Clustering What Matters in Constrained Settings

Improved Outlier to Outlier-Free Reductions

Ragesh Jaiswal[1] ✉ 📧
CSE, IIT Delhi, India

Amit Kumar ✉ 📧
CSE, IIT Delhi, India

─── **Abstract** ───

Constrained clustering problems generalize classical clustering formulations, e.g., k-median, k-means, by imposing additional constraints on the feasibility of a clustering. There has been significant recent progress in obtaining approximation algorithms for these problems, both in the metric and the Euclidean settings. However, the outlier version of these problems, where the solution is allowed to leave out m points from the clustering, is not well understood. In this work, we give a general framework for reducing the outlier version of a constrained k-median or k-means problem to the corresponding outlier-free version with only $(1 + \varepsilon)$-loss in the approximation ratio. The reduction is obtained by mapping the original instance of the problem to $f(k, m, \varepsilon)$ instances of the outlier-free version, where $f(k, m, \varepsilon) = \left(\frac{k+m}{\varepsilon} \right)^{O(m)}$. As specific applications, we get the following results:

- First FPT (*in the parameters k and m*) $(1 + \varepsilon)$-approximation algorithm for the outlier version of capacitated k-median and k-means in Euclidean spaces with *hard* capacities.
- First FPT (*in the parameters k and m*) $(3 + \varepsilon)$ and $(9 + \varepsilon)$ approximation algorithms for the outlier version of capacitated k-median and k-means, respectively, in general metric spaces with *hard* capacities.
- First FPT (*in the parameters k and m*) $(2 - \delta)$-approximation algorithm for the outlier version of the k-median problem under the Ulam metric.

Our work generalizes the results of Bhattacharya et al. and Agrawal et al. to a larger class of constrained clustering problems. Further, our reduction works for arbitrary metric spaces and so can extend clustering algorithms for outlier-free versions in both Euclidean and arbitrary metric spaces.

1 Introduction

Center-based clustering problems such as k-median and the k-means are important data processing tasks. Given a metric D on a set of n points $\mathcal{X}$ and a parameter k, the goal here is to partition the set of points into k *clusters*, say $C_1, \ldots, C_k$, and assign the points in each cluster to a corresponding *cluster center*, say $c_1, \ldots, c_k$, respectively, such that the objective $\sum_{i=1}^{k} \sum_{x \in C_i} D(x, c_i)^z$ is minimized. Here z is a parameter which is 1 for k-median and 2 for k-means. The *outlier* version of these problems is specified by another parameter m, where a solution is allowed to leave out up to m points from the clusters. Outlier versions capture settings where the input may contain a few highly erroneous data points. Both the

---

outlier and the outlier-free versions have been well-studied in the literature with constant factor approximations known for both the k-means and the k-median problem [3, 4, 12]. In addition, fixed-parameter tractable (FPT) $(1 + \varepsilon)$-approximation algorithms are known for these problems in the Euclidean setting [26, 19, 8]: the running time of such algorithms is of the form $f(k, m, \varepsilon) \cdot poly(n, d)$, where $f()$ is an exponential function of the parameters k, m, ε and d denotes the dimensionality of the points.

A more recent development in clustering problems has been the notion of *constrained clustering*. A constrained clustering problem specifies additional conditions on a feasible partitioning of the input points into k clusters. For example, the r-gathering problem requires that each cluster in a feasible partitioning must contain at least r data points. Similarly, the well-known *capacitated* clustering problem specifies an upper bound on the size of each cluster. Constrained clustering formulations can also capture various types of *fairness* constraints: each data point has a *label* assigned to it, and we may require upper or lower bounds on the number (or fraction) of points with a certain label in each cluster. Table 1 in the Appendix gives a list of some of these problems. FPT (in the parameter k) constant factor approximation algorithms are known for a large class of these problems (see Table 2 in the Appendix).

It is worth noting that constrained clustering problems are distinct from outlier clustering: the former restricts the set of feasible partitioning of input points, whereas the latter allows us to reduce the set of points that need to be partitioned into clusters. There has not been much progress on constrained clustering problems in the outlier setting (also see [25] for unbounded integrality gap for the natural LP relaxation for the outlier clustering versions). In this work, we bridge this gap between the outlier and the outlier-free versions of constrained clustering problems by giving an *almost approximation-preserving* reduction from the former to the latter. As long as the parameters of interest (i.e., k, m) are small, the reduction works in polynomial time. Using our reduction, an FPT α-approximation algorithm for the outlier-free version of a constrained clustering problem leads to an FPT $(\alpha + \varepsilon)$-approximation algorithm for the outlier version of the same problem. For general metric spaces, this implies the first FPT constant-approximation for outlier versions of several constrained clustering problems; and similarly, we get new FPT $(1 + \varepsilon)$-approximation algorithms for several outlier constrained clustering problems –see Table 2 in the Appendix for the precise details.

This kind of FPT approximation preserving reduction in the context of Euclidean k-means was first given by [8] using a sampling-based approach. [20] extended the sampling ideas of [8] to general metric spaces but did not give an approximation-preserving reduction. [2] gave a reduction for general metric spaces using a coreset construction. In this work, we use the sampling-based ideas of [8] to obtain an approximation-preserving reduction from the outlier version to the outlier-free version with improved parameters over [2]. Moreover, our reduction works for most known constrained clustering settings as well.

1.1 Preliminaries

We give a general definition of a constrained clustering problem. For a positive integer k, we shall use $[k]$ to denote the set $\{1, \ldots, k\}$. Let $(\mathcal{X}, D)$ denote the metric space with distance function D. For a point x and a subset S of points, we shall use $D(x, S)$ to denote $\min_{y \in S} D(x, y)$. The set $\mathcal{X}$ contains subsets F and X: here X denotes the set of input points and F is the set of points where a center can be located. An outlier constrained clustering problem is specified by the following parameters and functions:

- k: the number of clusters.
- m: the number of points which can be left out from the clusters.

- a function check: given a partitioning $X_0, X_1, \ldots, X_k$ of X (here X_0 is the set of outliers) and centers $f_1, \ldots, f_k$, each lying in the set F, the function check$(X_0, X_1, \ldots, X_k, f_1, \ldots, f_k)$ outputs 1 iff this is a feasible clustering. For example, in the r-gathering problem, the check$(X_0, X_1, \ldots, X_k, f_1, \ldots, f_k)$ outputs 1 iff $|X_i| \geq r$ for each $i \in [k]$. The check function depends only on the cardinality of the sets $X_1, \ldots, X_k$ and the locations $f_1, \ldots, f_k$. This already captures many of the constrained clustering problems. Our framework also applies to the more general labelled version (see details below).

- a cost function cost: given a partitioning $X_0, X_1, \ldots, X_k$ of X and centers $f_1, \ldots, f_k$,

$$\mathsf{cost}(X_0, X_1, \ldots, X_k, f_1, \ldots, f_k) := \sum_{i \in [k]} \sum_{x \in X_i} D^z(x, f_i),$$

  where z is either 1 (the outlier constrained k-median problem) or 2 (the outlier constrained k-means problem).

Given an instance $\mathcal{I} = (X, F, k, m, \mathsf{check}, \mathsf{cost})$ of an outlier constrained clustering problem as above, the goal is to find a partitioning $X_0, X_1, \ldots, X_k$ of X and centers $f_1, \ldots, f_k \in F$ such that $|X_0| \leq m$, check$(X_0, X_1, \ldots, X_k, f_1, \ldots, f_k)$ is 1 and cost$(X_0, X_1, \ldots, X_k, f_1, \ldots, f_k)$ is minimized. The outlier-free constrained clustering problem is specified as above, except that the parameter m is 0. For the sake of brevity, we leave out the parameter m and the set X_0 while defining the instance $\mathcal{I}$, and functions check and cost.

We shall also consider a more general class of constrained clustering problems, where each input point is assigned a *label*. In other words, an instance $\mathcal{I}$ of such a problem is specified by a tuple $(X, F, k, m, \sigma, \mathsf{check}, \mathsf{cost})$, where $\sigma : X \to L$ for a finite set L. Note that the check function may depend on the function σ. For example, σ could assign a label "red" or "blue" to each point in X and the check function would require that each cluster X_i should have an equal number of red and blue points. In addition to the locations $f_1, \ldots, f_k$, the check$(X_1, \ldots, X_k, f_1, \ldots, f_k, \sigma)$ function also depends on $|\sigma^{-1}(l) \cap X_j|$ for each $l \in L, j \in [k]$, i.e., the number of points with a particular label in each of the clusters. Indirectly, this also implies that the check function can impose conditions on the labels of the outliers points. For example, the colorful k-median problem discussed in [2] has the constraint that m_i clients from the label type i should be designated as outliers, given that every client has a unique label. Table 1 in the Appendix gives a description of some of these problems.

We shall use the approximate triangle inequality, which states that for $z \in \{1, 2\}$ and any three points $x_1, x_2, x_3 \in \mathcal{X}$,

$$D^z(x_1, x_3) \leq z \left(D^z(x_1, x_2) + D^z(x_2, x_3) \right). \tag{1}$$

1.2 Our results

Our main result reduces the outlier constrained clustering problem to the outlier-free version. In our reduction, we shall also use approximation algorithms for the (unconstrained) k-median and k-means problems. We assume we have a constant factor approximation algorithm for these problems[2]: let $\mathcal{C}$ denote such an algorithm with running time $T_{\mathcal{C}}(n)$ on an input of size n. Note that $\mathcal{C}$ would be an algorithm for the k-means or the k-median problem depending on whether $z = 1$ or 2 in the definition of the cost function.

---

[2] Several such constant factor approximation algorithms exist [3, 4, 12].

▶ **Theorem 1** (Main Theorem). *Consider an instance $\mathcal{I} = (X, F, k, m, \mathsf{check}, \mathsf{cost})$ of an outlier constrained clustering problem. Let $\mathcal{A}$ be an α-approximation algorithm for the corresponding outlier-free constrained clustering problem; let $T_{\mathcal{A}}(n)$ be the running time of $\mathcal{A}$ on an input of size n. Given a positive $\varepsilon > 0$, there is an $\alpha(1 + \varepsilon)$-approximation algorithm for $\mathcal{I}$ with running time $T_{\mathcal{C}}(n) + q \cdot T_{\mathcal{A}}(n) + O\left(n \cdot (k + \frac{m^{z+1} \log m}{\varepsilon^z})\right) + O\left(qm^2(k+m)^3\right)$, where n is the size of $\mathcal{I}$ and $q = f(k, m, \varepsilon) = \left(\frac{k+m}{\varepsilon}\right)^{O(m)}$, and $z = 1$ or 2 depending on the* cost *function (i.e., $z = 1$ for k-median objection and $z = 2$ for k-means objective).*

The above theorem implies that as long as there is an FPT or polynomial-time approximation algorithm for the constrained, outlier-free k-median or k-means clustering problem, there is an FPT approximation algorithm (with almost the same approximation ratio) for the corresponding outlier version. We prove this result by creating q instances of the outlier-free version of $\mathcal{I}$ and picking the best solution on these instances using the algorithm $\mathcal{A}$. We also extend the above result to the labelled version:

▶ **Theorem 2** (Main Theorem: labelled version). *Consider an instance $\mathcal{I} = (X, F, k, m, \sigma, \mathsf{check}, \mathsf{cost})$ of an outlier constrained clustering problem with labels on input points. Let $\mathcal{A}$ be an α-approximation algorithm for the corresponding outlier-free constrained clustering problem; let $T_{\mathcal{A}}(n)$ be the running time of $\mathcal{A}$ on an input of size n. Given a positive $\varepsilon > 0$, there is an $\alpha(1 + \varepsilon)$-approximation algorithm for $\mathcal{I}$ with running time $T_{\mathcal{C}}(n) + q \cdot T_{\mathcal{A}}(n) + O\left(n \cdot (k + \frac{m^{z+1} \log m}{\varepsilon^z})\right) + O\left(q\ell m^2(k+m)^3\right)$, where n is the size of $\mathcal{I}$, $q = f(k, m, \varepsilon) = \left(\frac{(k+m)\ell}{\varepsilon}\right)^{O(m)}$ with ℓ being the number of distinct labels, and $z = 1$ or 2 depending on the* cost *function (i.e., $z = 1$ for k-median objection and $z = 2$ for k-means objective).*

The algorithms given in Theorem 1 and Theorem 2 are randomized algorithms that guarantee the stated approximation factor with high probability. The consequences of our results for specific constrained clustering problems are summarized in Table 2 in the Appendix. We give the results of related works [8, 20, 2] in the same table to see the contributions of this work. Our contributions can be divided into two main categories:

1. *Matching the best-known result*: This can be further divided into two categories:
   a. *Matching results of [2]*: [2] gives an outlier to outlier-free reduction. We also give such a reduction using a different technique with better parameters. This means that we match all the results of [2], which includes problems such as the classical k-median/means problems, the Matroid k-median problem, the colorful k-median problem, and k-median in certain special metrics. See rows 2-6 in Table 2 given in the Appendix.
   b. *Matching results of [20]*: [20] gives FPT approximation algorithms for certain constrained problems on which the coreset-based approach of [2] is not known to work. See the last row of Table 2. [20] gives algorithms for outlier and outlier-free versions with the same approximation guarantee. Since the best outlier-free approximation is also from [20], our results currently only match the approximation guarantees of [20]. However, if there is an improvement in any of these problems, our results will immediately beat the known outlier results of [20].
2. *Best known results*: Since our results hold for a larger class of constrained problems than earlier works, there are certain problems for which our results give the best-known FPT approximation algorithm. The list includes capacitated k-median/k-means with hard capacities in general metric and Euclidean spaces. It also includes the k-median problem in the Ulam metric. A recent development in the Ulam k-median problem [11] has broken

the 2-approximation barrier. Our reduction allows us to take this development to the outlier setting as well. The outlier-free results from which our best results are derived using our reduction are given in Table 2 (see rows 7-9) given in the Appendix.

1.3   Comparison with earlier work

As discussed earlier, the idea of a reduction from an outlier clustering problem to the corresponding outlier-free version in the context of the Euclidean k-means problem was suggested by [8] using a D^2-sampling based idea. [20] used the sampling ideas to design approximation algorithms for the outlier versions of various constrained clustering problems. However, the approximation guarantee obtained by [20] was limited to $(3 + \varepsilon)$ for a large class of constrained k-median and $(9 + \varepsilon)$ for the constrained k-means problems, and it was not clear how to extend these techniques to get improved guarantees. As a result, their techniques could not exploit the recent developments by [14] in the design of $(1 + 2/e + \varepsilon)$ and $(1 + 8/e + \varepsilon)$ FPT approximation algorithms for the classical outlier-free k-median and k-means problems respectively in general metric spaces. [2] gave an outlier-to-outlier-free reduction, making it possible to extend the above-mentioned FPT approximation guarantees for the outlier-free setting to the outlier setting.

The reduction of [2] is based on the coreset construction by [13] using uniform sampling. A coreset for a dataset is a weighted set of points such that the clustering of the coreset points with respect to any set of k centers is the same (within a $1 \pm \varepsilon$ factor) as that of the original set points. The coreset construction in [13] starts with a set C of centers that give constant factor approximation. They consider $O(\log n)$ "ring" around these centers, uniformly sample points from each of these rings, and set the weight of the sampled points appropriately. The number of sampled points, and hence the size of the coreset, is $\left(\frac{|C|\log n}{\varepsilon}\right)^2$. [2] showed that when starting with $(k+m)$ centers that give a constant approximation to the classical $(k+m)$-median problem, the coreset obtained as above has the following additional property: for any set of k centers, the clustering cost of the original set of points excluding m outliers is same (again, within $1 \pm \varepsilon$ factor) as that of the coreset, again allowing for exclusion of a subset of m points from it. This means that by trying out all m subsets from the coreset, we ensure that at least one subset acts as a good outlier set. Since the coreset size is $\left(\frac{(k+m)\log n}{\varepsilon}\right)^2$, the number of outlier-free instances that we construct is $\left(\frac{(k+m)\log n}{\varepsilon}\right)^{O(m)}$. Using $(\log n)^{O(m)} = \max\{m^{O(m)}, n^{O(1)}\}$, this is of the form $f(k, m, \varepsilon) \cdot n^{O(1)}$ for a suitable function f. At this point, we note the first quantitative difference from our result. In our algorithm, we save the $(\log n)^{O(m)}$ factor, which also means that the number of instances does not depend on the problem size n. Further, a coreset-based construction restricts the kind of problems it can be applied to. The coreset property that the cost of original points is the same as that of the weighted cost of coreset points holds when points are assigned to the closest center (*i.e., the entire weight of the coreset goes to the closest center*).[3] This works for the classical unconstrained k-median and k-means problems (as well as the few other settings considered in [2]). However, for several constrained clustering problems, it may not hold that every point is assigned to the closest center. There have been some recent developments [5, 10] in designing coresets for constrained clustering settings. However, they have not been shown to apply to the outlier setting. Another recent work [22] designs coresets for the outlier setting, but like [2], it has limited scope and has not been shown to extend for most constrained settings. Our D^z-sampling-based technique has the advantage that instead of running the outlier-free algorithm on a coreset as in [2], it works directly with the dataset.

---

[3] The reason is how Haussler's lemma is applied to bound the cost difference.

That is, we run the outlier-free algorithm on the dataset (after removing outlier candidates). This also makes our results helpful in weighted settings (e.g., see [11]) where the outlier-free algorithm is known to work only for unweighted datasets – note a coreset is a weighted set).

Recent independent work.   In recent and independent work, [17] design similar approximation preserving reductions for a restricted class of constrained clustering settings, namely capacitated clustering and (α, β)-fair clustering. Further, their results are obtained by extending coreset based ideas of [2].

1.4   Our Techniques

In this section, we give a high-level description of our algorithm. Let $\mathcal{I}$ denote an instance of outlier constrained clustering on a set of points X and $\mathcal{O}$ denote an optimal solution to $\mathcal{I}$. The first observation is that the optimal cost of the outlier-free and unconstrained clustering with $k + m$ centers on X is a lower bound on the cost of $\mathcal{O}$ (Claim 1). [4] Let C denote the set of these $(k + m)$ centers (we can use any constant factor approximation for the unconstrained version to find C). The intuition behind choosing C is that the centers in $\mathcal{O}$ should be close to C.

Now we divide the set of m outliers in $\mathcal{O}$ into two subsets: those which are far from C and the remaining ones close to C ("near" outliers). Our first idea is to randomly sample a subset S of $O(m \log m)$ points from X with sampling probability proportional to distance (or square of distance) from the set C. This sampling ensures that S contains the far outliers with high probability (Claim 2). We can then iterate over all subsets of S to guess the exact subset of far outliers. Handling the near outliers is more challenging and forms the heart of the technical contribution of this paper.

We "assign" each near outlier to its closest point in C – let $X_{N,j}^{\mathsf{opt}}$ be the set of outliers assigned to c_j. By iterating over all choices, we can guess the cardinality t_j of each of the sets $X_{N,j}^{\mathsf{opt}}$. We now set up a suitable minimum cost bipartite b-matching instance which assigns a set of t_j points to each center c_j. Let $\widehat{X}_j$ be the set of points assigned to c_j. Our algorithm uses $\cup_j \widehat{X}_j$ as the set of near outliers. In the analysis, we need to argue that there is a way of matching the points in $X_{N,j}^{\mathsf{opt}}$ to $\widehat{X}_j$ whose total cost (sum of distances or squared distances between matched points) is small (Lemma 4). The hope is that we can go from the optimal set of outliers in $\mathcal{O}$ to the ones in the algorithm and argue that the increase in cost is small. Since we are dealing with constrained clustering, we need to ensure that this process does not change the size of each of the clusters. To achieve this, we need to further modify the matching between the two sets of outliers (Lemma 5). Finally, with this modified matching, we are able to argue that the cost of the solution produced by the algorithm is close to that of the optimal solution. The extension to the labelled version follows along similar lines.

In the remaining paper, we prove our two main results, Theorem 1 and Theorem 2. The main discussion will be for Theorem 1 since Theorem 2 is an extension of Theorem 1 that uses the same proof ideas. In the following sections, we give the details of our algorithm (Section 2) and its analysis (Section 3). In Section 3, we discuss the extension to the labelled version.

2   Algorithm

In this section, we describe the algorithm for the outlier constrained clustering problem. Consider an instance $\mathcal{I} = (X, F, k, m, \mathsf{check}, \mathsf{cost})$ of this problem. Recall that the parameter $z = 1$ or 2 depends on whether the cost function is like the $k\text{-median}$ or the $k\text{-means}$ objective respectively. In addition, we assume the existence of the following algorithms:

---

[4] This observation was used in both [8] and [2].

- A constant β-factor algorithm $\mathcal{C}$ for the k-median or the k-means problem (depending on $z = 1$ or $z = 2$ respectively): an instance here is specified by a tuple (X', F', k') only, where X' is the set of input points, F' is the set of potential locations for a center, and k' denotes the number of clusters.

- An algorithm $\mathcal{A}$ for the outlier-free version of this problem. An instance here is given by a tuple $(X', F', k, \mathsf{check}, \mathsf{cost})$ where the check and the cost functions are the same as those in $\mathcal{I}$.

- An algorithm $\mathcal{M}$ for the b-matching problem: an instance of the b-matching problem is specified by a weighted bi-partite graph $G = (L, R = \{v_1, \ldots, v_r\}, E)$, with edge e having weight w_e; and a tuple $(t_1, \ldots, t_r)$, where $t_i, i \in [r]$, are non-negative integers. A solution needs to find a subset of E' of E such each vertex of L is incident with at most one edge of E', and each vertex $v_j \in R$ is incident with *exactly* t_j edges of E'. The goal is to find such a set E' of minimum total weight.

We now define D^z-sampling:

▶ **Definition 3.** *Given sets C and X of points, D^z-sampling from X w.r.t. C samples a point $x \in X$, where the probability of sampling x is proportional to $D^z(x, C)$.*

The algorithm is described in Algorithm 1. It first runs the algorithm $\mathcal{C}$ to obtain a set of $(k + m)$ centers C in line 1.2. In line 1.3, we sample a subset S where each point in S is sampled independently using D^z-sampling w.r.t. C. Given a subset Y, we say that a tuple $\boldsymbol{\tau} = (t_1, \ldots, t_{k+m})$ is *valid* if $t_j \geq 0$ for all $j \in [k + m]$, and $\sum_j t_j + |Y| = m$. For each subset Y of size $\leq m$ of S and for each valid tuple $\boldsymbol{\tau}$, the algorithm constructs a solution $(X_0^{(Y, \boldsymbol{\tau})}, X_1^{(Y, \boldsymbol{\tau})}, \ldots, X_k^{(Y, \boldsymbol{\tau})})$, where $X_0^{(Y, \boldsymbol{\tau})}$ denotes the set of outlier points. This is done by first computing the set $X_0^{(Y, \boldsymbol{\tau})}$, and then using the algorithm $\mathcal{A}$ on the remaining points $X \setminus (X_0^{(Y, \boldsymbol{\tau})} \cup Y)$ (line 1.8). To find the set $X_0^{(Y, \boldsymbol{\tau})}$, we construct an instance $\mathcal{I}^{(Y, \boldsymbol{\tau})}$ of b-matching first (line 1.6). This instance is defined as follows: the bipartite graph has the set of $(k + m)$ centers C on the right side and the set of points X on the left side. The weight of an edge between a vertex $v \in C$ and $w \in X$ is equal to $D^z(v, w)$. For each vertex $v_j \in C$, we require that it is matched to exactly t_j points of X. We run the algorithm $\mathcal{M}$ on this instance of b-matching (line 1.7). We define $X_0^{(Y, \boldsymbol{\tau})}$ as the set of points of X matched by this algorithm. Finally, we output the solution of minimum cost (line 1.10).

🟨 **Algorithm 1** Algorithm for outlier constrained clustering.

1.1 **Input:** $\mathcal{I} := (X, F, k, m, \mathsf{check}, \mathsf{cost})$

1.2 Execute $\mathcal{C}$ on the instance $\mathcal{I}' := (X, F, k + m)$ to obtain a set C of $k + m$ centers.

1.3 Sample a set S of $\lceil \frac{4\beta m \log m}{\varepsilon} \rceil$ points with replacement, each using D^z-sampling from X w.r.t. C.

1.4 **for** *each subset $Y \subset S, |Y| \leq m$* **do**

1.5   **for** *each valid tuple $\boldsymbol{\tau} = (t_1, \ldots, t_{k+m})$* **do**

1.6   Construct the instance $\mathcal{I}^{(Y, \tau)}$

1.7   Run $\mathcal{M}$ on $\mathcal{I}^{(Y, \tau)}$ and let $X_0^{(Y, \tau)}$ be the set of matched points in X.

1.8   Run the algorithm $\mathcal{A}$ on the instance $(X \setminus (X_0^{(Y, \tau)} \cup Y), F, k, \mathsf{check}, \mathsf{cost})$.

1.9   Let $(X_1^{(Y, \tau)}, \ldots, X_k^{(Y, \tau)})$ be the clustering produced by $\mathcal{A}$.

1.10 Let $(Y^\star, \boldsymbol{\tau}^\star)$ be the pair for which $\mathsf{cost}(X_1^{(Y, \tau)}, \ldots, X_k^{(Y, \tau)})$ is minimized.

1.11 **Output** $(X_0^{(Y^\star, \tau^\star)}, X_1^{(Y^\star, \tau^\star)}, \ldots, X_k^{(Y^\star, \tau^\star)})$.

---

3   Analysis

We now analyze Algorithm 1. We refer to the notation used in this algorithm. Let $\mathcal{I} = (X, F, k, m, \mathsf{check}, \mathsf{cost})$ be the instance of the outlier constrained clustering problem. Let $\mathsf{opt}(\mathcal{I})$ denote the optimal cost of a solution for the instance $\mathcal{I}$. Assume that the algorithm $\mathcal{C}$ for the unconstrained clustering problem (used in line 1.2) is a β-approximation algorithm. We overload notation and use $\mathsf{cost}_{\mathcal{I}'}(C)$ to denote the cost of the solution C for the instance $\mathcal{I}'$. Observe that the quantity $\mathsf{cost}_{\mathcal{I}'}(C)$ can be computed as follows: each point in X is assigned to the closest point in C, and then we compute the total cost (which could be the k-median or the k-means cost based on the value of the parameter z) of this assignment. We first relate $\mathsf{cost}_{\mathcal{I}'}(C)$ to $\mathsf{opt}(\mathcal{I})$.

▷ **Claim 1.** $\mathsf{cost}_{\mathcal{I}'}(C) \leq \beta \cdot \mathsf{opt}(\mathcal{I})$.

Proof. Let $(X_0, X_1, ..., X_k)$ denote the optimal solution for $\mathcal{I}$, where X_0 denotes the set of m outlier points (without loss of generality, we can assume that the number of outlier points in the optimal solution is exactly m). Let $c_1, \ldots, c_k$ be the centers of the clusters $X_1, \ldots, X_k$ respectively. Consider the solution to $\mathcal{I}'$ consisting of centers $C' := X_0 \cup \{c_1, \ldots, c_k\}$. Clearly, $\mathsf{cost}_{\mathcal{I}'}(C') \leq \mathsf{opt}(\mathcal{I})$ (we have inequality here because the solution $X_1, \ldots, X_k$ may not be a Voronoi partition with respect to $c_1, \ldots, c_k$). Since $\mathcal{C}$ is a β-approximation algorithm, we know that $\mathsf{cost}_{\mathcal{I}'}(C) \leq \beta \cdot \mathsf{cost}_{\mathcal{I}'}(C')$. Combining these two facts implies the desired result. ◁

We now consider an optimal solution for the instance $\mathcal{I}$: let $X_0^{\mathsf{opt}}, X_1^{\mathsf{opt}}, \ldots, X_k^{\mathsf{opt}}$ be the partition of the input points X in this solution, with X_0^{opt} being the set of m outliers. Depending on the distance from C, we divide the set X_0^{opt} into two subsets – X_F^{opt} ("far" points) and X_N^{opt} ("near" points) as follows:

$$X_F^{\mathsf{opt}} := \left\{ x \in X_0^{\mathsf{opt}} \mid D^z(x, C) \geq \frac{\varepsilon \, \mathsf{cost}_{\mathcal{I}'}(C)}{2\beta m} \right\}, \quad X_N^{\mathsf{opt}} := X \setminus X_F^{\mathsf{opt}}.$$

Recall that we sample a set S of $\frac{4\beta m \log m}{\varepsilon}$ clients using D^z-sampling with respect to center set C (line 1.3 in Algorithm 1). Note that the probability of sampling a point x is given by

$$\frac{D^z(x, C)}{\sum_{x' \in X} D^z(x, C)} = \frac{D^z(x, C)}{\mathsf{cost}_{\mathcal{I}'}(C)}. \tag{2}$$

We first show that S contains all the points in X_F^{opt} with high probability.

▷ **Claim 2.** $\mathbf{Pr}[X_F^{\mathsf{opt}} \subseteq S] \geq 1 - 1/m$.

Proof. Inequality Equation (2) shows that the probability of sampling a point $x \in X_F^{\mathsf{opt}}$ is $\frac{D^z(x, C)}{\mathsf{cost}_{\mathcal{I}'}(C)} \geq \frac{\varepsilon}{2\beta m}$. So the probability that x is not present in S is at most $\left(1 - \frac{\varepsilon}{2\beta m} \right)^{\frac{4\beta m \log m}{\varepsilon}} \leq \frac{1}{m^2}$. The desired result now follows from union bound. ◁

For the rest of the analysis, we assume that the event in Claim 2 holds. We now note that the sum of the cost of assigning X_N^{opt} to C is at most $\varepsilon \cdot \mathsf{opt}(\mathcal{I})$.

▷ **Claim 3.** $\sum_{x \in X_N^{\mathsf{opt}}} D^z(x, C) \leq \frac{\varepsilon}{2} \cdot \mathsf{opt}(\mathcal{I})$.

Proof. The claim follows from the following sequence of inequalities:

$$\sum_{x \in X_N^{\mathsf{opt}}} D^z(x, C) < \sum_{x \in X_N^{\mathsf{opt}}} \frac{\varepsilon \, \mathsf{cost}_{\mathcal{I}'}(C)}{2\beta m} \leq \sum_{x \in X_N^{\mathsf{opt}}} \frac{\varepsilon \cdot \mathsf{opt}(\mathcal{I})}{2m} \leq \frac{\varepsilon}{2} \cdot \mathsf{opt}(\mathcal{I}),$$

where the first inequality follows from the definition of X_N^{opt} and the second inequality follows from Claim 1.   ◁

For every point in X_N^{opt}, we identify the closest center in $C = \{c_1, \ldots, c_{m+k}\}$ (breaking ties arbitrarily). For each $j \in [k+m]$, let $X_{N,j}^{\mathsf{opt}}$ be the set of points in X_N^{opt} which are closest to c_j. Let $\hat{t}_j$ denote $|X_{N,j}^{\mathsf{opt}}|$. Consider an iteration of line 1.7–1.9 where $Y = X_F^{\mathsf{opt}}, \boldsymbol{\tau} = (\hat{t}_1, \ldots, \hat{t}_{k+m})$. Observe that $\boldsymbol{\tau}$ is valid with respect to Y because $\sum_{j \in [m+k]} |\hat{t}_j| + |Y| = m$. Let $\widehat{X}_1, \ldots, \widehat{X}_{m+k}$ be the set of points assigned to $c_1, \ldots, c_{m+k}$ respectively by the algorithm $\mathcal{M}$. Intuitively, we will like to construct a solution where the set of outliers is given by $\widehat{X} := X_F^{\mathsf{opt}} \cup \widehat{X}_1 \cup \cdots \cup \widehat{X}_{m+k}$. We now show that the set $\widehat{X}$ is "close" to X_0^{opt}, the set of outliers in the optimal solution. In order to do this, we set up a bijection $\mu : X_0^{\mathsf{opt}} \to \widehat{X}$, where μ restricted to X_F^{opt} is identity, and μ restricted to any of the sets $X_{N,j}^{\mathsf{opt}}$ is a bijection from $X_{N,j}^{\mathsf{opt}}$ to $\widehat{X}_j$. Such a function μ is possible because for each $j \in [m+k]$, $|X_{N,j}^{\mathsf{opt}}| = |\widehat{X}_j| = \hat{t}_j$. We now prove this closeness property.

▶ **Lemma 4.** $\sum_{x \in X_0^{\mathsf{opt}}} D^z(x, \mu(x)) \leq \varepsilon \cdot z \cdot \mathsf{opt}(\mathcal{I})$.

Proof. We first note a useful property of the solution given by the algorithm $\mathcal{M}$. One of the possible solutions for the instance $\mathcal{I}^{(Y,\boldsymbol{\tau})}$ could have been assigning $X_{N,j}^{\mathsf{opt}}$ to the center c_j. Since $\mathcal{M}$ is an optimal algorithm for b-matching, we get

$$\sum_{j \in [k+m]} \sum_{x \in \widehat{X}_j} D^z(x, c_j) \leq \sum_{j \in [k+m]} \sum_{x \in X_{N,j}^{\mathsf{opt}}} D^z(x, c_j) = \sum_{x \in X_N^{\mathsf{opt}}} D^z(x, C) \leq \frac{\varepsilon}{2} \cdot \mathsf{opt}(\mathcal{I}), \qquad (3)$$

where the last inequality follows from Claim 3. Now,

$$\sum_{x \in X_0^{\mathsf{opt}}} D^z(x, \mu(x)) = \sum_{x \in X_N^{\mathsf{opt}}} D^z(x, \mu(x)) = \sum_{j \in [k+m]} \sum_{x \in X_{N,j}^{\mathsf{opt}}} D^z(x, \mu(x))$$

$$\overset{(1)}{\leq} z \cdot \sum_{j \in [k+m]} \sum_{x \in X_{N,j}^{\mathsf{opt}}} \left(D^z(x, c_j) + D^z(c_j, \mu(x)) \right), \qquad (4)$$

where the first equality follows from the fact that μ is identity on X_F^{opt}. Since μ is a bijection from $X_{N,j}^{\mathsf{opt}}$ to $\widehat{X}_j$, the above can also be written as

$$z \cdot \sum_{j \in [k+m]} \sum_{x \in X_{N,j}^{\mathsf{opt}}} D^z(x, c_j) + z \cdot \sum_{j \in [k+m]} \sum_{x \in \widehat{X}_j} D^z(x, c_j) \leq z \cdot \varepsilon \, \mathsf{opt}(\mathcal{I}),$$

where the last inequality follows from Claim 3 and (3). This proves the desired result.   ◀

The mapping μ described above may have the following undesirable property: there could be a point $x \in X_0^{\mathsf{opt}} \cap \widehat{X}$ such that $\mu(x) \neq x$. This could happen if $x \in X_{N,j}^{\mathsf{opt}}$ and $x \in \widehat{X}_i$ where $i \neq j$. We now show that μ can be modified to another bijection $\widehat{\mu} : X_0^{\mathsf{opt}} \to \widehat{X}$ which is identity on $X_0^{\mathsf{opt}} \cap \widehat{X}$. Note that the mapping $\widehat{\mu}$ is only needed for the analysis of the algorithm.

▶ **Lemma 5.** *There is a bijection $\widehat{\mu} : X_0^{\mathsf{opt}} \to \widehat{X}$ such that $\widehat{\mu}(x) = x$ for all $x \in X_0^{\mathsf{opt}} \cap \widehat{X}$ and*

$$\sum_{x \in X_0^{\mathsf{opt}}} D^z(x, \widehat{\mu}(x)) \le m^{z-1} \varepsilon \cdot z \cdot \mathsf{opt}(\mathcal{I}).$$

Proof. We construct a directed graph $H = (V_1, E_1)$ where $V_1 = X_0^{\mathsf{opt}} \cup \widehat{X}$. For every $x \in X_0^{\mathsf{opt}}$, we add the directed arc $(x, \mu(x))$ to E_1. Observe that a self loop in H implies that $\mu(x) = x$. Every vertex in $X_0^{\mathsf{opt}} \setminus \widehat{X}$ has 0 in-degree and out-degree 1; whereas a vertex in $\widehat{X} \setminus X_0^{\mathsf{opt}}$ has in-degree 1 and 0 out-degree. Vertices in $\widehat{X} \cap X_0^{\mathsf{opt}}$ have exactly one incoming and outgoing arc (in case of a self-loop, it counts towards both the in-degree and the out-degree of the corresponding vertex).

The desired bijection $\widehat{\mu}$ is initialized to μ. Let $\mathsf{cost}(\widehat{\mu})$ denote $\sum_{x \in X_0^{\mathsf{opt}}} D^z(x, \widehat{\mu}(x))$; define $\mathsf{cost}(\mu)$ similarly. It is easy to check H is vertex disjoint union of directed cycles and paths. In case of a directed cycle C on more than 1 vertex, it must be the case that each of the vertices in C belong to $\widehat{X} \cap X_0^{\mathsf{opt}}$. In this case, we update $\widehat{\mu}$ be defining $\widehat{\mu}(x) = x$ for each $x \in C$. Clearly this can only decrease $\mathsf{cost}(\widehat{\mu})$. Let $P_1, \ldots, P_l$ be the set of directed paths in H. For each path P_j, we perform the following update: let P_j be a path from a_j to b_j. We know that $a_j \in X^{\mathsf{opt}} \setminus \widehat{X}$, $b_j \in \widehat{X} \setminus X_0^{\mathsf{opt}}$ and each internal vertex of P_j lies in $\widehat{X} \cap X_0^{\mathsf{opt}}$. We update $\widehat{\mu}$ as follows; $\widehat{\mu}(a_j) = b_j$ and $\widehat{\mu}(v) = v$ for each internal vertex v of P_j. The overall increase in $\mathsf{cost}(\widehat{\mu})$ is equal to

$$\sum_{j \in [l]} \left(D^z(a_j, b_j) - \sum_{i=1}^{n_j} D^z(v_j^i, v_j^{i-1}) \right), \tag{5}$$

where $a_j = v_j^0, v_j^1, \ldots, v_j^{n_j} = b_j$ denotes the sequence of vertices in P_j. If $z = 1$, triangle inequality shows that the above quantity is at most 0. In case $z = 2$,

$$D^2(a_j, b_j) \le n_j \left(\sum_{i=1}^{n_j} D^2(v_j^i, v_j^{i-1}) \right),$$

and so the quantity in (5) is at most $(n_j - 1) \sum_{i=1}^{n_j} D^2(v_j^i, v_j^{i-1})$.

It follows that $\mathsf{cost}(\widehat{\mu}) \le m^{z-1} \mathsf{cost}(\mu)$. The desired result now follows from Lemma 4. ◀

We run the algorithm $\mathcal{A}$ on the outlier-free constrained clustering instance $\mathcal{I}'' = (X \setminus \widehat{X}, F, k, \mathsf{check}, \mathsf{cost})$ (line 1.8 in Algorithm 1). Let $\mathsf{opt}(\mathcal{I}'')$ be the optimal cost of a solution for this instance. The following key lemma shows that $\mathsf{opt}(\mathcal{I}'')$ is close to $\mathsf{opt}(\mathcal{I})$.

▶ **Lemma 6.** $\mathsf{opt}(\mathcal{I}'') \le (1 + \varepsilon^{\frac{1}{z}}(4m + 1)^{z-1})\mathsf{opt}(\mathcal{I}).$

Proof. We shall use the solution $(X_0^{\mathsf{opt}}, \ldots, X_k^{\mathsf{opt}})$ to construct a feasible solution for $\mathcal{I}''$. For each $j \in [k]$, let Z_j denote $X_j^{\mathsf{opt}} \cap \widehat{X}$. Let $\widehat{\mu}^{-1}(Z_j)$ denote the pre-image under $\widehat{\mu}$ of Z_j. Since $Z_j \subseteq \widehat{X} \setminus X_0^{\mathsf{opt}}$, $\widehat{\mu}^{-1}(Z_j) \subseteq X_0^{\mathsf{opt}} \setminus \widehat{X}$. For each $j \in [k]$, define $X_j' := (X_j^{\mathsf{opt}} \setminus Z_j) \cup \widehat{\mu}^{-1}(Z_j)$.

▷ **Claim 4.** $\bigcup_{j=1}^{k} X_j' = X \setminus \widehat{X}$.

Proof. For any $j \in [k]$, we have already argued that $\widehat{\mu}^{-1}(Z_j) \subseteq X_0^{\mathsf{opt}} \setminus \widehat{X} \subseteq X \setminus \widehat{X}$. Clearly, $X_j^{\mathsf{opt}} \setminus Z_j \subseteq X \setminus \widehat{X}$. Therefore $X_j' \subseteq X \setminus \widehat{X}$. Therefore, $\cup_{j \in [k]} X_j' \subseteq X \setminus \widehat{X}$. Since $|X_j'| = |X_j^{\mathsf{opt}}|$,

$$\sum_{j \in [k]} |X_j'| = n - m = |X \setminus \widehat{X}|.$$

This proves the claim. ◁

The above claim implies that $(X_1', \ldots, X_k')$ is a partition of $X \setminus \widehat{X}$. Since $|X_j'| = |X_j^{\mathsf{opt}}|$ for all $j \in [k]$ and the function check only depends on the cardinality of the sets in the partition, $(X_1', \ldots, X_k')$ is a feasible partition (under check) of $X \setminus \widehat{X}$. In the optimal solution for $\mathcal{I}$, let $f_1^{\mathsf{opt}}, \ldots, f_k^{\mathsf{opt}}$ be the k centers corresponding to the clusters $X_1^{\mathsf{opt}}, \ldots, X_k^{\mathsf{opt}}$ respectively. Now,

$$\mathsf{opt}(\mathcal{I}'') \leq \mathsf{cost}(X_1', \ldots, X_k') \leq \sum_{j \in [k]} \sum_{x \in X_j'} D^z(x, f_j^{\mathsf{opt}}) \tag{6}$$

For each $j \in [k]$, we estimate the quantity $\sum_{x \in X_j'} D^z(x, f_j^{\mathsf{opt}})$. Using the definition of X_j' and triangle inequality, this quantity can be expressed as

$$\sum_{x \in X_j^{\mathsf{opt}} \setminus Z_j} D^z(x, f_j^{\mathsf{opt}}) + \sum_{x \in \widehat{\mu}^{-1}(Z_j)} D^z(x, f_j^{\mathsf{opt}})$$

$$\leq \sum_{x \in X_j^{\mathsf{opt}} \setminus Z_j} D^z(x, f_j^{\mathsf{opt}}) + \sum_{x \in \widehat{\mu}^{-1}(Z_j)} \left(D(x, \widehat{\mu}(x)) + D(\widehat{\mu}(x), f_j^{\mathsf{opt}}) \right)^z \tag{7}$$

When $z = 1$, the above is at most (replacing x by $\widehat{\mu}(x)$ in the second expression on RHS)

$$\sum_{x \in X_j^{\mathsf{opt}}} D(x, f_j^{\mathsf{opt}}) + \sum_{x \in Z_j} D(x, \widehat{\mu}(x)).$$

Using this bound in (6), we see that

$$\mathsf{opt}(\mathcal{I}'') \leq \mathsf{opt}(\mathcal{I}) + \sum_{x \in X_0^{\mathsf{opt}}} D(x, \widehat{\mu}(x)) \leq (1 + \varepsilon)\mathsf{opt}(\mathcal{I}),$$

where the last inequality follows from Lemma 5. This proves the desired result for $z = 1$. When $z = 2$, we use the fact that for any two reals a, b,

$$(a + b)^2 \leq (1 + \sqrt{\varepsilon})a^2 + b^2 \left(1 + \frac{1}{\sqrt{\varepsilon}} \right).$$

Using this fact, the expression in the RHS of (7) can be upper bounded by

$$(1 + \sqrt{\varepsilon}) \sum_{x \in X_j^{\mathsf{opt}}} D^2(x, f_j^{\mathsf{opt}}) + \left(1 + \frac{1}{\sqrt{\varepsilon}} \right) \sum_{x \in Z_j} D^2(x, \widehat{\mu}(x)).$$

Substituting this expression in (6) and using Lemma 5, we see that

$$\mathsf{opt}(\mathcal{I}'') \leq (1 + \sqrt{\varepsilon})\mathsf{opt}(\mathcal{I}) + 4m\sqrt{\varepsilon}\,\mathsf{opt}(\mathcal{I}).$$

This proves the desired result for $z = 2$. ◀

The approximation preserving properties of Theorem 1 follow from the above analysis. For the k-means problem, since the approximation term is $(1 + \sqrt{\varepsilon}(4m + 1))$, we can replace ε with $\varepsilon^2/(4m + 1)^2$ in the algorithm and analysis to obtain a $(1 + \varepsilon)$ factor. Let us quickly check the running time of the algorithm. The algorithm first runs $\mathcal{C}$ that takes $T_{\mathcal{C}}(n)$ time. This is followed by D^z-sampling $O(\frac{m^{z+1}\log m}{\varepsilon^z})$ points, which takes $O(n \cdot (k + \frac{m^{z+1}\log m}{\varepsilon^z}))$ time. The number of iterations of the for-loops is determined by the number of subsets of S, which is $\sum_{i=0}^{m} \binom{|S|}{i} = \left(\frac{m}{\varepsilon}\right)^{O(m)}$, and the number of possibilities for τ, which is at most $\binom{2m+k-1}{m} = (m + k)^{O(m)}$. This gives the number of iterations $q = f(k, m, \varepsilon) = \left(\frac{k+m}{\varepsilon}\right)^{O(m)}$. In every iteration, in addition to running $\mathcal{A}$, we solve a weighted b-matching problem on a bipartite graph $(L \cup R, E)$ where R has $(k + m)$ vertices (corresponding to the $k + m$ centers in the center set C) and L has at most $(k + m) \cdot m$ vertices (considering m closest clients for every center is sufficient which can be found using a pre-processing step). So, every iteration costs $T_{\mathcal{A}}(n) + O((k + m)^3 m^2)$ time. This gives the running time expression in Theorem 1.

Extension to labelled version

In this section, we extend Algorithm 1 to the setting where points in X have labels from a finite set L and the check() function can also depend on the number of points with a certain label in a cluster. The overall structure of Algorithm 1 remains unchanged; we just indicate the changes needed in this algorithm.

Given a non-negative integer p, a label partition of p is defined as a tuple $\psi = (q_1, \ldots, q_{|L|})$ such that $\sum_i q_i = p$. The intuition is that given a set S of size p, q_1 points get the first label in L, q_2 points in S get the second label in L, and so on. Now, given a subset Y, define a valid tuple τ w.r.t. Y as a tuple $((t_1, \psi_1), \ldots, (t_{k+m}, \psi_{k+m}))$, where (i) $\sum_j t_j + |Y| = m$, and (ii) ψ_j is a label partition of t_j. As in line 1.5 in Algorithm 1, we iterate over all such valid tuples. The definition of a solution to the b-matching instance $\mathcal{I}^{(Y,\tau)}$ changes as follows. Let $\psi_j = (n_j^1, \ldots, n_j^\ell)$, where $\ell = |L|$. Then a solution to $\mathcal{I}^{(Y,\tau)}$ needs to satisfy the condition that for each point $c_j \in C$ and each label $l \in L$, exactly n_j^l points in X are matched to c_j. Note that this also implies that exactly t_j points are matched to c_j. This matching problem can be easily reduced to weighted bipartite matching by making t_j copies of each point c_j, and for each label l, adding edges between n_j^l distinct copies of c_j to vertices of label l only. The rest of the details of Algorithm 1 remain unchanged. Note that the running time of the algorithm changes because we now have to iterate over all partitions of each of the numbers t_j.

The analysis of the algorithm proceeds in an analogous manner as that of Algorithm 1. We just need to consider the iteration of the algorithm, where we correctly guess the size of each of the sets $X_{N,j}^{\mathsf{opt}}$ and the number of points of each label in this set.

References

1   Gagan Aggarwal, Rina Panigrahy, Tomás Feder, Dilys Thomas, Krishnaram Kenthapadi, Samir Khuller, and An Zhu. Achieving anonymity via clustering. *ACM Trans. Algorithms*, 6(3), July 2010. `doi:10.1145/1798596.1798602`.

2   Akanksha Agrawal, Tanmay Inamdar, Saket Saurabh, and Jie Xue. Clustering what matters: Optimal approximation for clustering with outliers, 2023. `arXiv:2212.00696`.

3   S. Ahmadian, A. Norouzi-Fard, O. Svensson, and J. Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 61–72, October 2017. `doi:10.1109/FOCS.2017.15`.

4   Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004. `doi:10.1137/S0097539702416402`.

5   Sayan Bandyapadhyay, Fedor V. Fomin, and Kirill Simonov. On Coresets for Fair Clustering in Metric and Euclidean Spaces and Their Applications. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:15, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.ICALP.2021.23`.

6   Suman Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. Fair algorithms for clustering. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL: `https://proceedings.neurips.cc/paper_files/paper/2019/file/fc192b0c0d270dbf41870a63a8c76c2f-Paper.pdf`.

7   Ioana O. Bercea, Martin Groß, Samir Khuller, Aounon Kumar, Clemens Rösner, Daniel R. Schmidt, and Melanie Schmidt. On the Cost of Essentially Fair Clusterings. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, volume 145 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:22, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.APPROX-RANDOM.2019.18`.

8   Anup Bhattacharya, Dishant Goyal, Ragesh Jaiswal, and Amit Kumar. On Sampling Based Algorithms for k-Means. In Nitin Saxena and Sunil Simon, editors, *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020)*, volume 182 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.FSTTCS.2020.13`.

9   Anup Bhattacharya, Ragesh Jaiswal, and Amit Kumar. Faster algorithms for the constrained k-means problem. *Theor. Comp. Sys.*, 62(1):93–115, January 2018. `doi:10.1007/s00224-017-9820-7`.

10   V. Braverman, V. Cohen-Addad, H. Jiang, R. Krauthgamer, C. Schwiegelshohn, M. Toftrup, and X. Wu. The power of uniform sampling for coresets. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 462–473, Los Alamitos, CA, USA, November 2022. IEEE Computer Society. `doi:10.1109/FOCS54457.2022.00051`.

11   Diptarka Chakraborty, Debarati Das, and Robert Krauthgamer. Clustering permutations: New techniques with streaming applications. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 31:1–31:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. `doi:10.4230/LIPIcs.ITCS.2023.31`.

12   Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002. `doi:10.1006/jcss.2002.1882`.

13   Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009. `doi:10.1137/070699007`.

14   Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li. Tight FPT Approximations for k-Median and k-Means. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.ICALP.2019.42`.

15   Vincent Cohen-Addad and Jason Li. On the Fixed-Parameter Tractability of Capacitated Clustering. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 41:1–41:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.ICALP.2019.41`.

16   Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 169–182, New York, NY, USA, 2021. Association for Computing Machinery. `doi:10.1145/3406325.3451022`.

17   Rajni Dabas, Neelima Gupta, and Tanmay Inamdar. Fpt approximations for capacitated/fair clustering with outliers, 2023. `arXiv:2305.01471`.

18   Hu Ding. Faster balanced clusterings in high dimension. *Theoretical Computer Science*, 842:28–40, 2020. `doi:10.1016/j.tcs.2020.07.022`.

19 Dan Feldman, Morteza Monemizadeh, and Christian Sohler. A PTAS for k-means clustering based on weak coresets. In *Proceedings of the twenty-third annual symposium on Computational geometry*, SCG '07, pages 11–18, New York, NY, USA, 2007. ACM. `doi:10.1145/1247069.1247072`.

20 Dishant Goyal, Ragesh Jaiswal, and Amit Kumar. FPT Approximation for Constrained Metric k-Median/Means. In Yixin Cao and Marcin Pilipczuk, editors, *15th International Symposium on Parameterized and Exact Computation (IPEC 2020)*, volume 180 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:19, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.IPEC.2020.14`.

21 Mohammadtaghi Hajiaghayi, Wei Hu, Jian Li, Shi Li, and Barna Saha. A constant factor approximation algorithm for fault-tolerant k-median. *ACM Trans. Algorithms*, 12(3), April 2016. `doi:10.1145/2854153`.

22 Lingxiao Huang, Shaofeng H. C. Jiang, Jianing Lou, and Xuan Wu. Near-optimal coresets for robust clustering, 2022. `arXiv:2210.10394`.

23 Tanmay Inamdar and Kasturi Varadarajan. Fault tolerant clustering with outliers. In Evripidis Bampis and Nicole Megow, editors, *Approximation and Online Algorithms*, pages 188–201, Cham, 2020. Springer International Publishing.

24 Ravishankar Krishnaswamy, Amit Kumar, Viswanath Nagarajan, Yogish Sabharwal, and Barna Saha. The matroid median problem. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 1117–1130, USA, 2011. Society for Industrial and Applied Mathematics.

25 Ravishankar Krishnaswamy, Shi Li, and Sai Sandeep. Constant approximation for k-median and k-means with outliers via iterative rounding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 646–659, New York, NY, USA, 2018. Association for Computing Machinery. `doi:10.1145/3188745.3188882`.

26 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM*, 57(2):5:1–5:32, February 2010. `doi:10.1145/1667053.1667054`.

27 Clemens Rösner and Melanie Schmidt. Privacy Preserving Clustering with Constraints. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 96:1–96:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.ICALP.2018.96`.

A   Tables

Table 1 The table defines various outlier-free versions of the constrained k-median problem. The k-means versions are defined similarly using D^2 instead of D. We include a few references. The problems are categorized based on the *type* of constraints. There are three main types of constraints (i) *size* (constraints on the cluster size), (ii) *center* (constraints on the points a center can service), and (iii) *label* (constraints on the label of points in clusters). A constrained problem can have a combination of these constraint types.

Problem	Description
Unconstrained k-median (*Constraint type: unconstrained*)	*Input*: (F, X, k) *Output*: $(X_1, ..., X_k, f_1, ..., f_k)$ *Constraints*: None, i.e., $\mathsf{check}(X_1, ..., X_k, f_1, ..., f_k)$ always equals 1. *Objective*: Minimise $\sum_i \sum_{x \in X_i} D(x, f_i)$. (This includes various versions corresponding to specific metrics such as Ulam metric on permutations, metric spaces with constant doubling dimension etc.)
Fault-tolerant k-median (*Constraint type: unconstrained but labelled*) [21, 23]	*Input*: (F, X, k) and a number $h(x) \le k$ for every facility $x \in X$ *Output*: $(f_1, ..., f_k)$ *Constraints*: None. *Objective*: Minimise $\sum_{x \in X} \sum_{j=1}^{h(x)} D(x, f_{\pi_x(j)})$, where $\pi_x(j)$ is the index of jth nearest center to x in $(f_1, ..., f_k)$ (*Label*: $h(x)$ may be regarded as the label of the client x. So, the number of distinct labels $\ell \le k$.)
Balanced k-median (*Constraint type: size*) [1, 18]	*Input*: (F, X, k) and integers $(r_1, ..., r_k), (l_1, ..., l_k)$, *Output*: $(X_1, ..., X_k, f_1, ..., f_k)$ *Constraints*: X_i should have at least r_i and at most l_i clients, i.e., $\mathsf{check}(X_1, ..., X_k, f_1, ..., f_k) = 1$ iff $\forall i, r_i \le |X_i| \le l_i$. *Objective*: Minimise $\sum_i \sum_{x \in X_i} D(x, f_i)$. (Versions corresponding to specific values of r_i's and l_i's are known by different names. The version corresponding to $l_1 = ... = l_k = |X|$ is called the r-gather problem and the version where $r_1 = ... = r_k = 0$ is called the l-capacity problem.)
Capacitated k-median (*Constraint type: center + size*) [15]	*Input*: (F, X, k) and with capacity $s(f)$ for every facility $f \in F$ *Output*: $(X_1, ..., X_k, f_1, ..., f_k)$ *Constraints*: The number of clients, X_i, assigned to f_i is at most $s(f_i)$, i.e., $\mathsf{check}(X_1, ..., X_k, f_1, ..., f_k) = 1$ iff $\forall i, |X_i| \le s(f_i)$. *Objective*: Minimise $\sum_i \sum_{x \in X_i} D(x, f_i)$.
Matroid k-median (*Constraint type: center*) [24, 14]	*Input*: (F, X, k) and a Matroid on F *Output*: $(X_1, ..., X_k, f_1, ..., f_k)$ *Constraints*: The number of clients, X_i, assigned to f_i is at most $s(f_i)$, i.e., $\mathsf{check}(X_1, ..., X_k, f_1, ..., f_k) = 1$ iff $\{f_1, ..., f_k\}$ is an independent set of the Matroid . *Objective*: Minimise $\sum_i \sum_{x \in X_i} D(x, f_i)$.
Strongly private k-median (*Constraint type: label + size*) [27]	*Input*: (F, X, k) and numbers $(l_1, ..., l_w)$. Each client has a label $\in \{1, ..., w\}$. *Output*: $(X_1, ..., X_k, f_1, ..., f_k)$ *Constraints*: Every X_i has at least l_j clients with label j, i.e., $\mathsf{check}(X_1, ..., X_k, f_1, ..., f_k) = 1$ iff $\forall i, j, |X_i \cap S_j| \ge l_j$, where S_j is the set of clients with label j . *Objective*: Minimise $\sum_i \sum_{x \in X_i} D(x, f_i)$. (*Labels*: The number of distinct labels $\ell = w$).
l-diversity k-median (*Constraint type: label + size*) [7]	*Input*: (F, X, k) and a number $l > 1$. Each client has one colour from $\in \{1, ..., w\}$ *Output*: $(X_1, ..., X_k, f_1, ..., f_k)$ *Constraints*: The fraction of clients with colour j in every X_i is at least $1/l$, i.e., $\mathsf{check}(X_1, ..., X_k, f_1, ..., f_k) = 1$ iff $\forall i, j, |X_i \cap S_j| \le |X_i|/l$, where S_j is the set of clients with colour j . *Objective*: Minimise $\sum_i \sum_{x \in X_i} D(x, f_i)$. (*Labels*: Each colour can be regarded as a label and hence the number of distinct labels $\ell = w$.)
Fair k-median (*Constraint type: label + size*) [7, 6]	*Input*: (F, X, k) and fairness values $(\alpha_1, ..., \alpha_w), (\beta_1, ..., \beta_w)$. Each client has colours from $\in \{1, ..., w\}$ *Output*: $(X_1, ..., X_k, f_1, ..., f_k)$ *Constraints*: The fraction of clients with colour j in every X_i is between α_j and β_j, i.e., $\mathsf{check}(X_1, ..., X_k, f_1, ..., f_k) = 1$ iff $\forall i, j, \alpha_j |X_i| \le |X_i \cap S_j| \le \beta |X_i|$, where S_j is the set of clients with colour j . *Objective*: Minimise $\sum_i \sum_{x \in X_i} D(x, f_i)$. (There are two versions: (i) each client has a unique label, and (ii) a client can have multiple labels.) (*Labels*: For the first version $\ell = w$ and for the second version $\ell = 2^w$.)

■ **Table 2** A × means that the techniques are not known to apply to the problem. The new results that do not follow from the previously known results are shaded ▬. The results that were not explicitly reported but follow from the techniques in the paper are shaded ▬ The techniques of [2] do not apply to the Ulam k-median problem since the outlier-free algorithm works on unweighted instances. Note that all the FPT $(3 + \varepsilon)$ and $(9 + \varepsilon)$ approximations for the outlier-free versions (leftmost column) in the last row follow from the outlier-free results in [20]. However, the approximation guarantees in the rightmost column depend on those in the leftmost. This means, unlike the rigid $(3 + \varepsilon)$ and $(9 + \varepsilon)$ approximation of [20] in the middle column, the approximation guarantee in the rightmost column will improve with every improvement in the leftmost.

Problem	Outlier-free	Outlier version		
		[20]	[2]	**This work**
Euclidean k-means (i.e., $F = \mathbb{R}^d, X \subset \mathbb{R}^d$)	$(1 + \varepsilon)$ [9]	×	$(1 + \varepsilon)$	$(1 + \varepsilon)$
k-median	$\left(1 + \frac{2}{e} + \varepsilon\right)$ [14]	$(3 + \varepsilon)$	$\left(1 + \frac{2}{e} + \varepsilon\right)$	$\left(1 + \frac{2}{e} + \varepsilon\right)$
k-means	$\left(1 + \frac{8}{e} + \varepsilon\right)$ [14]	$(9 + \varepsilon)$	$\left(1 + \frac{8}{e} + \varepsilon\right)$	$\left(1 + \frac{8}{e} + \varepsilon\right)$
k-median/means in metrics: (i) constant doubling dimension (ii) metrics induced by graphs of bounded treewidth (iii) metrics induced by graphs that exclude a fixed graph as a minor	$(1 + \varepsilon)$ [16]	$(3 + \varepsilon)$ k-median $(9 + \varepsilon)$ k-means	$(1 + \varepsilon)$	$(1 + \varepsilon)$
Matroid k-median	$(2 + \varepsilon)$ [14]	$(3 + \varepsilon)$	$(2 + \varepsilon)$	$(2 + \varepsilon)$
Colourful k-median	$\left(1 + \frac{2}{e} + \varepsilon\right)$ [14]	$(3 + \varepsilon)$	$\left(1 + \frac{2}{e} + \varepsilon\right)$	$\left(1 + \frac{2}{e} + \varepsilon\right)$
Ulam k-median (here $F = X$)	$(2 - \delta)$ [11]	$(2 + \varepsilon)$	×	$(2 - \delta)$
Euclidean Capacitated k-median/means	$(1 + \varepsilon)$ [15]	×	×	$(1 + \varepsilon)$
Capacitated k-median Capacitated k-means	$(3 + \varepsilon)$ $(9 + \varepsilon)$ [15]	× ×	× ×	$(3 + \varepsilon)$ $(9 + \varepsilon)$
Uniform/non-uniform r-gather k-median/means (uniform implies $r_1 = r_2 = ... = r_k$) Uniform/non-uniform l-capacity k-median/means (uniform implies $l_1 = l_2 = ... = l_k$) Uniform/non-uniform balanced k-median/means (uniform implies $r_1 = r_2 = ... = r_k$ and $l_1 = l_2 = ... = l_k$) Uniform/non-uniform fault tolerant k-median/means (uniform implies same $h(x)$ for every x) Strongly private k-median/means l-diversity k-median/means Fair k-median/means	$(3 + \varepsilon)$ (k-median) $(9 + \varepsilon)$ (k-means) [20]	$(3 + \varepsilon)$ (k-median) $(9 + \varepsilon)$ (k-means)	× ×	$(3 + \varepsilon)$ (k-median) $(9 + \varepsilon)$ (k-means)