reuters dataset. Here, we compare the setting $\epsilon = 0.1$ (Figure 3(a)) and $\epsilon = 0.5$ (Figure 3(b)). Notice how the approximation ratio (over greedy) is always $\lessapprox 2$ for the $\epsilon = 0.1$ case and $\lessapprox 3$ for $\epsilon = 0.5$ case. As is expected, notice that larger α's are associated with lower cost ratios (it is easier to find a low cost solution with higher α). Finally, despite the pattern being less strong, we observe generally larger ratios for larger k's.



(a)

Figure 4: (a) Cost of the solution vs Greedy baseline for various k and m over the reuters dataset, using $\alpha = 0.05$, $\epsilon = 0.1$. (b) Time vs k and α for 4area dataset, $\epsilon = 0.5$, $m = 2$. We report the ratio of running time of a given instance over the the fastest instance.

In Figure 4(a) we evaluate the effect of the m factor used in the core-set to reduce the number of y_i's variables to $m \times k$. Notice how generally larger m's are associated with lower cost (ratio), but the algorithm obtains good results even with $m = 2$, allowing to use small LP instances in our algorithm.

Time. In Figure 4(b) we show how the running time is affected by k and α. As expected, larger k's correspond to increased running times. Similarly, larger α's mostly correspond to lower running time because it is easier to find a solution with larger α and hence fewer $\lambda \in \Lambda$ need to be evaluated to find a non-empty $\mathcal{P}(\lambda, \alpha)$.

6 HARDNESS

In this section, we complement our algorithmic results by proving a factor-2 approximation hardness for minimizing the k-center cost of a α-capped clustering, of arbitrary number of cluster, for $\alpha \in (0, 0.5]$. This shows the hardness of α-capped clustering, with k-center objective, even allowing arbitrary many clusters.

As in [9], we use a reduction from the t-Star-Decomposition problem defined as follows. Given an undirected n-vertex graph $G = (V, E)$, and a positive integer t, can V be partitioned into pairwise disjoint subsets $V_1, \ldots, V_{n/t}$ so that $|V_i| = t$ and $G[V_i]$ contains a star of size t, i.e., a center and $t-1$ leaves? Two well-known special cases of t-Star-Decompositionare the case $t = 2$ (finding a perfect matching) and the case $t = 3$ also known as $P3$-decomposition (finding a partition into connected triplets). Since a perfect matching can be found in polynomial time, t-Star-Decompositionis tractable for $t = 2$. Kirkpatrick and Hell [23] showed that t-Star-Decompositionis NP-hard for $t \geq 3$. t-Star-Decompositionremains NP-hard [11] even if the graph is planar and bipartite, for any $t \geq 3$. In our proofs we will use that the problem is NP-hard.

Our reduction starts from input G of a t-Star-Decomposition instance, and defines a set D of points in a metric space with distance function $d(\cdot, \cdot)$ and a color assignment $c(j)$ for each point $j \in D$. More precisely, we construct a graph $G' = (D, E')$ and define the metric space to be the shortest path metric where edges have unit length. Before proceeding to the main hardness result, we explain

how graph G' is constructed in polynomial time from the bipartite graph $G = (V_1 \cup V_2, E)$ input of t-Star-Decomposition. In the following we use the word point and vertex interchangeably.

Construction of the graph G'. The construction of G' depends on the solution to the following system of linear equations:

$$2t_r + 1t_b = |V_1| \qquad \text{and} \qquad 1t_r + 2t_b = |V_2| \tag{8}$$

Since this is a system of two equations in two variables, and the determinant of the system is non-zero, there exists a unique solution (t_r, t_b). If the unique solution has at least a variable that is not a non-negative integer, we construct G' as a trivial instance with no fair coverage (say one `red` node). For the rest of the construction we assume we are in the case that t_r, t_b are both non-negative integers.

First we define the construction for the $\alpha = \frac{1}{2}$, case then we show how to extend this to the $\alpha = \frac{1}{2+t}$ case for any integer $t > 0$. In the $\alpha = \frac{1}{2}$ case, the construction proceeds as follows. The graph $G' = (V', E')$ has four layers of nodes L_1, L_2, L_3, L_4, where each layer L_i consists of two disjoint sets R_i, B_i of respectively of color `red` and `blue`. The layer L_1 has a 1-to-1 correspondence with nodes in V. More precisely, L_1 consists of $R_1 \equiv V_1$ and $B_1 \equiv V_2$, corresponding to the two sides of the graphs G and two nodes in L_1 are connected in E' iff their equivalent nodes are connected in E. Then, L_2 consists of R_2, B_2 such that $|R_2| = |R_1|, |B_2| = |B_1|$. In E', there is a matching between each node in R_2 (resp. B_2), and a node in R_1 (resp. B_1). Now let $u_b = |B_2| - t_r$ and $u_r = |R_2| - t_b$. Notice that from the Equations (8) u_b, u_r are non-negative integers. Layer L_3 has components B_3, R_3 of size $|B_3| = u_b$, $R_3 = u_r$ and E' contains a complete bipartite graphs between sides R_2,R_3 and another complete bipartite graph between sides B_2, B_3. Finally layer L_4 consists of R_4, B_4 such that $|R_4| = 2|B_3|$ and $|B_4| = 2|R_3|$ and each node in R_3 is connected with exactly two nodes in B_4 (resp. each node in B_3 is connected with exactly two nodes in R_4). This completes the construction for the $\alpha = \frac{1}{2}$ case, for the general $\alpha = \frac{1}{2+t}$ we add to each layer L_2 and L_4, t disjoint sets C_i^t ($i = 2, 4$) such that all nodes in C_i^t have color c_t (distinct from `red` and `blue`). For each t, $|C_2^t| = 2(t_r + t_b)$ and C_2^t is further subdivided in two disjoint parts $C_{2,r}^t$, $C_{2,b}^t$ such that $|C_{2,b}^t| = 2t_b$, $|C_{2,r}^t| = 2t_r$, and $B_1, C_{2,b}^t$ for a complete bipartite graph (reps. $R_1, C_{2,r}^t$ form a complete bipartite graph). Finally for each t, $|C_4^t| = 2|L_3|$ and each node in L_3 is connected with exactly 2 nodes in C_4^t.

The following states our main hardness result for $\alpha \in (0, 0.5]$.

Theorem 6.1. *It is NP-hard to approximate the α-capped clustering with k-center objective with $\alpha \in (0, 0.5]$ within a factor better than 2.*

The theorem follows from the following two lemmas, whose proofs are deferred to the extended version of the paper.

Lemma 6.2. *Fix $t \geq 0$ integer. Suppose the bipartite graph G admits a t-Star-Decomposition, then G' has a $\frac{1}{2+t}$-capped clustering of k-center cost 1.*

Lemma 6.3. *Fix $t \geq 0$ integer. If there exists a solution of k-center cost at most 2 to $\frac{1}{2+t}$-capped clustering of G', then the bipartite graph G admits a t-Star-Decomposition.*

7 CONCLUSIONS

Clustering with color constraints is an algorithmic take on ensuring balance and fairness in applications. In this paper we addressed capped clustering, which is the problem of finding the best clustering where no cluster has an over-represented color. We obtained provably good algorithms for this problem; our experiments show that the algorithms are effective on different real-world datasets. While our general algorithm is based on solving an LP, it can be challenging for large number of points. It is an interesting question to develop a combinatorial algorithm for the general case that can scale to large datasets. It is also interesting to improve the bounds guaranteed by our algorithms and extend them to other clustering objectives such as k-means and k-median.

REFERENCES

[1] Arturs Backurs, Piotr Indyk, Krzysztof Onak, Baruch Schieber, Ali Vakilian, and Tal Wagner. 2019. Scalable Fair Clustering. In *ICML*.
[2] Sugato Basu, Ian Davidson, and Kiri Wagstaff. 2008. *Constrained Clustering: Algorithms, Applications and Theory.* CRC Press.
[3] Suman K Bera, Deeparnab Chakrabarty, and Maryam Negahbani. 2019. *Fair Algorithms for Clustering.* Technical Report 1901.02393. arXiv.
[4] Ioana O. Bercea, Martin Gross, Samir Khuller, Aounon Kumar, Clemens Rösner, Daniel R. Schmidt, and Melanie Schmidt. 2018. *On the cost of essentially fair clusterings.* Technical Report 1811.10319. arXiv.
[5] Toon Calders and Sicco Verwer. 2010. Three naive Bayes approaches for discrimination-free classification. *DMKD* 21, 2 (2010), 277–292.
[6] L Elisa Celis, Lingxiao Huang, and Nisheeth K Vishnoi. 2018. Multiwinner Voting with Fairness Constraints.. In *IJCAI*. 144–151.
[7] L. Elisa Celis, Damian Straszak, and Nisheeth K. Vishnoi. 2018. Ranking with Fairness Constraints. In *ICALP*. 28:1–28:15.
[8] Miriam Cha, Youngjune Gwon, and HT Kung. 2017. Language modeling by clustering with word embeddings for text readability assessment. In *CIKM*. 2003–2006.
[9] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. 2017. Fair clustering through fairlets. In *NIPS*. 5029–5037.
[10] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *ITCS*. 214–226.
[11] Martin E Dyer and Alan M Frieze. 1985. On the complexity of partitioning graphs into connected subgraphs. *Discrete Applied Mathematics* 10, 2 (1985), 139–153.
[12] Alessandro Epasto, Mohammad Mahdian, Vahab S. Mirrokni, and Song Zuo. 2018. Incentive-Aware Learning for Large Markets. In *WWW*. 1369–1378.
[13] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *KDD*. 259–268.
[14] Benjamin Fish, Jeremy Kun, and Ádám D Lelkes. 2016. A confidence-based approach for balancing fairness and accuracy. In *SDM*. 144–152.
[15] Teofilo F Gonzalez. 1985. Clustering to minimize the maximum intercluster distance. *TCS* 38 (1985), 293–306.
[16] Abdulmecit Gungor. 2018. Fifty Victorian Era Novelists Authorship Attribution Data. (2018).
[17] Dorit S Hochbaum and David B Shmoys. 1985. A best possible heuristic for the k-center problem. *Mathematics of operations research* 10, 2 (1985), 180–184.
[18] Wen-Lian Hsu and George L Nemhauser. 1979. Easy and hard bottleneck location problems. *Discrete Applied Mathematics* 1, 3 (1979), 209–215.
[19] Anil K Jain. 2010. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters* 31, 8 (2010), 651–666.
[20] Matthew Joseph, Michael Kearns, Jamie H Morgenstern, and Aaron Roth. 2016. Fairness in learning: Classic and contextual bandits. In *NIPS*. 325–333.
[21] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. 2012. Fairness-aware classifier with prejudice remover regularizer. In *PKDD*. 35–50.
[22] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. 2011. Fairness-aware learning through regularization approach. In *ICDMW*. 643–650.
[23] David G. Kirkpatrick and Pavol Hell. 1983. On the complexity of general graph factor problems. *SIAM J. Comput.* 12, 3 (1983), 601–609.
[24] Jian Li, Ke Yi, and Qin Zhang. 2010. Clustering with Diversity. In *ICALP*. 188–200.
[25] B. Perozzi, R. Al-Rfou, and S. Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *KDD*. 701–710.
[26] Clemens Rösner and Melanie Schmidt. 2018. Privacy preserving clustering with constraints. In *ICALP*. 96:1–96:14.
[27] Ke Yang and Julia Stoyanovich. 2017. Measuring fairness in ranked outputs. In *SSDBM*. 22.