Small Space Representations for Metric Min-Sum k-Clustering and their Applications [*†]

Artur Czumaj
Department of Computer Science
University of Warwick
Coventry CV7 4AL, United Kingdom
czumaj@dcs.warwick.ac.uk

Christian Sohler
Heinz Nixdorf Institute and
Department of Computer Science
University of Paderborn
D-33102 Paderborn, Germany
csohler@upb.de

Abstract

The *min-sum* k-*clustering* problem is to partition a metric space (P, d) into k clusters $C_1, \dots, C_k \subseteq P$ such that $\sum_{i=1}^{k} \sum_{p,q \in C_i} d(p,q)$ is minimized. We show the first efficient construction of a *coreset* for this problem. Our coreset construction is based on a new adaptive sampling algorithm. With our construction of coresets we obtain three main algorithmic results.

The first result is a sublinear time $(4 + \epsilon)$-approximation algorithm for the min-sum k-clustering problem in metric spaces. The running time of this algorithm is $\widetilde{\mathcal{O}}(n)$ for any constant k and ϵ, and it is $o(n^2)$ for all $k = o(\log n/\log\log n)$. Since the full description size of the input is $\Theta(n^2)$, this is *sublinear* in the input size. The fastest previously known $o(\log n)$-factor approximation algorithm for $k > 2$ achieved a running time of $\Omega(n^k)$, and no non-trivial $o(n^2)$-time algorithm was known before.

Our second result is the first *pass-efficient data streaming algorithm* for min-sum k-clustering in the distance oracle model, i.e., an algorithm that uses *poly*$(\log n, k)$ space and makes 2 passes over the input point set, which arrives in form of a data stream in arbitrary order. The algorithm can compute the distance between two points in constant time when they are both stored in the local memory. It computes an implicit representation of a clustering of (P, d) with cost at most a constant factor larger than that of an optimal partition. Using one further pass, we can assign each point to its corresponding cluster.

Our third result is a *sublinear-time* polylogarithmic-factor-approximation algorithm for the min-sum k-clustering problem for arbitrary values of k, possibly very large. This algorithm uses the coresets developed in this paper and combines them with an efficient algorithm for a weighted variant of the min-sum k-clustering problem.

To develop the coresets, we introduce the concept of α-*preserving metric embeddings*. Such an embedding satisfies properties that the distance between any pair of points does not decrease and the cost of an optimal solution for the considered problem on input (P, d') is within a constant factor of the optimal solution on input (P, d). In other words, the goal is find a metric embedding into a (structurally simpler) metric space that approximates the original metric up to a factor of α *with respect to a given problem*. We believe that this concept is an interesting generalization of coresets.

[*]Research supported in part by NSF ITR grant CCR-0313219, by EPSRC grant EP/D063191/1, by DFG grant Me 872/8-3, and by *Centre for Discrete Mathematics and its Applications (DIMAP)*, University of Warwick.

[†]A preliminary version of this paper appeared in *Proceedings of the 24th International Symposium on Theoretical Aspects of Computer Science (STACS'07)*, pages 536–548, Aachen, Germany, February 22–24, 2007. Lectures Notes in Computer Science vol. 4393, Springer-Verlag, Berlin, 2007.

1 Introduction

Clustering problems deal with the task of partitioning an input set of objects into subsets called *clusters*. In typical applications as they occur in bioinformatics, pattern recognition, data compression, data mining, and many other fields (see, e.g., [9, 33, 48]), clustering algorithms are either used to find certain patterns in large data sets or to find a lossy representation of the data that still maintains important features of the original data set. Ideally, objects in the same cluster are "similar" while objects in different clusters are not. To measure similarity between objects one typically defines a metric on the set of objects. The closer the objects are in this metric, the more similar they are, i.e., the metric is a measure of dissimilarity of objects. One of the most natural definitions of clustering is to minimize the intra cluster dissimilarity of objects. This problem is known as the *min-sum k-clustering* problem and has received considerably attention in the past [8, 14, 15, 23, 30, 44, 45]. Min-sum k-clustering is the *dual problem* to another well-known clustering problem, the *Max-k-cut problem* (see, e.g., [16, 19, 20, 21, 30]), where the goal is to maximize the inter cluster dissimilarity.

Unlike other classical clustering problems studied frequently in computer science, like k-median and k-means clustering, in min-sum k-clustering it is possible that clusters of different densities "overlap" (see, e.g., Figure 1 (b)). This makes the problem significantly different from k-median and k-means, and combinatorially considerably more challenging. In many scenarios the property that the clusters may "overlap" can lead to a better clustering, for example, it can be used to detect *outliers*. Consider an input instance containing ℓ well defined clusters and a few outliers (see Figure 1). If one uses min-sum k-clustering with $k = \ell + 1$, then an optimal solution will contain the ℓ original clusters and one separate cluster for the outliers. This will happen, if the number of outliers is at most a small fraction of the number of points in the smallest cluster. In this case, assigning an outlier to one of the ℓ initial clusters will be much more expensive than putting it in the outlier cluster that contains only a few points. This stands in sharp contrast to k-median and k-means clustering, where an optimal solution is always induced by a Voronoi partition with respect to the cluster centers, i.e., each point is assigned to the nearest cluster center (see Figure 1 (c)).

In typical applications of clustering, the data sets to be clustered tend to be very large. Therefore, the *scalability* to massive data sets is one of the most important requirements a good clustering algorithm should satisfy. In this context, the classical definition that a problem can be efficiently solved if it has a polynomial time algorithm does not always apply. Even though polynomial-time algorithms may be efficient for small and medium-sized inputs, they may become impractical for input sizes of several gigabytes. For example, when we consider approximation algorithms for clustering problems in metric spaces then they typically run in time $\Omega(n^2)$, where n is the number of input points. Clearly, such a running time is not feasible for massive data sets. Similarly, we do not have several gigabytes of main memory available and so our algorithm cannot maintain the entire input in main memory. Since access to secondary memory typically dominates the running time of an algorithm, we would like to do this access as fast as possible. This is typically achieved by a data streaming algorithm that passes only few times over the data. Our goal is to develop clustering algorithms that require *near linear (in n) running time* and/or *polylogarithmic space* and not more than a few passes over the input data.

1.1 Related work

The *min-sum k-clustering* problem was first formulated (for general graphs) by Sahni and Gonzales [44]. It is known to be $\mathcal{NP}$-hard and there is a 2-approximation algorithm by Guttman-Beck and Hassin [23] with running time $n^{\mathcal{O}(k)}$. Bartal et al. [8] presented an $\mathcal{O}(\epsilon^{-1} \log^{1+\epsilon} n)$-approximation algorithm running in time $n^{2+\mathcal{O}(1/\epsilon)}$ and Fernandez de la Vega et al. [15] gave an $(1 + \epsilon)$-approximation algorithm with the

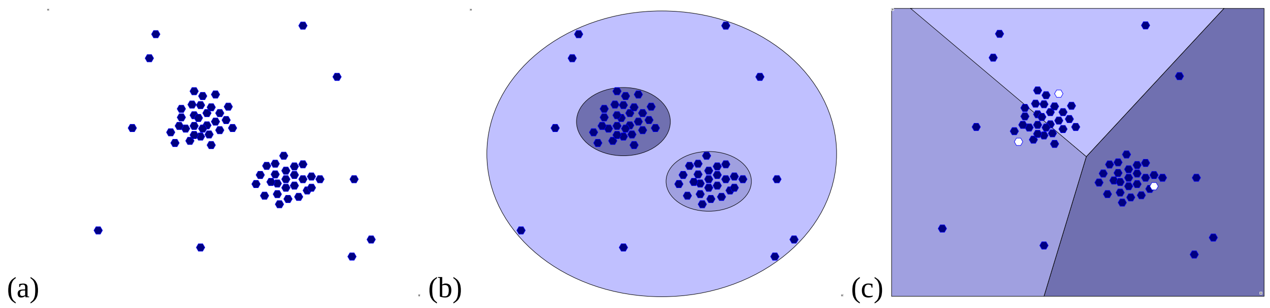

Figure 1: For a given set of points on the plane in Figure (a), Figure (b) presents a possible solution to min-sum 3-clustering. Observe that the cluster with the outliers *"overlaps"* the other two clusters. Figure (c) presents a solution to (discrete) 3-means; white points represent the three centers. Note that an optimal solution for k-median or k-means is always induced by a Voronoi diagram of the cluster centers.

running time of $\mathcal{O}(n^{3k} \cdot 2^{\mathcal{O}(\epsilon^{-k^2})})$. For point sets in $\mathbb{R}^d$, Schulman [45] introduced an algorithm for distance functions ℓ_2^2, ℓ_1 and ℓ_2 that computes a solution that is either within $(1+\epsilon)$ of the optimum or that disagrees with the optimum in at most an ϵ fraction of points. For the basic case of $k = 2$ (which is complement to the Max-Cut), Indyk [30] gave an $(1+\epsilon)$-approximation algorithm that runs in time $\mathcal{O}(2^{1/\epsilon^{\mathcal{O}(1)}} n (\log n)^{\mathcal{O}(1)})$, which is sublinear in the full input description size. In [14], the authors analyze the quality of a uniform random sampling approach for the min-sum k-clustering problem.

Sublinear time algorithms for other clustering problems. A number of sublinear time algorithms have been developed for other clustering problems. For radius (k-center) and diameter clustering in Euclidean spaces, property testing algorithms [2, 13] and tolerant testing algorithms [42] have been developed. For the k-median problem (in general metrics), the quality of random sampling has been investigated in [14] and [40]. Other sublinear time algorithms have been developed in [28, 39] and $\widetilde{\mathcal{O}}(nk)$ algorithms for the k-median problem in metric spaces have been presented in [37, 46]. Note that the running time of these algorithms is sublinear in the full description size of the metric space. In [4], a sublinear time algorithm to estimate the cost of the facility location problem has been developed.

Data streaming algorithms for other clustering problems. Several data streaming algorithms have been designed for clustering problems in Euclidean and general metric spaces.

For a data stream of points from the *Euclidean* space $\mathbb{R}^d$ one can compute a $(1+\epsilon)$-approximation for k-median and k-means clustering in space polynomial in $\log n, k$, and ϵ and exponential in d [25]. This result was improved to polynomial dependence in d in [12]. In [31] the study of dynamic geometric data streams was initiated. In this model the input points come from the discrete space $\{1, \dots, \Delta\}^d$ and the stream consists of insert and delete operations. It was shown in [31] that a $(1+\epsilon)$-approximation for the k-median problem can, in principle, be computed in small space. However, the running time to compute the approximation from the summary was prohibitively large. Using a different approach, it was shown in [19] how to quickly get a $(1+\epsilon)$-approximation for k-median, k-means and Max-Cut in the dynamic model. For the facility location problem a $\mathcal{O}(d \log^2 \Delta)$-approximation has been given in [31]; see also [18] for a more recent work on this subject.

For *general metric spaces*, there are constant factor approximation algorithms for the k-center problem [10] and the k-median problem [11, 22, 46]. Also variants of facility location problems [34, 38] has been considered. For more work on data streaming, we refer to the recent survey due to Muthukrishnan [41].

Coresets for other clustering problems. The concept of *coresets* (see, e.g., [3]) has been playing a critical role in the recent developments of efficient clustering algorithms. Informally, a coreset is a small weighted point set that approximates a larger unweighted point set with respect to a certain (clustering) problem. For some *geometric* problems, like k-center or k-median clustering, coresets with size independent of the number of points and the dimension of the underlying Euclidean space exist [5]. Other coreset constructions for low-dimensional Euclidean spaces have been developed and used to improve existing approximation algorithms and develop data streaming clustering algorithms [19, 25, 26]. It is even possible to compute a coreset for the k-median clustering problem, whose number of points is independent of the dimension [12]; the latter construction works for arbitrary metric spaces. Coresets have also been used to obtain fast algorithm in the context of projective clustering [27] and moving data sets [24].

To the best of our knowledge, previous approximation techniques for k-means and k-median like [5, 35, 36] as well as the combination of the coreset construction from [12] with the bicriteria approximation from [8] and the analysis of [14] cannot be used to obtain coresets for min-sum k-clustering and the related balanced k-median problem (even in the Euclidean case). For further details, see Section 1.3.

1.2 New contributions

In this paper we construct small space representations for approximately good solutions for the min-sum k-clustering problem and the related *balanced* k-*median* problem, which is known to be within a factor 2 of the solution for the min-sum k-clustering problem. We apply our constructions to design new efficient algorithms for these problems. In particular, we develop a new adaptive sampling scheme for balanced k-median. Based on this sampling scheme we obtain an α-preserving embedding and a coreset for the balanced k-median and min-sum k-clustering problem. Our constructions run in near linear time and can be implemented to use small space. Using the developed tools, we obtain three main algorithmic results.

First, we present a *sublinear time* constant factor approximation algorithm for the case that k is small, i.e., $k = o(\log n/\log\log n)$. For this choice of k, our algorithm runs in $\mathcal{O}(n \cdot k \cdot (k \log n/\epsilon)^{\mathcal{O}(k)})$ time and computes a $(4+\epsilon)$-approximation for the min-sum k-clustering problem. For $\omega(1) \le k \le \mathcal{O}(\log n/\log\log n)$, this is the first constant-factor polynomial-time approximation algorithm for this problem. Note that the running time of this algorithm is *sublinear* in the full description size of the metric space, which is $\Theta(n^2)$. Furthermore, we can speed-up the algorithm to run in time $\widetilde{\mathcal{O}}(n\,k) + (k \log n)^{\mathcal{O}(k)}$ and still achieve a constant (but a slightly worse one) factor approximation.

Our second result is a 2-*pass data streaming* algorithm that is a constant-factor approximation for the min-sum k-clustering problem, i.e., an algorithm that uses *poly*$(\log n, k)$ space and requires two passes over the point set P which arrives in an arbitrary order. The output of the algorithm is a succinct representation of the clustering. One more pass is needed to assign each point to the corresponding cluster. This is the first data streaming algorithm for this problem.

Our third result is a $(\log n)^{\mathcal{O}(1)}$-approximation algorithm for the min-sum k-clustering problem that runs in $\widetilde{\mathcal{O}}(n \cdot k^{\mathcal{O}(1)})$ time; hence, its running time is sublinear even for large values of k. This result almost matches the approximation guarantee of the best polynomial-time algorithm for this problem for large k due to Bartal et al. [8], and at the same time, it significantly improves the runtime.

New concepts and techniques. To obtain our first result we develop a new adaptive random sampling algorithm PARTITIONINGSCHEME. This algorithm computes a set S of *poly*$(k, \log n, 1/\epsilon)$ points that contains k centers, which (together with the right partition of points) are a $(2+\epsilon)$-approximation to the balanced k-median problem. Then we use a variation of exhaustive search together with a modified algorithm for the minimum cost assignment problem to find the best centers in S. The idea to compute a small set of points that contain good centers has been previously used in the context of k-median clustering [46]. However, both, our algorithm and its analysis for the *balanced* k-median are substantially different from previous work.

To obtain our second and the third algorithmic result, we introduce the concept of α-preserving embedding, which we believe is interesting beyond the applications in this paper. To define the concept of an *α-preserving embedding*, let Π be a minimization problem defined on finite metric spaces such that the cost of any solution does not decrease when the distances in the metric space are increased. Given two metric spaces (P, d) and (P, d'), we say that (P, d) has an α-preserving embedding into (P, d'), if *(i)* for every two points $p, q \in P$ it holds $d(p, q) \le d'(p, q)$, and *(ii)* the cost of an optimal solution for instance (P, d) is within factor α of the cost of an optimal solution for instance (P, d'). (For a more detailed discussion on α-preserving embeddings we refer to Section 5.) We use α-preserving embeddings to develop a *coreset* construction for the balanced k-median and min-sum k-clustering problem. Such a coreset can be seen as a small-space representation of the input metric space (P, d) that changes the cost of an optimal solution by at most a constant factor and does not improve the cost of other (bad) solutions significantly.

1.3 Comparison with related techniques

In this section we discuss briefly the most relevant techniques and concepts related to our work. We also discuss why we believe that these techniques are not applicable to min-sum k-clustering.

Clustering and coresets via sampling and pruning. One very successful approach to compute coresets and to solve clustering problems with a small number of clusters is sampling and pruning. Assume we want to cluster a point set P of n points. Then the idea behind the sampling and pruning approach is to exploit the fact that the largest of k clusters must have at least n/k points. Hence, a random sample of size $\Theta(s\,k)$ will typically contain $\Omega(s)$ points from the largest cluster. By exhaustive search we can find these points and hopefully get from them a good approximation of the cluster center. Once we have found the cluster center, we determine a set of points that certainly belongs to the largest cluster. To determine these points we typically make use of special properties of the clustering problem at hand. In particular, in the case of k-median and k-means clustering, we can exploit the fact that the clusters are induced by the Voronoi diagram of its centers. This approach has been used to obtain coresets and fast approximation algorithms for k-median and k-means clustering problems [5, 35, 36]. The difficulty in extending this approach to the min-sum k-clustering problem or related balanced k-median problem lies in the pruning step. Unlike in k-median or k-means clustering, we cannot easily determine a set of points that necessarily belongs to the largest cluster.

Coresets via space decomposition. Another successful approach to obtain coresets is via space decomposition. Here the idea is to compute a clever subdivision of the input space into few regions. Then each region of this subdivision is represented by only one or a small number of points. In [25], the space partition is given by k exponentially growing grids which are centered at a constant factor approximation to the k-median/k-means problem (i.e., the grids are finest near the centers and the size of a grid cell is roughly proportional to its distance from the center). Each non-empty grid cell will contain a single point whose weight equals the number of points in that cell. The quality can be analyzed by relating the overall move-

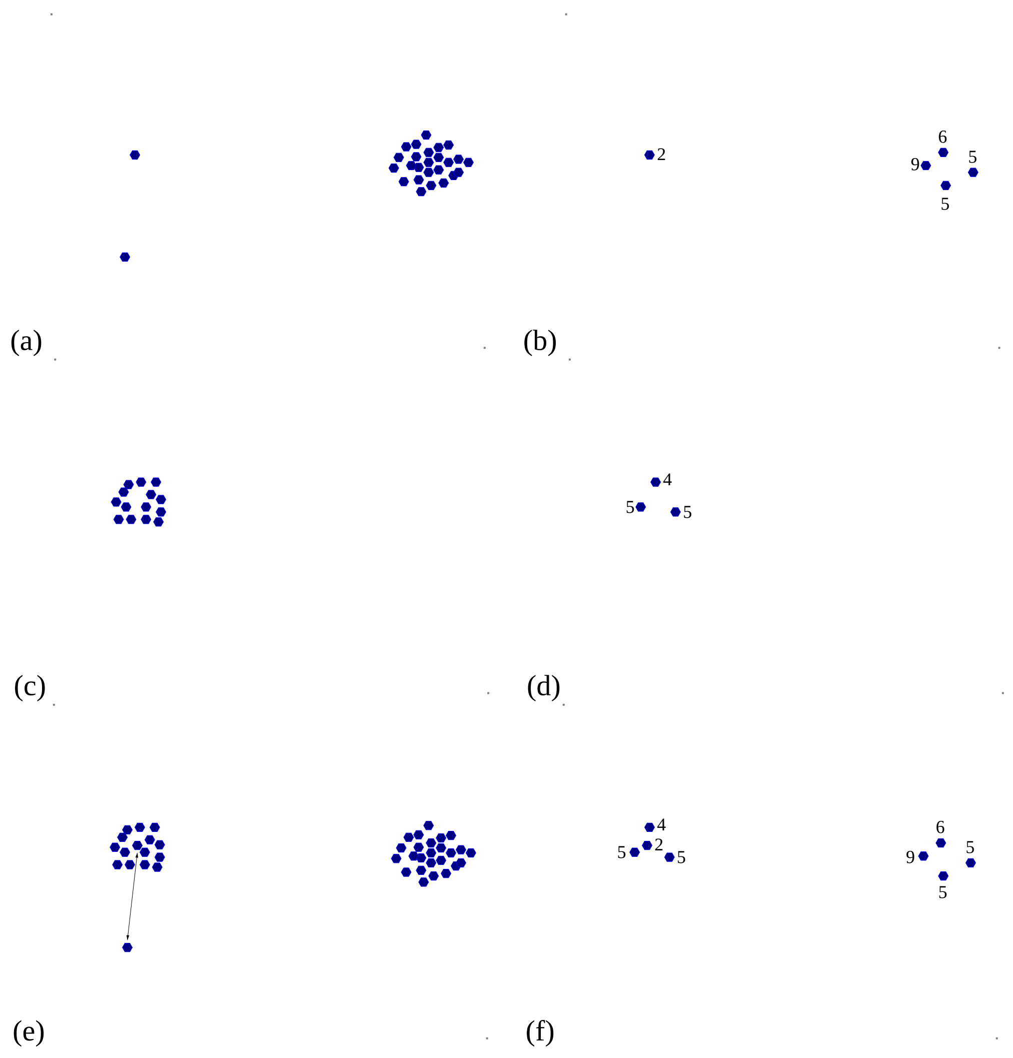

Figure 2: (a) A point set A and (b) its coreset. (c) A point set B and (d) its coreset. (e) The union of $A \cup B$. (f) The union of the coresets of A and of B is not the coreset of $A \cup B$.

ment of points to the distance to the nearest centers. This way an ϵ-coreset is obtained. In [19], a somewhat similar decomposition is obtained that does not require to first compute a constant factor approximation. Here the idea is to do a quadtree approach. We start with a coarse grid. Then we consider each non-empty cell and subdivide it, only if the overall movement of its points to the center of the cells exceeds a small fraction of the optimal cost. To obtain coresets in high dimensional spaces it has been proposed [12] to subdivide the input space in rings of exponential increasing width centered at the centers of a constant factor approximation (that possibly uses more centers than an optimal algorithm). From each ring a small number of points is sampled. It is known that an optimal solution to the k-median problem [40, 14] on a small sample of points from a space of diameter D differs w.h.p. at most by $\pm\epsilon\, D\, n$ from the optimal solution for the original point set. Since all points in a ring of diameter D have distance at least $D/2$ to the center of the constant factor approximation, one can charge this error against the cost of an optimal solution and obtain an ϵ-coreset.

The difficulty with applying space decomposition to the min-sum k-clustering can be easiest seen by considering the related balanced k-median problem. This problem differs from k-median only because the distance of each point to its cluster center is weighted by also the number of points in the cluster. Therefore, the cost of moving a point depends not only on the moved distance but the number of points in its cluster. This in turn is something that depends on the current clustering and so we do not know "how much to charge for moving a point." Another difficulty stems from the fact that the union of two coresets is a coreset (for an example, see Figure 2). This does not allow us to apply grid based approaches like [19, 25]. Another tempting approach is to use the approach from [12] with the bicriteria approximation from [8] as a constant factor approximation and the analysis of uniform sampling for min-sum k-clustering, as it is done in [14]. However, we do not know how to analyze this approach. The difficulty lies in the fact that unlike in the case of k-median clustering, the error in each ring is not a local property. Again, the error we make by moving a point by a certain distance is multiplied by the number of points in its cluster, which depends on the entire point set and not only on the points within the ring. The techniques we introduce in this paper are less sensitive to these problems than previous methods.

2 Preliminaries

Let (P, d) be a finite metric space and let $n = |P|$. For $S \subseteq P$ and $p \in P$ we will define $d(p, S) = \min_{q \in S} d(p, q)$. The *min-sum k-clustering* problem is to partition P into k sets (called *clusters*) $C_1, \ldots, C_k$ such that $\sum_{i=1}^{k} \sum_{p,q \in C_i} d(p, q)$ is minimized. It is known that this problem is within a factor two approximation (see, e.g., [8, Claim 1]) of the *balanced k-median* problem, which is to find a set $\mathcal{C} = \{c_1, \ldots, c_k\} \subseteq P$ of k points (centers) and a partition of P into sets $C_1, \ldots, C_k$ that minimizes $\sum_{i=1}^{k} |C_i| \cdot \sum_{p \in C_i} d(p, c_i)$.

Since min-sum k-clustering and balanced k-median approximate each other to within a factor of 2, in our analysis, we will focus on the latter problem and only state some of the final results for the former one.

For any set $\mathcal{C} = \{c_1, \ldots, c_k\}$ of k points (centers) in P, we define

$$\mathit{cost}_k(P, \mathcal{C}) = \min_{\substack{\text{partition of } P \text{ into} \\ C_1 \cup C_2 \cup \cdots \cup C_k}} \sum_{i=1}^{k} |C_i| \cdot \sum_{p \in C_i} d(p, c_i) \ .$$

We will abbreviate the cost of cluster C_i with associated center c_i as $\mathit{cost}(C_i) = \mathit{cost}_1(C_i, \{c_i\})$.

A *balanced k-median* of P is a set $\mathcal{C}^* = \{c_1^*, \ldots, c_k^*\}$ of k points (centers) in P that minimizes the value of $\mathit{cost}_k(P, \mathcal{C}^*)$. We will use Opt_k to denote the cost of a *balanced k-median* for P.

The next two notions, kernels and points close to kernels, play a key role in our analysis.

Definition 1 (Kernels and points close to kernel) *Let $\mathcal{C}^* = \{c_1^*, \ldots, c_k^*\}$ be a balanced k-median and let $C_1^*, \ldots, C_k^*$ be the corresponding optimal partition of P. We define the kernel $\mathit{Kern}(C_i^*)$ of cluster C_i^* as*

$$\mathit{Kern}(C_i^*) = \left\{ p \in C_i^* \ : \ d(p, c_i^*) \le (1 + \epsilon) \cdot \frac{\mathit{cost}(C_i^*)}{|C_i^*|^2} \right\} \ .$$

We say that a point p is close to $\mathit{Kern}(C_i^*)$, if

$$d\left(p, \mathit{Kern}(C_i^*)\right) \le \frac{\epsilon}{k} \cdot \frac{\mathit{Opt}_k}{|C_i^*|^2} \ .$$

The next lemma follows easily from the triangle inequality.

Lemma 2 *Let $\mathcal{C}^* = \{c_1^*, \dots, c_k^*\}$ be a balanced k-median and let $C_1^*, \dots, C_k^*$ be the corresponding optimal partition of P. Let $\mathcal{C} = \{c_1, \dots, c_k\}$ be any set of points such that each c_i is close to $Kern(C_i^*)$. Then*

$$\mathit{cost}_k(P, \mathcal{C}) \quad \le \quad (2 + 2\epsilon) \cdot \mathit{Opt}_k \ .$$

Proof: By the triangle inequality, for every point $p \in C_i^*$, we have

$$d(p, c_i) \quad \le \quad d(p, c_i^*) + d(c_i^*, c_i) \ \le \ d(p, c_i^*) + \frac{(1+\epsilon) \cdot \mathit{cost}(C_i^*)}{|C_i^*|^2} + \frac{\epsilon}{k} \cdot \frac{\mathit{Opt}_k}{|C_i^*|^2} \ ,$$

where the last inequality follows from the fact that c_i is close to $Kern(C_i^*)$. Hence, the cost of the partitioning $C_1^*, \dots, C_k^*$ with centers $c_1, \dots, c_k$ is at most

$$\sum_{i=1}^{k} |C_i^*| \cdot \sum_{p \in C_i^*} \left(d(p, c_i^*) + \frac{(1+\epsilon) \cdot \mathit{cost}(C_i^*)}{|C_i^*|^2} + \frac{\epsilon}{k} \cdot \frac{\mathit{Opt}_k}{|C_i^*|^2} \right) \quad \le \quad (2 + 2\epsilon) \cdot \mathit{Opt}_k \ .$$

□

Furthermore, simple counting arguments show the following.

Lemma 3 *For every i, $|Kern(C_i^*)| \ \ge \ \frac{\epsilon}{1+\epsilon} \cdot |C_i^*|$.*

Proof: Any point that is not in $Kern(C_i^*)$ contributes to $\mathit{cost}(C_i^*)$ at least $|C_i^*| \cdot (1+\epsilon) \cdot \frac{\mathit{cost}(C_i^*)}{|C_i^*|^2}$. Any point in $Kern(C_i^*)$ contributes to the cost of the cluster with at least 0. Hence, the number of points that are not in $Kern(C_i^*)$ is upper bounded by

$$\frac{\mathit{cost}(C_i^*)}{(1+\epsilon) \cdot \frac{\mathit{cost}(C_i^*)}{|C_i^*|}} \quad = \quad \frac{|C_i^*|}{1+\epsilon} \ = \ \left(1 - \frac{\epsilon}{1+\epsilon}\right) \cdot |C_i^*| \ .$$

□

The balanced k-median problem is defined similarly to another standard problem of k-*median*. The difference is that in the k-median problem, our goal is to find a set $\mathcal{C}^* = \{c_1^*, \dots, c_k^*\}$ of k centers in P and a partition of P into $C_1, \dots, C_k$ that minimize $\sum_{i=1}^{k} \sum_{p \in C_i} d(p, c_i)$. We will use the well-known and easy fact that if c_{med} is the cost of an optimal solution of k-median for P, then $c_{med} \le \mathit{Opt}_k \le |P| \cdot c_{med}$.

3 New sampling-based partitioning scheme

In this section we develop a partitioning scheme that with high probability finds a set S of size $\widetilde{\mathcal{O}}(k \log n / \epsilon^3)$ that contains a point close to $Kern(C_i^*)$ for every cluster C_i^* of an optimal solution. By Lemma 2, it follows that these points contain k centers that provide a good approximation to a balanced k-median. In the next section we will see how to compute a $(2+\epsilon)$-approximation for the balanced k-median from this set.

3.1 The algorithm RandomPartition

Our partitioning scheme uses an algorithm RandomPartition. This algorithm gets a "guess" $\widetilde{Opt}_k$ for the cost Opt_k of an optimal solution and a parameter ℓ, which can be viewed as a "guess" for the cluster size. Given these parameters, the algorithm selects a set $S \subseteq P$ using a simple adaptive sampling rule. As we will prove later in Lemma 8, if our "guess" $\widetilde{Opt}_k$ for Opt_k is good, then for every cluster C_i^* of size $(1-\epsilon)\cdot\ell \le |C_i^*| \le \ell$, set S contains, with high probability, a point p that is close to $Kern(C_i^*)$. RandomPartition is parameterized by a value s which is closely related to the sample size and will be specified later. (In the algorithm we assume that for any $p \in P$ we have $d(p,\emptyset) = +\infty$.)

```
RandomPartition $(P, s, \widetilde{Opt}_k, \ell)$
    for each $p \in P$ do
        if $d(p,S) > \frac{\epsilon}{k}\cdot\frac{\widetilde{Opt}_k}{\ell^2}$ then with probability $\min\left\{\frac{s}{\epsilon\cdot\ell}, 1\right\}$ put $p$ into $S$
    return $S$
```

The running time of algorithm RandomPartition is $\mathcal{O}(n\cdot|S|)$. Therefore, we want to find an upper bound on the size of the sample set S.

Lemma 4 *Let $0 < \rho < 1$ be arbitrary. Let $\widetilde{Opt}_k \ge Opt_k/2$. Then, for $s \ge \frac{15\cdot\epsilon^2\cdot\ln(1/\rho)}{k}$ we have $|S| \le \frac{6\cdot s\cdot k}{\epsilon^2} + k$ with probability at least $1-\rho$.*

Proof: Let $C_1^*, \dots, C_k^*$ be an optimal partition of P and let $c_1^*, \dots, c_k^*$ be the corresponding optimal centers. We denote by $t = \frac{\epsilon}{k}\cdot\frac{\widetilde{Opt}_k}{\ell^2}$ the threshold for the distance to the set S used in the **if**-statement of algorithm RandomPartition. We use the following simple fact, which follows directly from the triangle inequality.

Claim 5 *For each center c_i^*, the set S contains at most one point p with $d(p, c_i^*) \le t/2$.*

Proof: Assume there are two points p and q in S with $d(p,c_i^*), d(q,c_i^*) \le t/2$. Without loss of generality, let p be the point considered first in the **for each**-loop. Since $d(p,q) \le d(p,c_i^*) + d(q,c_i^*) \le t$, we get that $d(q,S) \le t$ and so q cannot be put in S. Contradiction. □

It follows immediately that there can be at most k points in S with distance at most $t/2$ to one of the cluster centers. We say that a point $p \in P$ is an *outlier*, if it has distance more than $t/2$ to every cluster center. We next analyze the number of outliers.

Claim 6 *The number of outliers is at most $4\cdot\ell\cdot k/\epsilon + k\cdot\ell$.*

Proof: Let us call a cluster *small*, if it has at most ℓ points. Otherwise, we call a cluster *large*. The overall number of points in small clusters is at most $k\cdot\ell$ and so the number of outliers in small clusters is at most $k\cdot\ell$. Any outlier in a large cluster contributes with at least $\ell\cdot t/2$ to the cost of an optimal solution. Therefore, there can be at most $Opt_k/(t\cdot\ell/2) \le 2\cdot\widetilde{Opt}_k/(t\cdot\ell/2) = 4\cdot\ell\cdot k/\epsilon$ outliers in large clusters. The claim follows by summing up the number of outliers in small and large clusters. □

Algorithm RandomPartition picks every outlier with probability at most $\min\left\{\frac{s}{\epsilon\cdot\ell}, 1\right\}$. If $\frac{s}{\epsilon\cdot\ell} > 1$, then $s > \epsilon\cdot\ell$. Since by Claim 5, the cardinality of S is bounded by the number of outliers plus k, we get

$$|S| \le 4\cdot\ell\cdot k/\epsilon + k\cdot\ell + k \le 6\cdot\ell\cdot k/\epsilon + k = \frac{6\cdot\epsilon\cdot\ell\cdot k}{\epsilon^2} + k \le \frac{6\cdot s\cdot k}{\epsilon^2} + k\ .$$

Hence, we only have to consider the case when $\frac{s}{\epsilon\cdot\ell} \le 1$. Let *Out* denote the set of outliers. For $1 \le Y_i \le |Out|$, let Y_i denote an independent 0-1-random variable with $\mathbf{Pr}[Y_i = 1] = \frac{s}{\epsilon\cdot\ell}$. We get

$$\mathbf{Pr}\Big[|S \cap Out| > 6 \cdot \tfrac{s\cdot k}{\epsilon^2}\Big] \;\le\; \mathbf{Pr}\Big[\sum_{i=1}^{|Out|} Y_i > 6 \cdot \tfrac{s\cdot k}{\epsilon^2}\Big] \; .$$

Clearly, the latter probability is maximized by maximizing $|Out|$. Let $M = 5 \cdot \ell \cdot k/\epsilon$. Since by Claim 6, $|Out| \le 4 \cdot \ell \cdot k/\epsilon + k \cdot \ell \le M$, we have $\mathbf{Pr}[\sum_{i=1}^{|Out|} Y_i > 6 \cdot \frac{s\cdot k}{\epsilon^2}] \le \mathbf{Pr}[\sum_{i=1}^{M} Y_i > 6 \cdot \frac{s\cdot k}{\epsilon^2}]$. Let us study the latter probability. We have, $\mathbf{E}[\sum_{i=1}^{M} Y_i] = \frac{s}{\epsilon\cdot\ell} \cdot M = \frac{5\cdot s\cdot k}{\epsilon^2}$. Now we can apply Chernoff bounds. We get

$$\mathbf{Pr}\Big[\sum_{i=1}^{M} Y_i > 6\tfrac{sk}{\epsilon^2}\Big] \;=\; \mathbf{Pr}\Big[\sum_{i=1}^{M} Y_i > (1+1/5)\mathbf{E}[\sum_{i=1}^{M} Y_i]\Big] \;\le\; e^{-(1/5)^2\mathbf{E}[\sum_{i=1}^{M} Y_i]/3} \;\le\; \rho$$

for $s \ge \frac{15\cdot\epsilon^2\cdot\ln(1/\rho)}{k}$. It follows that

$$\mathbf{Pr}\Big[|S| \le 6 \cdot \tfrac{s\cdot k}{\epsilon^2} + k\Big] \;\ge\; \mathbf{Pr}\Big[|S \cap Out| \le 6 \cdot \tfrac{s\cdot k}{\epsilon^2}\Big] \;\ge\; 1 - \rho \; .$$

□

The following immediate corollary of Lemma 4 will be useful to understand our line of proceeding.

Corollary 7 *Let $0 < \rho < 1$ and let $s \ge \frac{15\cdot\epsilon^2\cdot\ln(1/\rho)}{k}$. Suppose that* RandomPartition $(P, s, \widetilde{Opt}_k, \ell)$ *returns set S with $|S| > \frac{6\cdot s\cdot k}{\epsilon^2} + k$. Then, $\widetilde{Opt}_k < Opt_k/2$ with probability at least $1 - \rho$.* □

Our next key lemma shows that our sampling algorithm ensures that with a high probability, every optimal cluster has at least one point that is close to its kernel and that is in the sample set.

Lemma 8 *Let $\widetilde{Opt}_k \le Opt_k$, $\epsilon \le 1/2$ and $(1-\epsilon)\cdot\ell \le |C_i^*| \le \ell$. Then,*

$$\mathbf{Pr}[\exists p \in S : p \textit{ is close to } Kern(C_i^*)] \;\ge\; 1 - \rho \;\; \textit{for } s \ge 4 \cdot \ln(1/\rho) \; .$$

Proof: Let p be an arbitrary point in P that is not close to $Kern(C_i^*)$. By definition, we have in this case

$$d(p, Kern(C_i^*)) \;>\; \frac{\epsilon}{k} \cdot \frac{Opt_k}{|C_i^*|^2} \;\ge\; \frac{\epsilon}{k} \cdot \frac{\widetilde{Opt}_k}{\ell^2} \; .$$

Now, if no point in S is close to $Kern(C_i^*)$, then for every point p in $Kern(C_i^*)$ we have $d(p, S) > \frac{\epsilon}{k} \cdot \frac{\widetilde{Opt}_k}{\ell^2}$ and hence, every point from $Kern(C_i^*)$ was considered for being taken into S. Therefore, the probability that S has no point that is close to $Kern(C_i^*)$ is upper bounded by $\mathbf{Pr}[\sum_{j=1}^{|Kern(C_i^*)|} Y_i = 0]$, where the Y_i are 0–1-random variables with $\mathbf{Pr}[Y_i = 1] = \frac{s}{\epsilon\cdot\ell}$. This yields:

$$\mathbf{Pr}\Big[\sum_{j=1}^{|Kern(C_i^*)|} Y_i = 0\Big] \;=\; \Big(1 - \frac{s}{\epsilon\cdot\ell}\Big)^{|Kern(C_i^*)|} \;\le\; \Big(1 - \frac{s}{\epsilon\cdot\ell}\Big)^{\epsilon\cdot\ell/4} \;\le\; e^{-s/4} \; ,$$

where the second last inequality follows from Lemma 3, $|C_i^*| \ge (1-\epsilon)\,\ell$ and $\epsilon \le 1/2$. Hence, the lemma follows with $s = 4 \cdot \ln(1/\rho)$. □

3.2 The partitioning scheme

Now we are ready to present our partitioning scheme. First, we use the fact that the cost c_{med} of an α-approximation for k-median provides us with a lower bound of c_{med}/α and an upper bound of $c_{med} \cdot n$ for balanced k-median. We apply an $\widetilde{\mathcal{O}}(n\,k)$-time constant-factor approximation algorithm for k-median to compute an approximate solution to this problem (see, e.g., [22, 28, 46]). Next, we are trying to guess the right value $\widetilde{Opt}_k$ for Opt_k. We use $\widetilde{Opt}_k \approx c_{med} \cdot n$ as a first guess for the cost of an optimal solution. Then, we run RANDOMPARTITION for $\ell = 1, \lceil (1+\epsilon) \rceil, \lceil (1+\epsilon)^2 \rceil, \dots, n$ and build the union of the returned sets. Next, we halve our guess Opt_k for the cost of the optimal solution and proceed in a similar way until, eventually, RANDOMPARTITION returns a set which is larger than the bound from Lemma 4. If s is sufficiently large then we know at this point that with high probability our current guess $\widetilde{Opt}_k$ is smaller than $Opt_k/2$. Therefore, in the previous iteration (for the previous guess $\widetilde{Opt}_k$ of Opt_k), we had $\widetilde{Opt}_k \le Opt_k$, in which case we can use properties of the set S proven in Lemma 8.

We give our partitioning scheme in detail below.

```
PARTITIONINGSCHEME($P, s$)
    Compute cost $c_{med}$ of an $\alpha$-approximation for $k$-median in $\mathcal{O}(n \cdot k)$ time
    for $i = \lceil \log(\alpha \cdot n) \rceil$ downto $0$ do
        $S_i \leftarrow \emptyset$
        for $j = 0$ to $\lceil \log_{1+\epsilon} n \rceil$ do
            $X = $ RANDOMPARTITION($P, s, 2^i \cdot c_{med}/\alpha, \lceil (1+\epsilon)^j \rceil$)
            if $|X| > 6 \cdot \frac{s \cdot k}{\epsilon^2} + k$ then return $S_{i+1}$
            $S_i \leftarrow S_i \cup X$
    return $S_0$
```

We summarize properties of the partitioning scheme presented in this section in the following theorem.

Theorem 9 *Let* $N = (2 + \log(\alpha \cdot n)) \cdot (2 + \log_{1+\epsilon} n)$. *Run Algorithm* PARTITIONINGSCHEME *with parameter* $s \ge 4\ln(2Nk/\delta) = \mathcal{O}(\log(k \cdot \log n/(\delta \cdot \epsilon))$. *Then the algorithm runs in time* $\mathcal{O}(\frac{n \cdot \log^2 n \cdot k \cdot s}{\epsilon^3})$ *and finds a set* S *of* $\mathcal{O}(\frac{s \cdot k \cdot \log n}{\epsilon^3})$ *points in* P *such that with probability at least* $1 - \delta$, *there is a set* $\mathcal{C}$ *of* k *points in* S *for which cost*$(P, \mathcal{C}) \le (2 + \epsilon) \cdot Opt_k$.

Proof: We first observe that the cost c_{med} of an α-approximation of the k-median problem provides an upper bound of $c_{med} \cdot n$ and a lower bound of c_{med} on the cost of the balanced k-median.

Clearly, N is an upper bound on the number of calls to RANDOMPARTITION in Algorithm PARTITIONINGSCHEME. Let us first assume that in all calls to RANDOMPARTITION, if $\widetilde{Opt}_k \ge Opt_k/2$ then $|X| \le 6 \cdot \frac{sk}{\epsilon^2} + k$. By Lemma 4 and since $s \ge 4\ln(2Nk/\delta) \ge \frac{15\,\epsilon^2\,\ln(1/\rho)}{k}$, this holds with probability at least $1 - N \cdot \rho$. Furthermore, we assume that S contains a point p in the kernel of each cluster C_i^* of an optimal partition $C_1^*, \dots, C_k^*$. Lemma 8 ensures that this holds with probability at least $1 - k \cdot \rho$. Choosing $\rho = \delta/(2\,N\,k)$, all these events hold with probability at least $1 - \delta$ (what we assume for the rest of the proof).

Under these assumptions, we will eventually come to the situation that $1/2 \cdot Opt_k \le \widetilde{Opt}_k \le Opt_k$. Then, by Lemma 4 the sample will still be small, $|S_i| \le \mathcal{O}(s\,k\,\log_{1+\epsilon} n/\epsilon^2) = \mathcal{O}(s\,k\,\log n/\epsilon^3)$. Since we run RANDOMPARTITION with all powers of $(1+\epsilon)$ as values for ℓ, Lemma 8 ensures that for every cluster C_i^* sample set S contains a point close to its kernel. Therefore, Lemma 2 implies that there is a set $\mathcal{C}$ of k points in S for which *cost*$(P, \mathcal{C}) \le (2 + \epsilon) \cdot Opt_k$.

If we choose $s \geq 4\ln(1/\rho)$, then the prerequisites of Lemmas 4 and 8 are satisfied. The running time of algorithm PARTITIONINGSCHEME follows from the fact that we make $\mathcal{O}(\log^2 n/\epsilon)$ calls to RANDOMPARTITION. Next, we observe that each of these calls requires $\mathcal{O}(n \cdot |X|)$ running time. By Lemma 4 we get that $|X| = \mathcal{O}(\frac{k \cdot s}{\epsilon^2})$. Hence, the overall running time is as stated above. The overall sample size follows from the bound on $|X|$ and the fact that our sample set is built from $\mathcal{O}(\log n/\epsilon)$ calls to RANDOMPARTITION. $\square$

4 Polynomial-time constant factor approximation for small k

In this section we will show how the result from Theorem 9 can be used to obtain a polynomial-time constant factor approximation for balanced k-median for all values of $k = \mathcal{O}(\log n/\log\log n)$. The underlying idea of our algorithm is that if we have a good guess for a possible set of candidates for centers, then we can find an almost optimal partition for this set, which will yield a good approximation for balanced k-median.

4.1 Finding good partitions for fixed set of k centers

We consider the problem of finding a constant factor approximation for balanced k-median for a given *fixed* set of k centers. Let $\mathcal{C} = (c_1, \dots, c_k)$ be a sequence of k cluster centers and consider an integer k-tuple $\langle N_1, \dots, N_k \rangle$ with $\sum_{i=1}^{k} N_i = n$. Tokuyama and Nakano [47] showed that in time $\mathcal{O}(n\,k) + \widetilde{\mathcal{O}}(\sqrt{n}\,k^{2.5})$ one can find a partition of a set P into k subsets $C_1^*, \dots, C_k^*$ with $|C_i^*| = N_i$ for every i, that minimizes $cost_k(P, \mathcal{C}) = \sum_{i=1}^{k} |C_i^*| \sum_{p \in C_i^*} d(p, c_i)$. (We will use in our analysis an important property that the algorithm of Tokuyama and Nakano [47] does not require the input instance to be metric.) This algorithm can be used to solve the balanced k-median for a fixed set of k-centers by considering all possible values of N_i. However, the running time of this algorithm would be $\Omega(n^k)$, which is prohibitively expensive. In this section, we describe a $\mathcal{O}(1)$-approximation algorithm which has a significantly better running time.

Let us fix $\mathcal{C} = (c_1, \dots, c_k)$. Our goal is to partition P into k subsets $C_1, \dots, C_k$ for which the cost of the clustering $\sum_{i=1}^{k} |C_i| \sum_{p \in C_i} d(p, c_i)$ is close to $cost_k(P, \mathcal{C})$, i.e., to the optimal cost, and which can be found by applying the algorithm from [47] with the sizes of all clusters to be powers of $(1+\epsilon)$.

Our idea is simple. Instead of calling the algorithm of Tokuyama and Nakano [47] with the sizes of all clusters $N_1, \dots, N_k$ we do it with the sizes of clusters $\lambda_1, \dots, \lambda_k$ such that each λ_i is of the form $\lfloor (1+\epsilon)^m \rfloor$ for an integer m, and $N_i \leq \lambda_i < (1+\epsilon) \cdot N_i$. The "fake" points of each set (whose number is $\lambda_i - N_i$ in each cluster) are to be chosen from a set of additional points, from the set $\mathbb{O}_\ell$ for an appropriate value of ℓ. Here, for every integer ℓ we define set $\mathbb{O}_\ell$ to be the set of ℓ points such that for every $u \in \mathbb{O}_\ell$ and for every c_i, we have $d(u, c_i) = 0$. Observe that with such a definition of set $\mathbb{O}_\ell$ (and in particular, the requirement that $d(u, c_i) = 0$ for every $u \in \mathbb{O}_\ell$ and for every c_i) we will not obtain a metric instance of the problem, but as mentioned before, the algorithm of Tokuyama and Nakano [47] will still work in that case.

The following lemma shows that once we fix the set of centers and the sizes $N_1, \dots, N_k$, there is always a partition of P into k clusters whose sizes are "almost" (except for adding "fake" points of zero costs) integer powers of $(1+\epsilon)$ and whose cost is upper bounded by at most $(1+\epsilon)$ times the partition of P into k clusters $C_1^*, \dots, C_k^*$ with sizes $N_1, \dots, N_k$ that minimizes the sum $\sum_{i=1}^{k} |C_i^*| \sum_{p \in C_i^*} d(p, c_i)$.

Lemma 10 *Let $\mathcal{C} = (c_1, \dots, c_k)$ and let $N_1, \dots, N_k$ be a set of positive integers with $\sum_{i=1}^{k} N_i = n$. Let $C_1^*, \dots, C_k^*$ be a partition of P with $|C_i^*| = N_i$ for every i, that minimizes $\sum_{i=1}^{k} |C_i^*| \sum_{p \in C_i^*} d(p, c_i)$. Let $\lambda_1, \dots, \lambda_k$ be such that each λ_i is of the form $\lfloor (1+\epsilon)^m \rfloor$ for an integer m and such that $N_i \leq \lambda_i < (1+\epsilon) \cdot N_i$. Let $r = \sum_{i=1}^{k} (\lambda_i - N_i)$. Let $U_1, \dots U_k$ be a partition of the set $P \cup \mathbb{O}_r$ into k sets such that*

$|U_i| = \lambda_i$ *for every* i*, and so that the sum* $\sum_{i=1}^{k} |U_i| \sum_{u \in U_i} d(u, c_i)$ *is minimized. If we define a partition of* P *into* k *clusters* $C_1, \dots, C_k$ *by setting* $C_i = U_i \cap P$ *for every* i*, then*

$$\sum_{i=1}^{k} |C_i| \sum_{u \in C_i} d(u, c_i) \quad < \quad (1+\epsilon) \cdot \sum_{i=1}^{k} |C_i^*| \sum_{u \in C_i^*} d(u, c_i) \ .$$

Proof: Consider a partition of $P \cup \mathbb{O}_r$ into k sets $C_1^* \cup \mathbb{O}_{\lambda_1 - N_1}, \dots, C_k^* \cup \mathbb{O}_{\lambda_k - N_k}$. Since $\lambda_i - N_i < \epsilon \cdot N_i$ and since $U_1, \dots, U_k$ is an optimal partition of $P \cup \mathbb{O}_r$ subject to the constraints on the sizes of the clusters, we have the following bound:

$$\begin{aligned} \sum_{i=1}^{k} |C_i| \sum_{u \in C_i} d(u, c_i) \quad &\le \quad \sum_{i=1}^{k} |U_i| \sum_{u \in U_i} d(u, c_i) \ \le \ \sum_{i=1}^{k} |C_i^* \cup \mathbb{O}_{\lambda_i - N_i}| \sum_{u \in C_i^* \cup \mathbb{O}_{\lambda_i - N_i}} d(u, c_i) \\ &= \quad \sum_{i=1}^{k} \lambda_i \sum_{u \in C_i^*} d(u, c_i) \ < \ \sum_{i=1}^{k} (1+\epsilon) \cdot |C_i^*| \sum_{u \in C_i^*} d(u, c_i) \ . \end{aligned}$$

□

We can use Lemma 10 by considering all possible values for λ_i to obtain the following result.

Lemma 11 *Let* $\mathcal{C} = \{c_1, \dots, c_k\}$ *an arbitrary set of* k *centers. Then, in time* $(\mathcal{O}(n\,k) + \widetilde{\mathcal{O}}(\sqrt{n}\,k^{2.5})) \cdot (\mathcal{O}(\log n/\epsilon))^k$ *one can find a partition of* P *into* k *clusters* $C_1, \dots, C_k$ *such that*

$$\textit{cost}_k(P, \mathcal{C}) \quad \le \quad \sum_{k=1}^{k} |C_i| \sum_{u \in C_i} d(u, c_i) \ \le \ (1+\epsilon) \cdot \textit{cost}_k(P, \mathcal{C}) \ .$$

Proof: Since the first inequality $\textit{cost}_k(P, \mathcal{C}) \le \sum_{k=1}^{k} \sum_{k=1}^{k} |C_i| \sum_{u \in C_i} d(u, c_i)$ holds for every partition of P into clusters $C_1, \dots, C_k$, we only have to prove the second inequality.

Take the optimal partition of P into k clusters $C_1^*, \dots, C_k^*$ that minimizes $\sum_{k=1}^{k} |C_i^*| \sum_{u \in C_i^*} d(u, c_i)$. Let $N_1, \dots, N_k$ be such that $|C_i^*| = N_i$ for every i. We observe that among all sequences $\langle \lambda_1, \dots, \lambda_k \rangle$ such that each λ_i is of the form $\lfloor (1+\epsilon)^m \rfloor$ for an integer m and $n \le \sum_{i=1}^{k} \lambda_i < (1+\epsilon)\,n$, there must be a sequence $\langle \lambda_1^*, \dots, \lambda_k^* \rangle$ with $N_i \le \lambda_i^* < (1+\epsilon) \cdot N_i$ for every i. Take such $\lambda_1^*, \dots, \lambda_k^*$ and for an optimal partition of $P \cup \mathbb{O}_r$ (with $r = \sum_{i=1}^{k} (\lambda_i^* - N_i)$) into k clusters $U_1, \dots, U_k$ of sizes $\lambda_1^*, \dots, \lambda_k^*$ and with the centers $\mathcal{C} = \{c_1, \dots, c_k\}$, define the partition of P into clusters $C_1, \dots, C_k$ with $C_i = U_i \cap P$. By Lemma 10, the cost of that partition is upper bounded by $(1+\epsilon) \cdot \textit{cost}_k(P, \mathcal{C})$.

Now, we will find such a clustering, or a clustering which is not worse. We consider all sequences $\langle \lambda_1, \dots, \lambda_k \rangle$ such that each λ_i is of the form $\lfloor (1+\epsilon)^m \rfloor$ for an integer m and $n \le \sum_{i=1}^{k} \lambda_i < (1+\epsilon)\,n$. This will give us not more than $(\lceil \log_{1+\epsilon} n \rceil)^k = (\mathcal{O}(\log n/\epsilon))^k$ different problem instances. For each problem instance, we apply the algorithm of Tokuyama and Nakano [47] to find an optimal partition of $P \cup \mathbb{O}_r$ (with $r = \sum_{i=1}^{k} (\lambda_i - N_i)$) into k clusters $U_1, \dots, U_k$ of sizes $\lambda_1, \dots, \lambda_k$ and with the centers $\mathcal{C} = \{c_1, \dots, c_k\}$. Then, we take the partition of P into $C_1, \dots, C_k$ with $C_i = U_i \cap P$ to be associated with the given instance $\lambda_1, \dots, \lambda_k$. Once we found partitions of P associated with all instances $\lambda_1, \dots, \lambda_k$, the best of them will have the cost not worse than the partition of P associated with the instance $\lambda_1^*, \dots, \lambda_k^*$, and hence, its cost will be upper bounded by $(1+\epsilon) \cdot \textit{cost}_k(P, \mathcal{C})$; we will return this partition.

To complete the proof, we only observe that using the algorithm of Tokuyama and Nakano [47], for any given $\langle \lambda_1, \dots, \lambda_k \rangle$, we can find an optimal clustering (as described above) in time $\mathcal{O}(n\,k) + \widetilde{\mathcal{O}}(\sqrt{n}\,k^{2.5})$. This yields the promised running time of our algorithm, because we consider $(\mathcal{O}(\log n/\epsilon))^k$ instances. □

4.2 Constant-factor approximations for balanced k-median and min-sum k-clustering

Now we summarize our discussion in previous sections (Lemmas 10 and 11) and present approximation algorithms for clustering.

Theorem 12 *For every $\varepsilon \le 1/8$, there is a randomized algorithm for balanced k-median that finds a $(2+\varepsilon)$-approximate solution in time $(\mathcal{O}(n\,k) + \widetilde{\mathcal{O}}(\sqrt{n}\,k^{2.5})) \cdot (\mathcal{O}(k \log^2 n \log(n/\varepsilon)/\varepsilon^4))^k$. The algorithm fails with probability at most $1/poly(n)$. It returns also a $(4+\varepsilon)$-approximation for min-sum k-clustering.*

Proof: We first apply Theorem 9 to find a set S of $\mathcal{O}(k \log n \log(n/\epsilon)/\epsilon^3)$ points from P for which (with probability at least $1-1/poly(n)$) there is a set $\mathcal{C}_O$ of k points in S satisfying $cost(P, \mathcal{C}_O) \le (2+\epsilon)\cdot cost_k(P)$. Next, we take every subset $\mathcal{C}$ of S of size k and apply Lemma 11 to find for $\mathcal{C}$ an almost optimal k clustering $C_1, \dots, C_k$ that satisfies

$$\sum_{k=1}^{k} |C_i| \sum_{u \in C_i} d(u, c_i) \quad \le \quad (1+\epsilon)\cdot cost_k(P, \mathcal{C}) \ .$$

This bound together with the result from Theorem 9 implies that our algorithm will find (with probability at least $1 - 1/poly(n)$) a set of k clusters $\mathcal{C} = (c_1, \dots, c_k)$ and a k-clustering $C_1, \dots, C_k$ of P that satisfies the following:

$$\sum_{k=1}^{k} |C_i| \sum_{u \in C_i} d(u, c_i) \quad \le \quad (1+\epsilon)\cdot cost_k(P, \mathcal{C}_O) \ \le \ (1+\epsilon)\cdot(2+\epsilon)\cdot cost_k(P) \ ,$$

what gives an $(2+\varepsilon)$ approximation for the balanced k-median problem by setting $\varepsilon = \epsilon/4$.

The running time of the algorithm follows directly from Theorem 9 and Lemma 11. □

Since a solution to balanced k-median is a 2-approximation for min-sum k-clustering, we obtain the following corollary of Theorem 12.

Theorem 13 *For every $\varepsilon \le 1/16$, there is a randomized algorithm for min-sum k-clustering that finds a $(4+\varepsilon)$-approximate solution and runs in time $(\mathcal{O}(n\,k) + \widetilde{\mathcal{O}}(\sqrt{n}\,k^{2.5})) \cdot (\mathcal{O}(k \log^2 n \log(n/\epsilon)/\epsilon^4))^k$. The algorithm fails with probability at most $1/poly(n)$.* □

Basic calculations lead to the following corollary that highlights the results from Theorems 12 and 13.

Corollary 14 *For every positive constant ϵ and for every $k = \mathcal{O}(\log n/\log\log n)$, there are a polynomial-time randomized $(4+\varepsilon)$-approximation algorithm for min-sum k-clustering and a randomized $(2+\varepsilon)$-approximation algorithm for balanced k-median and a polynomial-time. Each of these algorithms fails with probability at most $1/poly(n)$.*

Furthermore, if in addition $k \le (\frac{1}{4} - o(1)) \cdot \frac{\log n}{\log\log n}$, then the running time of these algorithms are $o(n^2)$, that is, are sublinear in the input size.

Finally, if k is a constant, then the running times are $\widetilde{\mathcal{O}}(n)$. □

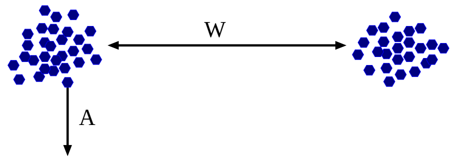

Figure 3: Consider a point set consisting of two well separated clusters with mutual distance at least W (W large). Moving a point in one cluster by a distance of $A \ll W$ may change the cost of the clustering arbitrarily, but the corresponding embedding has low average distortion.

5 α-preserving embeddings and Coresets

Let Π be any minimization problem on metric spaces that is *monotone with respect to distances*, i.e., if the distances in the metric space increase then the cost of an optimal solution for Π does not decrease. We say that an embedding from a metric space (P, d) into (P, d') is *α-preserving for* Π, if

(a) the distance between any pair of points does not decrease, and

(b) the cost of an optimal solution of Π for (P, d') is at most an α-factor larger than that for (P, d).

5.1 Basic properties of α-preserving embeddings

The concept of α-preserving embeddings can be viewed as a generalization of coresets. Recall that, intuitively, a coreset is a small (weighted) set of objects that approximates a much larger input set with respect to a given optimization problem. The idea is that for many problems it is possible to capture approximately the relevant structure of the problem using only few input objects. This approach has been used very successfully in the past to obtain a large number of improvements for geometric approximation problems.

Reduction of the number of input objects is only one possible approach to reduce the complexity of a problem while still maintaining a good approximation. An α-preserving embedding aims at mapping a metric space to a "less complex" one, while still preserving an α-approximation for the problem under consideration. In this context, "less complex" may mean a metric space with many similar points, i.e., a coreset. However, "less complex" may also mean to go from an arbitrary metric to the Euclidean space or to a (distribution of) tree metric (like in the context of low-distortion embeddings). We hope that by making the embedding dependent on the particular problem, one can achieve stronger approximation guarantees than in the general case (though, of course, the results only apply to one problem).

For example, there is a trivial 1-preserving embedding for the k-median problem from arbitrary metric spaces in tree metrics. This embedding can be constructed by using only the edges of the original space that connect the input points to the nearest center in an optimal solution plus some arbitrary tree connecting the centers of an optimal solution. Clearly, this embedding is 1-preserving, but it cannot be computed in polynomial time. However, this leads to the interesting open question, whether it is possible to compute a 2-preserving embedding in polynomial time (and hence obtain a 2-approximation algorithm for the k-median problem).

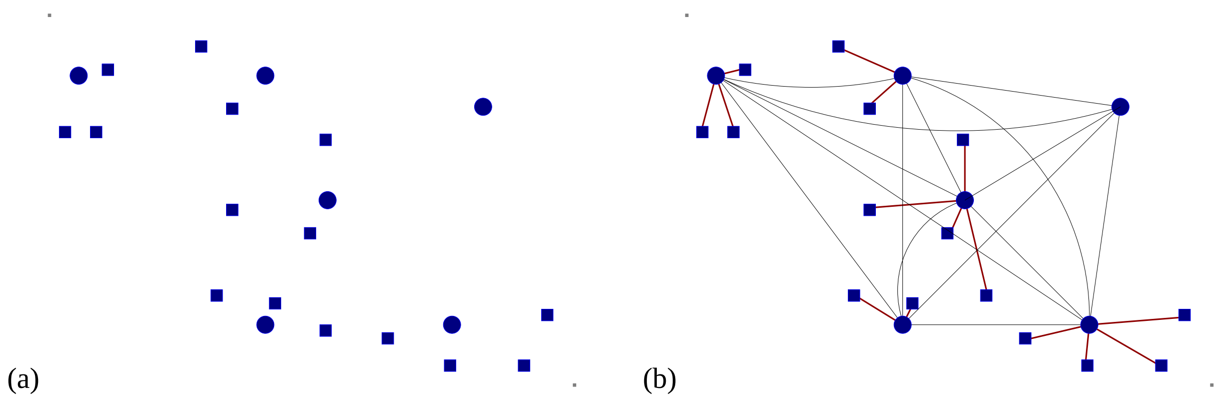

Figure 4: An example describing the construction of $(6+\epsilon)$-preserving embedding for balanced k-median. Figure (a) presents a set of points on the plane with points marked by a disk corresponding to the points from S and points marked by squares corresponding to the points in $P \setminus S$. Figure (b) presents the graph G.

Applied to the min-sum k-clustering problem, the concept of α-preserving embeddings might seem to be closely related to embeddings with constant average distortion (see, e.g., [1, 43]). However, an embedding with small average distortion does not preserve the clustering cost of min-sum k-clustering up to a constant factor and hence it is a too weak requirement. This can be easily seen by considering min-sum 2-clustering for an input of two well-separated clusters (see Figure 3). Let us assume the distance between the two clusters is at least W. Since W is unrelated to the cost of the clustering, we can make W arbitrarily large compared to the cost of the clustering. Then it is easy to see that, for example, moving all points of each cluster arbitrarily close to each other will not change the average distortion. However, this will change the cost of the clustering arbitrarily.

5.2 $(6+\epsilon)$-preserving embedding for balanced k-median

We begin with a $(6+\epsilon)$-preserving embedding into a *shortest path metric* (P, d_G) on a graph $G = (P, E)$ consisting of *poly*$(k, \log n, 1/\epsilon)$ *stars* whose centers (which we call *anchor points*) are connected in a clique.

Let G be the graph whose edges are defined by the following procedure (see also Figure 4):

- Choose the set of anchor points to be the set S computed by our algorithm PARTITIONINGSCHEME.
- Every pair of anchor points $p, q \in S$ is connected by an edge with distance the same as in the metric space (P, d), i.e., $d_G(p, q) = d(p, q)$.
- Every point $p \in P \setminus S$ is connected by an edge to the nearest anchor point; the cost of this edge is the same as in the metric space (P, d).

The next lemma follows essentially from the triangle inequality.

Lemma 15 *The embedding of* (P, d) *into* (P, d_G) *as described above is a* $(6+\epsilon)$*-preserving embedding for the balanced* k*-median problem.*

Proof: Since G is a subgraph of the complete graph describing the metric space (P, d), $d(p, q) \le d_G(p, q)$ holds for every pair of points $p, q \in P$. Hence, the property (a) of α-preserving embeddings holds.

Now we prove property (b). We first show that $d_G(p, q) \le 3d(p, q)$ for every $p \in S$ and $q \in P$. If $q \in S$ or q is adjacent to p in G, then $d_G(p, q) = d(p, q)$ and hence the inequality trivially holds. Otherwise, we consider the case that q is not adjacent to p in G. Let $r \in S$ be the anchor point adjacent to

q. Since r is the nearest anchor point to q, we have $d(p,q) \geq d(q,r)$. It follows

$$d_G(p,q) \;=\; d(p,r) + d(r,q) \;\leq\; d(p,q) + d(q,r) + d(r,q) \;\leq\; 3 \cdot d(p,q) \enspace .$$

By Theorem 9, the set S returned by PARTITIONINGSCHEME contains a set $\mathcal{C}$ such that $cost_k(P,\mathcal{C}) \leq (2+\epsilon)\cdot Opt_k$. Since for any points $p \in P$ and $q \in S$ we have $d_G(p,q) \leq 3 \cdot d(p,q)$, we immediately get that the cost of the corresponding optimal partition increases at most by a factor of 3 when we replace $d(\cdot,\cdot)$ by the shortest path metric in G. □

5.3 Coresets for balanced k-median

In this section, we will design a coreset for balanced k-median for a given input metric. We use the following notation of a coreset for the balanced k-median problem.

Definition 16 (α-coresets) *Let (P,d) be a metric space with $|P| = n$. Let $Q = \{q_1, \ldots, q_s\} \subseteq P$ be a set of poly$(\log n)$ points with integer weights $w_1, \ldots, w_s$ that sum up to n. Let $\alpha_1 \cdot \alpha_2 \leq \alpha$ be positive parameters. We say that the metric space (Q,d') is an α-coreset for (P,d) for the balanced k-median problem, if there exists a mapping $\tau : P \to Q$ with $|\tau^{-1}(q_i)| = w_i$ such that*

- *for every partition of P into sets $C_1, \ldots C_k$ and centers $c_1, \ldots, c_k \in Q$ we have*

$$\frac{1}{\alpha_1} \cdot \sum_{i=1}^{k} |C_i| \sum_{p \in C_i} d(p, c_i) \;\leq\; \sum_{i=1}^{k} |C_i| \sum_{p \in C_i} d'(\tau(p), \tau(c_i)) \enspace ,$$

- *there exists a partition $C_1, \ldots, C_k$ of P and centers $c_1, \ldots, c_k$ such that*

$$\sum_{i=1}^{k} |C_i| \sum_{p \in C_i} d'(\tau(p), \tau(c_i)) \;\leq\; \alpha_2 \cdot Opt_k \enspace .$$

The construction. To construct the coreset, we consider the graph G defined in Section 5.2, and modify some of its distances and group some of its vertices together. We begin with computing a constant approximation to the cost c_{med} of the k-median problem and use $\widetilde{Opt}_k = n \cdot c_{med}$ as a rough approximation of the cost of the balanced k-median. Then, we increase all edge distances smaller than $\epsilon \cdot \widetilde{Opt}_k/(c_{med} \cdot n^3)$ to the value of $\epsilon \cdot \widetilde{Opt}_k/(c_{med} \cdot n^3)$; it is easy to see that this transformation increases the cost of the optimal solution by not more than a factor of $(1+\epsilon)$. Next, let F be the set of edges connecting anchor points to non-anchor points, $F = \{(u,v) \in G : u \in S \text{ and } v \in P \setminus S\}$. We round the distances corresponding to these edges up to the next power of $(1+\epsilon)$. Clearly, this changes the cost of any solution by at most $(1+\epsilon)$ and only increases edge weights. Since there can be no edges in F with cost more than Opt_k, there are at most $\mathcal{O}(\log(n/\epsilon)/\epsilon)$ different edge weights.

Let H denote the graph with these modified distances and let d_H denote the shortest path metric on H. We group the points from $P \setminus S$ incident to the same anchor point $p \in S$ by their edge weights in H. This way, we obtain $\mathcal{O}(\log(n/\epsilon)/\epsilon)$ groups V_j^p such that each q in V_j^p is incident to $p \in S$ via an edge with cost w_j. Since all points in V_j^p have equal distance to p, we would like to replace them by a single weighted vertex with weight $|V_j^p|$. The only difficulty stems from the fact that the points in V_j^p have pairwise distance $2 \cdot w_j$ in H and this distance is reduced to 0 when we join them. To reduce this effect, we subdivide V_j^p into groups $V_{j,1}^p, \ldots, V_{j,t}^p$ of size $|V_j^p|/t$ for $t = 32 \cdot (k/\epsilon)^{3/2}$ (for simplicity, we assume that $|V_j^p|/t$ is integral). Each group $V^p_{j,\ell}$, $1 \le \ell \le t$, is replaced by a single vertex of weight $|V^p_j|/t$. Let us define the mapping τ according to this replacement, i.e., every anchor point is mapped to the corresponding anchor point, every non-anchor point is mapped by τ to its replacement. Let us call the resulting new graph H' and let $d_{H'}$ be the shortest path metric induced by H'. H' has $|S| = \mathcal{O}(k \log n \log(n/\epsilon)/\epsilon^3)$ anchor points, $\mathcal{O}(|S|\, t \log(n/\epsilon)/\epsilon)$ other points, and $\mathcal{O}(|S|)^2 + \mathcal{O}(t\, k \log n \log^2(n/\epsilon)/\epsilon^4)$ edges. In particular, if we set $t = 32 \cdot (k/\epsilon)^{3/2}$, then H' has $\mathcal{O}(k^{2.5} \log n \log^2(n/\epsilon)/\epsilon^{5.5})$ points and $\mathcal{O}((k^2 \log n \log^2(n/\epsilon)/\epsilon^5) \cdot (\log n + \sqrt{k/\epsilon}))$ edges.

The shortest path metric on the weighted graph H' is a coreset. In this section, we will prove Theorem 21, that is, that the shortest path metric on the weighted graph H' is a coreset.

We first prove the second condition in Definition 16. By Lemma 15 we know that the embedding of (P, d) into (P, d_H) is $(6 + \epsilon)$-preserving. Hence, there exists a set of k centers $\{c_1, \dots, c_k\}$ and a corresponding partition $C_1, \dots, C_k$ such that

$$\sum_{i=1}^{k} |C_i| \cdot \sum_{p \in C_i} d_H(p, c_i) \le (6 + \epsilon) \cdot \mathit{Opt}_k \ .$$

Therefore, the second coreset condition in Definition 16 follows now directly from the fact that for any pair of point p, q we have $d_H(p, q) \ge d_{H'}(\tau(p), \tau(q))$.

It remains to prove the first condition of Definition 16. This can be done as follows. Let $C_1, \dots, C_k$ be an arbitrary partition of P and let $c_1, \dots, c_k \in P$ be the corresponding centers. We only have to prove that

$$(1-\epsilon) \cdot \sum_{i=1}^{k} |C_i| \sum_{q \in C_i} d(q, c_i) \le (1-\epsilon) \cdot \sum_{i=1}^{k} |C_i| \sum_{q \in C_i} d_H(q, c_i) \le \sum_{i=1}^{k} |C_i| \sum_{q \in C_i} d_{H'}(\tau(q), \tau(c_i)) \ . \quad (1)$$

The first inequality follows immediately from the fact that for any pair of vertices p, q we have $d(p, q) \le d_H(p, q)$. Therefore, we concentrate on the second inequality. Let us denote by $\mathit{con}(q)$ the contribution of $q \in P$ to the cost (with respect to d_H) of the partitioning $C_1, \dots, C_k$ with centers $c_1, \dots, c_k$, i.e., $\mathit{con}(q) = |C_\ell| \cdot d_H(q, c_\ell)$, where ℓ is the index of the cluster C_ℓ that contains q. In a similar way, we define $\mathit{con}_\tau(q)$ as the contribution of $\tau(q)$ to the clustering $\tau(C_1), \dots, \tau(C_k)$, i.e., $\mathit{con}_\tau(q) = |C_\ell| \cdot d_{H'}(\tau(q), \tau(c_\ell))$, where ℓ is the index of the cluster C_ℓ that contains q.

To prove inequality (1), we consider each set V^p_j separately and show that when we move from $d_H(.,.)$ to $d_{H'}(.,.)$, the contribution of V^p_j to the cost of the current solution is reduced by at most a factor of $(1-\epsilon)$, what will directly yield (1). We assume that $|V^p_j| \ge t$. Otherwise, the distances from V^p_j to all other vertices are the same in d_h and $d_{H'}$.

Claim 17

$$\sum_{q \in V^p_j} \mathit{con}(q) \ge w_j \cdot (|V^p_j|/k)^2 \ .$$

Proof: There exists a cluster C_i that contains at least $|V^p_j|/k$ vertices from V^p_j. The center of this cluster can either be one of these vertices or not. In the first case, we have $\mathit{con}(q) = 0$ for the cluster center and $\mathit{con}(q) \ge 2 \cdot w_j \cdot |C_i| \ge 2 \cdot w_j \cdot |V^p_j|/k$ for all other vertices from $V^p_j \cap C_i$. In the second case, we have $\mathit{con}(q) \ge w_j$ for every $q \in V^p_j$ and the claim follows immediately. □

Claim 18 *Let C_i be an arbitrary cluster. If $c_i \notin V_j^p$ then we have*

$$\sum_{q \in V_j^p \cap C_i} con(q) = \sum_{q \in V_j^p \cap C_i} con_\tau(q) \ .$$

Proof: If $c_i \notin V_j^p$ then $d_H(q, c_i) = d_{H'}(\tau(q), \tau(c_i))$ for all vertices $q \in V_j^p$. Hence, the claim follows. □

Claim 19 *Let C_i be an arbitrary cluster with $c_i \in V_j^p$. If $|C_i| \le 8 \cdot k \cdot |V_j^p|/(\epsilon \cdot t)$ then we have*

$$\sum_{q \in V_j^p \cap C_i} con(q) - \sum_{q \in V_j^p \cap C_i} con_\tau(q) \le 2 \cdot w_j \cdot (8 \cdot k \cdot |V_j^p|/(\epsilon \cdot t))^2 \ .$$

Proof: We have $d_H(q, r) = 2 \cdot w_j$ for every pair $q, r \in V_j^p$ with $q \ne r$. In the worst case, this distance is reduced to 0. Since $|C_i| \le 8 \cdot k \cdot |V_j^p|/(\epsilon \cdot t)$ the claim follows by summing up over all $q \in V_j^p \cap C_i$. □

Claim 20 *Let C_i be an arbitrary cluster with $c_i \in V_j^p$. If $|C_i| \ge 8 \cdot k \cdot |V_j^p|/(\epsilon \cdot t)$ then we have*

$$\sum_{q \in V_j^p \cap C_i} con(q) - \sum_{q \in V_j^p \cap C_i} con_\tau(q) \le \frac{\epsilon}{2k} \cdot \sum_{q \in C_i} con(q) \ .$$

Proof: At most $|V_j^p|/t$ points are mapped to the same point as c_i. For these points the distance is reduced from $2 \cdot w_j$ to 0. This means that due to this change of distances the cost of the cluster is reduced by at most $2 \cdot w_j \cdot |V_j^p|/t \cdot |C_i|$, that is,

$$\sum_{q \in V_j^p \cap C_i} con(q) - \sum_{q \in V_j^p \cap C_i} con_\tau(q) \le 2 \cdot w_j \cdot |V_j^p|/t \cdot |C_i| \ .$$

For all other points $r \in C_i$ we have that $d_H(r, c_i) = d_{H'}(r, c_i) \ge w_j$. Hence,

$$\begin{aligned} \sum_{q \in C_i} con(q) &= |C_i| \cdot \sum_{q \in C_i} d(q, c_i) \ \ge \ |C_i| \cdot \sum_{q \in C_i} w_j \ \ge \ w_j \cdot |C_i| \cdot (|C_i| - 1) \\ &\ge \ w_j \cdot |C_i|^2/2 \ \ge \ 4 \cdot k \cdot w_j \cdot |C_i| \cdot |V_j^p|/(\epsilon \cdot t) \ . \end{aligned}$$

From this the claim follows immediately. □

Our next step is to put these four claims together. We get,

$$\begin{aligned} \sum_{q \in P} con(q) &= \sum_{i=1}^{k} \sum_{q \in C_i} con(q) \\ &\le \sum_{j : \exists c_i \in V_j^p} 2 \cdot w_j \cdot (2 \cdot |V_j^p|/(\epsilon \cdot t))^2 + \frac{\epsilon}{2} \sum_{q \in P} con(q) + \sum_{i=1}^{k} \sum_{q \in C_i} con_\tau(q) \end{aligned}$$

by Claims 18–20 and the observation that since the sets V_j^p are disjoint, Claims 19 and 20 are applied at most k times. By Claim 17 it then follows,

$$\begin{aligned}&\sum_{j:\exists c_i \in V_j^p} 2 \cdot w_j \cdot (8 \cdot k \cdot |V_j^p|/(\epsilon \cdot t))^2 + \frac{\epsilon}{2} \cdot \sum_{q \in P} con(q) + \sum_{i=1}^{k} \sum_{q \in C_i} con_\tau(q) \\ \leq\ & \frac{128 \cdot k^3}{(\epsilon \cdot t)^2} \cdot \sum_{q \in P} con(q) + \epsilon \cdot \sum_{q \in P} con(q) + \sum_{i=1}^{k} \sum_{q \in C_i} con_\tau(q) \ .\end{aligned}$$

Setting $t = 32 \cdot (k/\epsilon)^{3/2}$ we obtain

$$(1-\epsilon) \cdot \sum_{q \in P} con(q) \leq \sum_{q \in P} con_\tau(q) \ .$$

Now, we can conclude with the following theorem.

Theorem 21 *The shortest path metric on the weighted graph H' defined above is a $(6+\epsilon)$-coreset (Q, d') for the balanced k-median problem. It can be computed in $\widetilde{\mathcal{O}}(n \cdot k \cdot \ln(1/\delta)/\epsilon^3)$ time with probability $1-\delta$. It is also a $(12+\epsilon)$-coreset (Q, d') for the min-sum k-clustering problem.*

6 Faster constant-factor approximations for clustering problems

In this section, we briefly discuss an application of our coreset construction to improve the running time (and implicitly, space complexity) of the algorithm for balanced k-median from Theorem 12. The price we pay for the speed-up is a slightly weaker approximation bound; we obtain the approximation ratio of $6+\epsilon$.

Theorem 22 *For every $\varepsilon \leq \frac{1}{8}$, there is a randomized $(6+\varepsilon)$-approximation algorithm for balanced k-median that runs in time $\mathcal{O}(n\,k\,\log^2 n\,\log(n/\epsilon)/\epsilon^3) + (\mathcal{O}(k\,\log n/\epsilon))^{\mathcal{O}(k)}$. The same result holds for an $(12+\varepsilon)$-approximation for min-sum k-clustering. The algorithms fail with probability at most $1/poly(n)$.*

Proof: We first build a $(6+\epsilon)$-coreset for balanced k-median, as promised in Theorem 21. The coreset is represented by a weighted graph H' of size $poly(k, \log n, 1/\epsilon)$, in which vertices may represent multiple points from P; the multiplicity is specified by the weight of the vertex. For such a graph, we can apply a modification of the algorithm by Tokuyama and Nakano [47], similar to that discussed in Section 4, but this time, we additionally will use the compressed representation of (P, d) by the graph H'. In particular, since this graph has only $poly(k, \log n, 1/\epsilon)$ nodes, we can modify the algorithm from Section 4 to run in time $(k\,\log n/\epsilon)^{\mathcal{O}(k)}$. Therefore, the total running time is the sum of the time needed to create the $(6+\epsilon)$-coreset, which is $\mathcal{O}(n \cdot k \cdot \log^2 n \cdot \log(n/\epsilon)/\epsilon^3)$, and the time needed to find an approximate solution using the modified algorithm from Section 4, which is $(k\,\log n/\epsilon)^{\mathcal{O}(k)}$. □

7 2-pass streaming algorithm for min-sum clustering and balanced k-median

In this section, we present an application of our coresets to design the first efficient *streaming* algorithm for min-sum k-clustering and balanced k-median. Our algorithm uses two passes and polylogarithmic space.

A streaming algorithm for a problem in a metric space (P, d) takes as its input set P as a read-only sequence of points $p_1, \ldots, p_n$ given one-by-one in an arbitrary order. It can evaluate $d(p, q)$ in constant time for any two points p, q stored in local memory. The capacity of the local memory is polylogarithmic. A c-pass streaming algorithm is an algorithm that makes c passes over the input sequence. Using our results from previous sections, we can obtain a 2-pass data streaming algorithm for the balanced k-median and the min-sum k-clustering problems. Our algorithm begins with computing the coreset from Theorem 21. To do this, we first observe that RANDOMPARTITION is a data streaming algorithm. We can implement PARTITIONINGSCHEME by executing all calls to RANDOMPARTITION in parallel. This way we get a one pass algorithm to compute the set S. During the first pass we can also compute a constant factor approximation for the cost of k-median using the algorithm from [11]. In a second pass we can compute for every point in P its distance to the nearest neighbor in S and round it up to the next power of $(1+\epsilon)$. From this representation we can compute our coreset and from the coreset we can compute (an implicit representation of) a constant approximation to min-sum k-clustering using a variation of the algorithm used in Theorem 12.

Theorem 23 *There is a 2-pass, $\mathcal{O}(1)$-approximation algorithm for min-sum k-clustering and balanced k-median. The algorithm uses poly$(\log n, k)$ space. The time to compute the clustering is $\mathcal{O}(k \log n))^{\mathcal{O}(k)}$.*

8 Sublinear-time polylogarithmic-approximation factor algorithm

In Section 4, we presented a very fast algorithm that finds a $(2+\epsilon)$-approximation for balanced k-median and $(4+\epsilon)$-approximation for min-sum k-clustering. The running time of this algorithm is $o(n^2)$ (and hence, sublinear) for all values of k such that $k \le (\frac{1}{4} - o(1)) \cdot \frac{\log n}{\log(\log n/\epsilon)}$. A natural question is if we can apply similar techniques to obtain good polynomial-time approximation algorithms for larger values of k, in particular, for $k = \omega(\log n/\log\log n)$. Prior to our work, for values of $k = \omega(1)$, the best polynomial-time algorithm achieved the approximation ratio of $\mathcal{O}(c \log^{1+1/c} n)$ [8], for an arbitrary $c > 0$; the running time is $\Omega(n^{2+\mathcal{O}(1/c)})$. In this section, we combine the arguments from [8] with the techniques developed in previous sections to obtain an algorithm that has a similar approximation guarantee as that in [8], but which at the same time has superior running time: it runs in sublinear time, that is, in $o(n^2)$ time!

Let us define an extension of the balanced k-median problem: a *splittable weighted balanced k-median problem*. Let Q be a set of N points in a metric space. Let $w : Q \to \mathbb{N}$ be a function denoting the multiplicity of every point in Q and let Q^* be the multiset defined by taking every point $q \in Q$ with multiplicity $w(q)$. The *splittable weighted balanced k-median* problem for Q is to solve the balanced k-median problem for the set Q^*, i.e., to find a set of k points $c_1, \dots, c_k$ in Q^* and a partition of Q^* into $Q_1^*, \dots, Q_k^*$ that minimizes

$$\sum_{i=1}^{k} |Q_i^*| \cdot \sum_{q \in Q_i^*} d(q, c_i) \ .$$

The intuition behind our use of the splittable weighted balanced k-median problem is that it represents a problem instance of the balanced k-median problem in which some points may have identical locations, and that our hope is that it is possible to obtain an algorithm with the running time being a function of N, number of locations of the point, rather than n, number of points in the multiset.

Our key result is an extension of the algorithm for balanced k-median due to Bartal et al. [8] to the splittable weighted case:

Theorem 24 *Let λ be an arbitrary positive real. There exists an algorithm for the splittable weighted balanced k-median problem that in time $\mathcal{O}(N^2 + N \cdot k^2 \cdot W^{1/\lambda})$ approximates the optimal solution to within $\mathcal{O}(\frac{1}{\lambda} \cdot \log N \cdot (\log W)^{\mathcal{O}(\lambda)})$ factor, where $W = \sum_{q \in Q} w(q)$.*

The algorithm in Theorem 24 is an extension of the algorithm for balanced k-median due to Bartal et al. [8] to the splittable weighted case. Using the approach of Bartal et al. [8], we can reduce the problem to that of solving the splittable weighted balanced k-median problem in a σ-HST (hierarchically well-separated tree). If we combine their result with the recent algorithm for constructing HSTs in [17], then a λ-approximation algorithm for the splittable weighted balanced k-median problem in a σ-HST can be transformed in an $\mathcal{O}(N^2)$-time into a $\mathcal{O}(\lambda \cdot \sigma \cdot \log_\sigma N)$-approximation algorithm for the weighted balanced k-median problem in any metric space. This algorithm is obtained by a rather straightforward modification of the techniques used in [8], but for the sake of completeness, we present them in Section 9.

Once we have Theorem 24, we can combine it with our construction of coresets to directly obtain the following theorem.

Theorem 25 *There is a randomized $(\log n)^{\mathcal{O}(1)}$-approximation algorithm for balanced k-median and min-sum clustering that runs in $\mathcal{O}(n\, k^{\mathcal{O}(1)}\, (\log n)^{\mathcal{O}(1)})$ time. The algorithm fails with probability at most $\frac{1}{poly(n)}$.*

9 Approximating the splittable weighted balanced k-median problem

The goal of this section is to prove Theorem 24. Our algorithm is an extension of an algorithm due to Bartal et al. [8] and it uses many of the ideas developed therein. Before we will proceed with the details, let us first define the concept of *hierarchical well-separated trees* (SHSTs) [6, 7, 8, 17] (in this presentation we tightly follow [8]).

Definition 26 (σ-SHST) *For $\sigma > 1$, a* σ-hierarchical well-separated tree (σ-SHST) *(in a special form) is a metric space defined on the leaves of a weighted rooted tree T as described below.*

Let the level of an internal node in the tree be the number of edges on the path to the root. Let Δ denote the diameter of the resulting metric space. For any node $u \in T$, let $\Delta(u)$ denote the diameter of the subtree rooted at u. Then the tree T has the following properties:

- *all edges at a particular level are of the same weight (i.e., the same distance from the root),*
- *all leaves are at the same level, and*
- *for any internal node u at level i, $\Delta(u) = \Delta \cdot \sigma^{-i}$.*

Notice that in any σ-SHST, if $\sigma > 2$, then edge weights in level i are $\Delta \cdot (\sigma^{-i} - \sigma^{-i-1})/2$ and the edge weights at the last level ℓ are $\Delta \cdot \sigma^{-\ell}/2$. Furthermore, the distance between two leaves $u, v \in T$ is equal to $\Delta(\textsc{lca}(u,v))$, where $\textsc{lca}(u,v)$ is the least common ancestor of u and v in T.

It is now not difficult to see (see also [6, 7, 8]) that the recent algorithm for constructing HSTs in [17] implies that a λ-approximation algorithm for the splittable weighted balanced k-median problem in a σ-HST can be transformed in an $\mathcal{O}(N^2)$-time into a $\mathcal{O}(\lambda \cdot \sigma \cdot \log N / \log \sigma)$-approximation algorithm for the splittable weighted balanced k-median problem in any metric space (this construction is randomized). Therefore, from now, following the approach of Bartal et al. [8, Section 3.3], we will focus on the problem of solving splittable weighted balanced k-median in SHSTs only.

9.1 Solving splittable weighted balanced k-median in SHSTs

Bartal et al. [8, Section 3.3] presented a dynamic programming constant-factor approximation algorithm for the balanced k-median problem in a σ-SHST. The running time of this algorithm was $\mathcal{O}(n^3\, k^2) \cdot$

$n^{\mathcal{O}(\log\log n/\log\sigma)}$, where n the number of input points. By selecting $\sigma = O(\log^{\epsilon} n)$, the running-time of this algorithm is polynomial, and leads to an approximation guarantee for the balanced k-median in arbitrary metric to within a factor of $\mathcal{O}(\frac{1}{\epsilon}\log^{1+\epsilon} n/\log\log n)$. The bottleneck of the algorithm for balanced k-media described above is the number of points to be processed. Therefore, to achieves the running-time of $o(n^2)$ we had to reduce the number of points in the metric space — and this is why we introduced the splittable weighted balanced k-median problem, in which the number of points to be processed in the reduction is $\mathcal{O}(N)$ only, instead of $\mathcal{O}(n)$. (Notice that with our notation $n = W$.)

9.1.1 Basic dynamic programming for splittable weighted balanced k-median in SHSTs

In this section we describe a basic dynamic programming scheme that models the structure of a constant approximation for weighted splittable balanced k-median in SHSTs, following closely the algorithm due to Bartal et al. [8].

We first observe that we can assume that all the weights are in the range $[\Delta/W^{\mathcal{O}(1)}, \Delta]$, where Δ is the diameter of the metrics. (For if not, we could argue than any edge of length smaller than $\Delta/W^{\mathcal{O}(1)}$ can be neglected, because its cost and its contribution are negligible in comparison to the cost of the solution, see [8] for more discussion.) With this, we only have to consider σ-SHSTs with $\mathcal{O}(\log_{\sigma} W)$ levels.

Consider a σ-SHST. In order to efficiently run the dynamic programming we transform the tree into a binary tree. We do this by replacing a node at level ℓ with t children at level $\ell+1$ by a $\log t$ level binary tree, introducing dummy nodes in the tree (and the cost of all edges between the nodes at level $\ell+1$ will be zero). (We will still maintain the *levels* from the original SHST, not the binary SHST.) However, unlike in [8], we do not require that every leaf corresponds to a single input point; indeed, in our case every leaf corresponds to its multiplicity in the problem instance, and additionally, many points can be glued together during the reduction that ensures that all weights are in the range $[\Delta/W^{\mathcal{O}(1)}, \Delta]$. We use the notion of a *leaf-point* to denote any copy of a leaf in SHST that corresponds to multiple input point.

The dynamic programming will process the binary SHST tree bottom-up. For every node in the tree, in a bottom-up fashion we will compute all "optimal" solutions for the subtree rooted at that node.

Before we proceed we need yet another notion. We say a *leaf-point* u *connects to a center* in the subtree T_v rooted at v *from level* i, if u is located outside T_v and i is the highest level (with the smallest index) in the path from u to v. Notice that in that case the distance from u to any leaf in T_v is exactly Δ/σ^i.

Dynamic programming table. Let us take any node u in T and let ℓ be its level. We will set up a dynamic programming table $D(u, r, s, n_0, n_1, \dots, n_\ell)$ such that $D(u, r, s, n_0, n_1, \dots, n_\ell)$ is the value of the optimal solution for the subtree T_u rooted at u such that

- exactly r leaf-points are chosen as centers among the leaf-points in T_u,
- at most s leaf-points in T_u are in the clusters with the centers outside T_u and are not accounted for in the cost, and
- for $0 \le i \le \ell$, there are n_i leaf-points outside T_u that connect to a center in T_u from level i.

Basic case: cost at the leaves. First, let us notice that if u is a leaf with multiplicity $w(u)$ (that is, u corresponds to $w(u)$ leaf-points), then $D(u, r, s, n_0, n_1, \dots, n_\ell)$ can be computed as follows:

- if $r > \min\{k, w(u)\}$ then $D(u, r, s, n_0, n_1, \dots, n_\ell)$ is undefined;

- if $r = 0$, then
  - $D(u, r, s, n_0, n_1, \dots, n_\ell) = 0$ if $s \geq w(u)$ and all $n_i = 0$, and
  - $D(u, r, s, n_0, n_1, \dots, n_\ell)$ is undefined otherwise;
- if $1 \leq r \leq w(u)$, then $D(u, r, s, n_0, n_1, \dots, n_\ell)$ is taking the minimum value of the form

$$\sum_{\gamma=1}^{r} \left(\left((1 + \kappa_\gamma + \sum_{j=0}^{\ell} \lambda_{\gamma,j} \right) \cdot \sum_{j=0}^{\ell} \frac{\lambda_{\gamma,j} \cdot \Delta}{\sigma^j} \right) \quad ,$$

  over all settings of non-negative integers κ_γ with $\sum_{\gamma=1}^{r} \kappa_\gamma = \max\{w(u) - r - s, 0\}$, and all non-negative integers $\lambda_{\gamma,j}$ with $\sum_{\gamma=1}^{r} \lambda_{\gamma,j} = n_j$ for every $0 \leq j \leq \ell$.

  The expression $\sum_{\gamma=1}^{r} \left(1 + \kappa_\gamma + \sum_{j=0}^{\ell} \lambda_{\gamma,j}\right) \left(\sum_{j=0}^{\ell} \lambda_{\gamma,j} \cdot \Delta/\sigma^j\right)$ corresponds to the situation when we have r centers $c_1, \dots, c_r$ among the leaf-points at u, and for each center c_γ there are κ_γ other leaf-points at u that are at the cluster of c_γ, and for each j, there are exactly $\lambda_{\gamma,j}$ leaf-points outside T_u that connect to c_γ from level j. In that case, center c_γ has cluster of size $1 + \kappa_\gamma + \sum_{j=0}^{\ell} \lambda_{\gamma,j}$ and the sum of the distance of the points in this cluster to c_γ is equal to $\sum_{j=0}^{\ell} \lambda_{\gamma,j} \cdot \Delta/\sigma^j$; hence, the cost of this cluster is $\left(1 + \kappa_\gamma + \sum_{j=0}^{\ell} \lambda_{\gamma,j}\right) \left(\sum_{j=0}^{\ell} \lambda_{\gamma,j} \cdot \Delta/\sigma^j\right)$.

  The definition of s ensures that in order to minimize $D(u, r, s, n_0, n_1, \dots, n_\ell)$, exactly $\min\{w(u) - r, s\}$ leaf-points at u will be assigned to the centers outside T_u, and hence $\sum_{\gamma=1}^{r} \kappa_\gamma = \max\{w(u) - r - s, 0\}$ points will be associated with the r centers in T_u.

General case: cost at the internal nodes. Next, we claim that the following formula computes the value $D(u, r, s, n_0, n_1, \dots, n_\ell)$ for an internal node u with two children v_1 and v_2 at level ℓ' (notice that either $\ell = \ell'$, in which case u, v_1, v_2 have been siblings in the original (non-binary) SHST, or $\ell = \ell' - 1$):

$$D(u, r, s, n_0, \dots, n_\ell) \quad = \quad \min \left\{ D(v_1, r_1, s_1, n_0^{(1)}, \dots, n_{\ell'}^{(1)}) + D(v_2, r_2, s_2, n_0^{(2)}, \dots, n_{\ell'}^{(2)}) \right\} \quad ,$$

where we minimize over all settings for which these expressions make sense and lead to a feasible clustering. More formally, we minimize over all settings of the values such that:

(1) $r = r_1 + r_2$,

(2) r_1 is not greater than the multiplicity of the leaf-points in T_{v_1} and r_2 is not greater than the multiplicity of the leaf-points in T_{v_2},

(3) $s_1 = s_{1,2} + s_{1,\text{out}}$, $s_2 = s_{2,1} + s_{2,\text{out}}$, $s = s_{1,\text{out}} + s_{2,\text{out}}$, $n_{\ell'}^{(1)} \geq s_{2,1}$, and $n_{\ell'}^{(2)} \geq s_{1,2}$, where $s_{1,2}$ and $s_{2,1}$ correspond to the number of leaf-points in T_{v_1} and T_{v_2}, respectively, supported by centers in T_{v_2} and T_{v_1}, respectively, and $s_{1,\text{out}}$ and $s_{2,\text{out}}$ correspond to the number of leaf-points in T_{v_1} and T_{v_2}, respectively, supported by centers outside T_u,

(4) for every $i < \ell'$, $n_i = n_i^{(1)} + n_i^{(2)}$,

(5) if $\ell = \ell'$ then $n_\ell = n_\ell^{(1)} - s_{2,1} + n_\ell^{(2)} - s_{1,2}$,

(6) if $\ell = \ell' - 1$ then $n_{\ell'}^{(1)} = n_{\ell'}^{(2)} = 0$.

Identical arguments as those used in [8] can be applied to argue that $D(u, r, s, n_0, n_1, \ldots, n_\ell)$ is defined properly.

If we apply the dynamic programming scheme bottom-up then we could correctly compute the value of the optimal solution for weighted splittable balanced k-median in the SHST, and from that we could easily obtain the clustering and the centers itself. However, this scheme would have superpolynomial running time. Indeed, for every node u we have to compute $D(u, r, s, n_0, n_1, \ldots, n_\ell)$ for all integers $r, s, n_0, n_1, \ldots, n_\ell$ with $0 \le r \le k$ and $0 \le s, n_0, n_1, \ldots, n_\ell \le W$. Therefore, with each node u we will have to compute $W^{\mathcal{O}(\log_\sigma W)}$ values. To compute each entry $D(u, r, s, n_0, n_1, \ldots, n_\ell)$ we will need $W^{\mathcal{O}(\log_\sigma W)}$ time. Therefore, the entire dynamic programming can be solved in time $W^{\mathcal{O}(\log_\sigma W)}$, which is superpolynomial in W. In the next section we will discuss how to significantly improve the running time. Our approach is based on a rather simple modification of the scheme from [8].

9.1.2 Efficient dynamic programming for splittable weighted balanced k-median in SHSTs

Bartal et al. [8] improved the running time of the dynamic programming from Section 9.1.1 (for balanced k-median) by observing that by allowing an *approximation* of the optimal value, we can reduce the number of values computed by the dynamic programming. Let $\alpha = 1 + \frac{1}{\log^2 W}$. We call an integer *valid* if it is in the set $\{\lceil \alpha^i \rceil : 0 \le i \le B\} \cup \{0\}$, where $B = \mathcal{O}(\log^3 W)$ is such that $\alpha^B \ge W$.

Now, we modify the dynamic programming and compute the values of the form $PP^*(u, r, s, n_0, \ldots, n_\ell)$, where each n_i is a valid integer and (additionally to the scheme from [8]) the value s is valid as well. In other words, we assume that the upper bound s for the number of leaf-points in T_u that are in the clusters with the centers outside T_u and are not accounted for in the cost is of the form α^j, and so are the numbers n_i of leaf-points outside T_u that connect to a center in T_u from level i. Our goal is to use $PP^*(u, r, s, n_0, \ldots, n_\ell)$ to estimate the value of $D(u, r, \overline{s}, \overline{n_0}, \overline{n_1}, \ldots, \overline{n_\ell})$ for all values of $\overline{s}$ and $\overline{n_i}$ such that $s \le \overline{s} \le \alpha s$ and $n_i \le \overline{n_i} \le \alpha n_i$. To achieve this, we redefine the compututation of $PP^*(u, r, s, n_0, n_1, \ldots, n_\ell)$ as follows:

$$PP^*(u, r, s, n_0, \ldots, n_\ell) = \min \left\{ PP^*(v_1, r_1, s_1, n_0^{(1)}, \ldots, n_{\ell'}^{(1)}) + PP^*(v_2, r_2, s_2, n_0^{(2)}, \ldots, n_{\ell'}^{(2)}) \right\} \quad ,$$

where we minimize over all settings of the values such that:

(1) $r = r_1 + r_2$,

(2) r_1 is not greater than the multiplicity of the leaves in T_{v_1} and r_2 is not greater than the multiplicity of the leaves in T_{v_2},

(3) $s_1 = s_{1,2} + s_{1,\text{out}}$, $s_2 = s_{2,1} + s_{2,\text{out}}$, $\frac{1}{\alpha}(s_{1,\text{out}} + s_{2,\text{out}}) \le s \le s_{1,\text{out}} + s_{2,\text{out}}$, $n_{\ell'}^{(1)} \ge s_{2,1}$, and $n_{\ell'}^{(2)} \ge s_{1,2}$,

(4) for every $i < \ell$, $\frac{1}{\alpha}(n_\ell^{(1)} + n_\ell^{(2)}) \le n_i \le n_i^{(1)} + n_i^{(2)}$,

(5) if $\ell = \ell'$ then $\frac{1}{\alpha}(n_\ell^{(1)} + n_\ell^{(2)}) \le n_\ell + s_{2,1} + s_{1,2} \le n_\ell^{(1)} + n_\ell^{(2)}$,

(6) if $\ell = \ell' - 1$ then $n_{\ell'}^{(1)} = n_{\ell'}^{(2)} = 0$,

(7) values s_1, s_2, and all values $n_i^{(1)}$ and $n_i^{(2)}$ are valid.

With such a definition of dynamic programming, one can repeat the analysis from [8, Lemmas 4–7] to prove that for every node u and parameters $r, s, n_0, n_1, \ldots, n_\ell$, there exists $\widehat{s}, \widehat{n_0}, \widehat{n_1}, \ldots, \widehat{n_\ell}$ that are valid integers such that

$$\mathrm{PP}^*(u, r, \widehat{s}, \widehat{n_0}, \widehat{n_1}, \ldots, \widehat{n_\ell}) \quad \leq \quad \alpha^{\mathcal{O}(h(u))} \cdot D(u, r, s, n_0, n_1, \ldots, n_\ell) \ , \tag{2}$$

where $h(u)$ denotes the height of node u in the binary tree used for dynamic programming (that is, $h(u)$ is the longest distance from u to a leaf descendant in the binary tree).

Once we have set up this dynamic programming, the rest is easy. By inequality (2), the value obtained by the dynamic programming is at most a factor $\alpha^{\mathcal{O}(\text{height}(T))}$ away from the optimal value, where $\text{height}(T)$ denotes the height of the binary form of T. Since the height of T is $\mathcal{O}(\log^2 W)$ and $\alpha = 1 + \frac{1}{\log^2 W}$, we have $\alpha^{\mathcal{O}(\text{height}(T))} = \mathcal{O}(1)$, and hence the solution obtained is a constant approximation for balanced k-median in σ-SHST. By our discussion at the beginning of Section 9, this yields an $\mathcal{O}(\sigma \cdot \log N / \log \sigma)$-approximation algorithm for balanced k-median for arbitrary metric.

Next, let us analyze the running time of this algorithm. First, we observe that by [17], the reduction to the balanced k-median problem in σ-SHSTs requires $\mathcal{O}(N^2)$ time. Next, to solve the problem in σ-SHSTs, we have to give the running time for the dynamic programming procedure. The number of values to be computed in dynamic programming at any given node in the tree is $\mathcal{O}(k) \cdot (O(\log^3 W))^{\mathcal{O}(\text{max-level in } T)}$, where max-level in T is the maximum level in the original σ-SHST, which is $\mathcal{O}(\log_\sigma W)$. Therefore the computation of each value performed bottom-up will take time $\left(\mathcal{O}(k) \cdot (O(\log^3 W))^{\mathcal{O}(\text{max-level in } T)}\right)^2$. Since we have $\mathcal{O}(N)$ nodes to be considered, the total running time of this algorithm will be

$$\begin{aligned}\mathcal{O}(N) \cdot \left(O(k) \cdot (O(\log^3 W))^{\mathcal{O}(\text{max-level in } T)}\right)^2 \quad &\leq \quad \mathcal{O}(N \cdot k^2) \cdot (O(\log^3 W))^{\mathcal{O}(\log_\sigma W)} \\ &= \quad \mathcal{O}(N \cdot k^2) \cdot W^{\mathcal{O}(\log\log W / \log \sigma)} \quad .\end{aligned}$$

By setting $\sigma = \Theta(\log^\lambda W)$ we can obtain the total running time to be $\mathcal{O}(N^2 + N \cdot k^2 \cdot W^{1/\lambda})$, in which case the approximation guarantee is $\mathcal{O}(\frac{1}{\lambda} \cdot \log N \cdot \log^{\mathcal{O}(\lambda)} W)$. This yields Theorem 24.

Acknowledgements

We would like to thank Piotr Indyk for his contributions in the early stages of the work.

References

[1] I. Abraham, Y. Bartal, and O. Neiman. Advances in metric embedding theory. *Proc. 38th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 271–286, 2006

[2] N. Alon, S. Dar, M. Parnas, and D. Ron. Testing of clustering. *SIAM Journal on Discrete Mathematics*, 16(3): 393–417, 2003.

[3] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximation via coresets. In *Current Trends in Combinatorial and Computational Geometry*, edited by E. Welzl, Cambridge University Press, New York, 2005.

[4] M. Bădoiu, A. Czumaj, P. Indyk, and C. Sohler. Facility location in sublinear time. *Proc. 32nd Annual International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 866–877, 2005.

[5] M. Bădoiu, S. Har-Peled, and P. Indyk. Approximate clustering via core-sets. *Proc. 34th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 250–257, 2002.

[6] Y. Bartal Probabilistic approximation of metric spaces and its algorithmic applications. *Proc. 37th IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 184–193, 1996.

[7] Y. Bartal On approximating arbitrary metrics by tree metrics. *Proc. 30th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 161–168, 1998.

[8] Y. Bartal, M. Charikar, and D. Raz. Approximating min-sum k-clustering in metric spaces. *Proc. 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 11–20, 2001.

[9] P. Berkhin. Survey of clustering data mining techniques. Technical Report, Accrue Software, San Jose, CA, 2002.

[10] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. *Proc. 29th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 626–635, 1997.

[11] M. Charikar, L. O'Callaghan, and R. Panigrahy. Better streaming algorithms for clustering problems. *Proc. 35th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 30–39, 2003.

[12] K. Chen. On k-median clustering in high dimensions. *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1177–1185, 2006.

[13] A. Czumaj and C. Sohler. Abstract combinatorial programs and efficient property testers. *SIAM Journal on Computing*, 34(3):580–615, 2005.

[14] A. Czumaj and C. Sohler. Sublinear-time approximation for clustering via random sampling. *Random Structures and Algorithms*, 30(1–2): 226–256, January – March 2007.

[15] W. Fernandez de la Vega, M. Karpinski, C. Kenyon, and Y. Rabani. Approximation schemes for clustering problems. *Proc. 35th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 50–58, 2003.

[16] W. Fernandez de la Vega and C. Kenyon. A randomized approximation scheme for metric MAX-CUT. *Journal of Computer and System Sciences*, 63(4):531–541, 2001.

[17] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.

[18] D. Fotakis. Memoryless facility location in one pass. *Proc. 23rd Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pp. 608–620, 2006.

[19] G. Frahling and C. Sohler. Coresets in dynamic geometric data streams. *Proc. 37th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 209–217, 2005.

[20] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for Maximum Cut and satisfiability problems using semidefinite programming *Journal of the ACM*, 42:1115–1145, 1995.

[21] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.

[22] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. *Proc. 41st IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 359–366, 2000.

[23] N. Gutmann-Beck and R. Hassin. Approximation algorithms for min-sum p-clustering. *Discrete Applied Mathematics*, 89:125–142, 1998.

[24] S. Har-Peled. Clustering motion. *Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 84–93, 2001.

[25] S. Har-Peled and S. Mazumdar. Coresets for k-means and k-medians and their applications. *Proc. 36th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 291–300, 2004.

[26] S. Har-Peled and A. Kushal. Smaller coresets for k-median and k-means clustering. *Proc. 21st Annual ACM Symposium on Computational Geometry (SoCG)*, pp. 126–134, 2005.

[27] S. Har-Peled and K. Varadarajan. Projective clustering in high dimensions using core-sets. *Proc. 18th Annual ACM Symposium on Computational Geometry (SoCG)*, pp. 312–318, 2002.

[28] P. Indyk. Sublinear time algorithms for metric space problems. *Proc. 31st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 428–434, 1999.

[29] P. Indyk. A sublinear time approximation scheme for clustering in metric spaces. *Proc. 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 154–159, 1999.

[30] P. Indyk. *High-Dimensional Computational Geometry*. PhD thesis, Stanford University, 2000.

[31] P. Indyk. Algorithms for dynamic geometric problems over data streams. *Proc. 36th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 373–380, 2004.

[32] P. Indyk and J. Matoušek. Low-distortion embeddings of finite metric spaces. In *Handbook of Discrete and Computational Geometry*, 2nd edition, edited by J. E. Goodman and J. O'Rourke, CRC Press, 2004, pp. 177–196.

[33] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, September 2003.

[34] F. Korn, S. Muthukrishnan, D. Srivastava. Reverse nearest neighbor aggregates over data streams. *Proc. 28th International Conference on Very Large Data Bases (VLDB)*, pp. 814–825, 2002.

[35] A. Kumar, Y. Sabharwal, and S. Sen. A simple linear time $(1 + \varepsilon)$-approximation algorithm for k-means clustering in any dimensions. *Proc. 45th IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 454–462, 2004.

[36] A. Kumar, Y. Sabharwal, and S. Sen. Linear time algorithms for clustering problems in any dimensions. *Proc. 32nd Annual International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 1374–1385, 2005.

[37] R. Mettu and G. Plaxton. Optimal time bounds for approximate clustering. *Machine Learning*, 56(1-3):35–60, 2004.

[38] A. Meyerson. Online facility location. *Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 426–431, 2001.

[39] A. Meyerson, L. O'Callaghan, and S. Plotkin. A k-median algorithm with running time independent of data size. *Machine Learning*, 56(1–3): 61–87, July 2004.

[40] N. Mishra, D. Oblinger, and L. Pitt. Sublinear time approximate clustering. *Proc. 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 439–447, 2001.

[41] S. Muthukrishnan. Data streams: Algorithms and applications. In *Foundations and Trends in Theoretical Computer Science*, volume 1, issue 2, August 2005.

[42] M. Parnas, D. Ron, R. Rubinfeld. Tolerant property testing and distance approximation. *Electronic Colloquium on Computational Complexity (ECCC)*, Report No. 10, 2004.

[43] Y. Rabinovich. On average distortion of embedding metrics into the line and into L_1. *Proc. 35th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 456–462, 2003.

[44] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23:555–566, 1976.

[45] L. J. Schulman. Clustering for edge-cost minimization. *Proc. 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 547–555, 2000.

[46] M. Thorup. Quick k-median, k-center, and facility location for sparse graphs. *SIAM Journal on Computing*, 34(2):405–432, 2005.

[47] T. Tokuyama and J. Nakano. Geometric algorithms for the minimum cost assignment problem. *Random Structures and Algorithms*, 6(4):393–406, 1995.

[48] R. Xu and D. Wunsch II. Survey of clustering algorithms. *IEEE Transactions in Neural Networks*, 16(3): 645–678, May 2005.