Massively Parallel and Streaming Algorithms for Balanced Clustering*

Kian Mirjalali† Hamid Zarrabi-Zadeh‡

Abstract

Clustering is a fundamental problem with a wide range of applications. In this paper, we study a balanced version of the well-known k-center clustering problem, where the objective is to select k centers from a given set of points, and assign to each center at most L of the input points, so as to minimize the maximum distance from a point to the center to which it is assigned. We assume soft capacities, where points are allowed to be selected as center more than once. Motivated by applications involving massive datasets, we study the problem in the massively parallel computation (MPC) and streaming models. In particular, we present a two-round MPC algorithm for the balanced k-center problem, achieving an approximation factor of $5\alpha + 2$, where α is the approximation factor of the corresponding sequential algorithm. This substantially improves the currently best approximation factor of 32α, available for the problem. We show that the approximation factor of our algorithm can be further improved to $5 + \varepsilon$ in all spaces with bounded doubling dimension, including the Euclidean space. We also consider the balanced k-center problem in the streaming model, and present a constant-factor streaming algorithm in any metric space using $O(kn^{\varepsilon})$ memory, and a factor $5 + \varepsilon$ streaming algorithm using $O(k \operatorname{polylog} n)$ memory in doubling metrics.

Keywords Clustering; Massively parallel computation; Streaming model; Coresets; Capacitated clustering.

1 Introduction

Clustering is a common problem in various areas such as machine learning, data mining, and statistics. The *k-center* problem is one of the most basic and widely-used formulations of clustering, in which the objective is to select k centers from a set of n points, such that the maximum distance between the points and their nearest centers is minimized.

In the *balanced k-center* problem (also known as *capacitated k-center*), each center has a fixed capacity L, bounding the number of points that can be covered by that center. Clearly, when centers have capacity, each point is not necessarily covered by its nearest center. The *weighted balanced k-center* is a generalization of this problem where each point x in the input has a weight/demand $w(x)$. When a point x is assigned to a center, it uses $w(x)$ units of the center's capacity. Moreover, a point can be assigned to more than one center, each covering a portion of its demand. The problem in its general form has natural applications in facility location scenarios where resources have capacities, and users have specific demands.

*A preliminary version of this work was presented at TTCS 2020 [21].

†Department of Computer Engineering, Sharif University of Technology. Email: mirjalali@ce.sharif.edu

‡Department of Computer Engineering, Sharif University of Technology. Email: zarrabi@sharif.edu

Clustering is more challenging when we are dealing with large datasets. Several computational models have been developed over the past years to cope with large data. In particular, the massively parallel computation (MPC) model has become more prominent as a popular abstraction of modern parallel frameworks such as MapReduce, Hadoop, and Spark [5, 16]. In this model, the input data is initially partitioned among a set of machines, each having a local memory, such that the number of machines and the size of local memory for each machine is strongly sublinear in the size of the input. An algorithm in this model runs in a number of synchronous rounds. At each round, machines can perform arbitrary polynomial-time computations on their local memory, without any communication with other machines. After each round, machines are allowed to communicate with each other, provided that the total size of messages sent and received by each machine in that round does not exceed its local memory. The output of the algorithm is eventually computed and stored in one or more machines after several rounds. It is generally desired to keep the number of rounds used by an MPC algorithm as small as possible, ideally a constant.

Another popular model for handling massive data is the streaming model [22]. Here, input data is given to the algorithm one by one, as a stream of points, and the algorithm (running on a single machine) has a limited storage, much less than the size of the whole input. In general, the space available to the algorithm in this model is sublinear in the input size, and the algorithm has to work with one or few passes over the input.

Related work. There are well-known approximation algorithms for the k-center problem. The furthest-point greedy algorithm proposed by Gonzalez [12] yields a 2-approximate solution in any metric space. Hochbaum and Shmoys [14] proposed another 2-approximation algorithm based on the parametric pruning. It is known that no $2 - \varepsilon$ approximation algorithms is possible for the k-center problem, unless P = NP [14].

The capacitated version of the k-center problem, where each center has a fixed capacity L, was first studied by Bar-Ilan et al. [2]. They proposed a 10-approximation algorithm for the problem. The approximation factor was later improved by Khuller and Sussmann [17] to 6 in general case, and to 5 for soft capacities, where elements can be selected as center more than once. Cygan et al. [10] studied a non-uniform variant of the problem, where each element has its own specific capacity if selected as center. They presented a constant-factor approximation algorithm for this problem, whose factor was later improved to 9 by An et al. [1].

Several algorithms have been proposed over the past decade for the k-center problem and its variants in the MPC model. Ene et al. [11] were the first to present a constant-round algorithm for the k-center problem achieving an approximation factor of 10. The approximation factor was later improved to 4 by Malkomes et al. [18]. For metric spaces with bounded doubling dimension, Ceccarello et al. [8] improved the approximation factor to $2 + \varepsilon$, for any constant $\varepsilon > 0$. For the balanced k-center problem, Bateni et al. [3] presented a randomized MPC algorithm with an approximation factor of 32α, where α is the approximation factor of the corresponding sequential algorithm for the balanced k-center problem. The algorithms of [3, 8, 18] are all based on the *composable coreset* framework [15], in which a small subset of data, so-called a coreset, is extracted from each machine in such a way that the union of coresets contains a good approximation of the whole data set. Some other variants of the k-center clustering are studied in the MPC model as well (see, e.g., [4, 7, 8, 19]).

In the single-pass streaming model, Charikar et al. [9] proposed a streaming algorithm for the k-center problem, achieving an approximation factor of 8 in any metric space. Guha [13] and McCutchen and Khuller [20] improved the approximation factor to $2 + \varepsilon$, which is essentially the best possible. McCutchen and Khuller [20] also considered k-center with outliers in data streams.

Several other clustering problems are also studied in the streaming model (see for example the surveys in [23–25]).

Our results. In this paper, we study the weighted balanced k-center problem with soft capacities in the MPC and streaming models, and present several new results as summarized below.

- We provide a massively parallel algorithm for the weighted balanced k-center problem with an approximation guarantee of $5\alpha+2$, where α is the approximation factor of the corresponding sequential algorithm for the weighted balanced k-center problem. Our algorithm substantially improves the currently best approximation factor of 32α, due to Bateni et al. [3]. Our algorithm uses only two rounds of MPC and requires $O(n/m+mk)$ space per machine, where m is the number of machines, and n is the total input size. The average communication per machine is $O(k)$, which we show is optimal under a fair assumption.
- For metric spaces with bounded doubling dimension, including all $\mathbb{R}^d$ spaces under any ℓ_p metric, we show that the approximation factor of our algorithm can be further improved to $5+\varepsilon$ in two rounds of MPC. The core idea behind our improved algorithm is to enhance the precision of the coresets extracted in the first round, by exploiting the properties of doubling metrics, similar to the techniques used in [8]. To analyze our improved algorithm, we introduce a new notion of "covering dimension" of a metric space, and show how it is related to the doubling dimension of the corresponding space.
- We present one-pass streaming algorithms for the weighted balanced k-center problem. In particular, we present a constant-factor approximation algorithm in the streaming model using $O(kn^{\varepsilon})$ space, for any positive constant $\varepsilon < 1$. For metric spaces with bounded doubling dimension, we show that a factor $5+\varepsilon$ approximation can be achieved using only $O(k \operatorname{polylog} n)$ space. We also show that in general metric spaces, the memory size can be improved to $O(k \log n)$ at the expense of increasing the approximation guarantee by a factor of $O(\log n)$.

2 Preliminaries

Let (U, d) be a metric space. For a point $p \in U$ and a set $Q \subseteq U$, we denote by $d(p, Q)$ the distance of p to the set Q, i.e., $d(p, Q) = \min_{q \in Q} d(p, q)$. We call a set $P \subseteq U$ along with a function $w(\cdot)$ that assigns to each point $p \in P$ a non-negative weight $w(p)$ a *weighted point set*.

2.1 Balanced Clustering

Given a weighted point set S, and two positive integers k and L, a *balanced k-clustering* of S (with capacity L) consists of a set $C \subseteq S$ of k *centers*, along with an assignment function $\phi : S \times C \to \mathbb{N}_0$ that satisfies the following two conditions:

$$\forall c \in C : \sum_{p \in S} \phi(p, c) \leq L,$$

and

$$\forall p \in S : \sum_{c \in C} \phi(p, c) = w(p).$$

The former is called the *capacity constraint*, and the latter is called the *demand constraint*. We say that a point p is *contained* in a cluster centered at c (or equivalently, p is *assigned* to center c), if $\phi(p,c) > 0$. Note that in a balanced k-clustering, a point of weight greater than one may be assigned to more than one center. The cost of a balanced k-clustering (C, ϕ) of a weighted point set S, denoted by $\text{cost}_S(C)$, is defined as the maximum distance of any point in S to a center it is assigned to, i.e.,

$$\text{cost}_S(C) := \max_{\substack{p \in S, c \in C, \\ \phi(p,c)>0}} d(p,c).$$

In the *weighted balanced k-center problem*, the objective is to find a balanced k-clustering (C, ϕ) of a weighted point S that minimizes $\text{cost}_S(C)$. The cost of such an optimal clustering is denoted by $\text{OPT}(S)$, or OPT for short. When the input set S is unweighted, i.e., $w(p) = 1$ for all $p \in S$, we refer to the problem simply as the *balanced k-center problem*. Note than in this case, $\phi(p,c) \in \{0,1\}$ for all $p, c \in S$, and each point of S is assigned to exactly one center.

Throughout this paper, we consider a *soft version* of the problem, where a point can be selected as a center more than once [10,17]. There is also a *hard* version of the problem, in which each point can be selected as a center at most once. Note that any solution to the (soft) balanced k-center problem provides a 2-approximation to the hard version as follows: Simply replace any copy c of a center which is selected more than once by a distinct point c' in the cluster of c, and reassign all points previously assigned to c to the new center c'. This way, the distance of each point to its assigned center is at most doubled, while the capacity constraints are still satisfied. As such, any α-approximation algorithm presented in this paper for the (soft) balanced k-center problem automatically yields a 2α-approximation algorithm for the hard version as well.

2.2 Coresets

Let P be a point set, and δ be a positive number. A subset $Q \subseteq P$ is called a δ-coreset of P (with respect to balanced k-clustering), if for any point $p \in P$, there is a point $q \in Q$ such that

$$d(p,q) \leq \delta \cdot \text{OPT}(P).$$

A coreset indeed represents the original set by a smaller subset, while distances are approximately preserved. More precisely, each point $p \in P$ *is represented* by a point $f(p)$ in the coreset Q within distance at most $\delta \cdot \text{OPT}(P)$. In this paper, we use a stronger notion of coreset in which a subset $Q \subseteq P$ is called a *δ-coreset* of P, if for any point $p \in P$, there is a point $q \in Q$ such that

$$\forall S \supseteq P, \;\; d(p,q) \leq \delta \cdot \text{OPT}(S).$$

Note that by our definition, δ-coresets are "*composable*" in a sense that if Q_i is a δ-coreset of P_i, for $1 \leq i \leq m$, then $Q = \bigcup_{i=1}^m Q_i$ is a δ-coreset of $P = \bigcup_{i=1}^m P_i$.

If a point set P is weighted, any δ-coreset Q of P is also weighted, where the weight of each point $q \in Q$ is defined as

$$w_Q(q) = \sum_{p: f(p)=q} w(p). \tag{1}$$

2.3 GMM Algorithm

There is a simple greedy algorithm, called GMM, that computes a 2-approximation to the k-center problem in any metric space [12]. The algorithm repeatedly picks as center a point furthest away from the centers already chosen. The pseudo-code of GMM is provided in Algorithm 1. We will use this greedy algorithm as a subroutine in our MPC and streaming algorithms in the next sections.

Algorithm 1 GMM(S, k)

$T \leftarrow \{$an arbitrary point $p \in S\}$
while $|T| < k$ **do**
  $q \leftarrow \arg\max_{p \in S} d(p, T)$
  $T \leftarrow T \cup \{q\}$
return T

3 MPC Algorithm for Balanced Clustering

In this section, we present an approximation algorithm for the weighted balanced k-center problem in the MPC model. The algorithm receives as input a weighted point set S, partitioned into m subsets $S_1, \ldots, S_m$, stored in m local machines. Each local machine extracts a δ-coreset $T_i \subseteq S_i$, and sends T_i to a central machine. The central machine composes the m received coresets into a single set T, and applies a sequential algorithm on T to obtain a clustering containing k centers. The clustering is then sent back to the local machines to compute the final assignment. The algorithm is summarized in pseudo-code in Algorithm 2.

Algorithm 2 MPC Balanced Clustering

Input: Weighted point sets $S_1, \ldots, S_m$, an integer k, and a capacity L
Output: A balanced k-clustering of the set $S = \bigcup_{i=1}^{m} S_i$

For each $1 \leq i \leq m$, compute a δ-coreset T_i of S_i.
Send weighted sets T_i to a central machine.
Let $T = \bigcup_{i=1}^{m} T_i$.
Compute an α-approx balanced k-center (C, ϕ) of T.
Send (C, ϕ) back to local machines to obtain the final assignment.

Theorem 1. *The approximation factor of Algorithm 2 is* $(2\delta + 1)\alpha + \delta$.

Proof. Let S be the input point set, and T be the union of δ-coresets computed by the algorithm. The outline of our proof is as follows. We first show that there exists a balanced k-clustering (C', ϕ') of T of cost at most $(2\delta + 1) \cdot \mathrm{OPT}$, and hence, the α-approximate clustering (C, ϕ) of T computed by the algorithm has cost at most $\alpha(2\delta + 1) \cdot \mathrm{OPT}$. We then show that the clustering (C, ϕ) of T can be transformed to a clustering of S of cost at most $(\alpha(2\delta + 1) + \delta) \cdot \mathrm{OPT}$.

Note that by the composability of coresets, T is a δ-coreset of S. As such, for each point $p \in S$, there is a point $f(p) \in T$, representing p, such that $d(p, f(p)) \leq \delta \cdot \mathrm{OPT}$. For each point $q \in T$, we denote by $F(q)$ the set of points in S represented by q, i.e., $F(q) = \{p \in S \mid f(p) = q\}$. Let (C^*, ϕ^*) be an optimal balanced k-clustering of S with cost OPT. We build a balanced k-clustering (C', ϕ') of T with cost at most $(2\delta + 1) \cdot \mathrm{OPT}$ as follows. Let $\{c_1^*, \ldots, c_k^*\}$ be the k centers in C^*. We set $C' = \{c_1', \ldots, c_k'\}$, where $c_i' = f(c_i^*)$ for $1 \leq i \leq k$, and define

$$\phi'(q, c_i') := \sum_{p \in F(q)} \phi^*(p, c_i^*) \qquad (\text{for all } q \in T,\ 1 \leq i \leq k).$$

To see that (C', ϕ') is a feasible balanced k-clustering of T, we need to verify that the following two conditions hold:

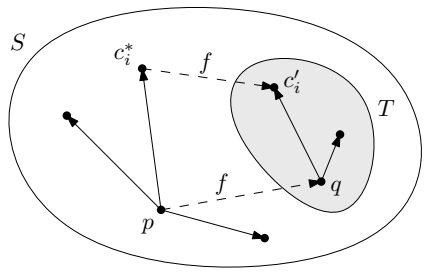

Figure 1: The distance of a point $q \in T$ to a center to which it is assigned.

- (Demand constraint) for each point $q \in T$, we have:

$$\begin{aligned}
\sum_{1\le i\le k} \phi'(q, c_i') &= \sum_{1\le i\le k} \sum_{p\in F(q)} \phi^*(p, c_i^*) && \text{(by definition of } \phi'\text{)} \\
&= \sum_{p\in F(q)} \sum_{1\le i\le k} \phi^*(p, c_i^*) \\
&= \sum_{p\in F(q)} w(p) && \text{(by } \phi^* \text{ demand)} \\
&= w_T(q) && \text{(by equality (1))}
\end{aligned}$$

- (Capacity constraint) for each center c_i' $(1 \le i \le k)$, we have:

$$\begin{aligned}
\sum_{q\in T} \phi'(q, c_i') &= \sum_{q\in T} \sum_{p\in F(q)} \phi^*(p, c_i^*) && \text{(by definition of } \phi'\text{)} \\
&= \sum_{p\in S} \phi^*(p, c_i^*) \\
&\le L && \text{(by } \phi^* \text{ capacity)}
\end{aligned}$$

Now, fix a point $q \in T$, and let c_i' be a center in C' for which $\phi'(q, c_i') > 0$. By definition of ϕ', there exists a point $p \in F(q)$ such that $\phi^*(p, c_i^*) > 0$. (See Figure 1.) Now, by triangle inequality, we have

$$d(q, c_i') \;\le\; d(q, p) + d(p, c_i^*) + d(c_i^*, c_i').$$

Both distances $d(q, p)$ and $d(c_i^*, c_i')$ are at most $\delta \cdot \mathrm{OPT}$. Moreover, $d(p, c_i^*) \le \mathrm{OPT}$, as p is assigned to c_i^* in an optimal clustering. Therefore, $d(q, c_i') \le (2\delta + 1) \cdot \mathrm{OPT}$. As this inequality holds for all $q \in T$, we have

$$\mathrm{cost}_T(C') \le (2\delta + 1) \cdot \mathrm{OPT}. \tag{2}$$

As (C', ϕ') is a feasible k-clustering of T, we have $\mathrm{cost}_T(C') \ge \mathrm{OPT}(T)$. On the other hand, the balanced k-clustering (C, ϕ) of T, computed in line 4 of the algorithm has cost at most $\alpha \cdot \mathrm{OPT}(T)$. Therefore, $\mathrm{cost}_T(C) \le \alpha \cdot \mathrm{OPT}(T) \le \alpha \cdot \mathrm{cost}_T(C')$, and hence, by inequality (2) we have

$$\mathrm{cost}_T(C) \le \alpha(2\delta + 1) \cdot \mathrm{OPT}. \tag{3}$$

To complete the proof, it just remains to show that the clustering (C, ϕ) of T yields a balanced k-clustering of S with cost at most $\mathrm{cost}_T(C) + \delta \cdot \mathrm{OPT}$. The transformation is as follows. We build a 3-partite directed graph G on the vertex set $S \cup T \cup C$ as follows. For each pair of points

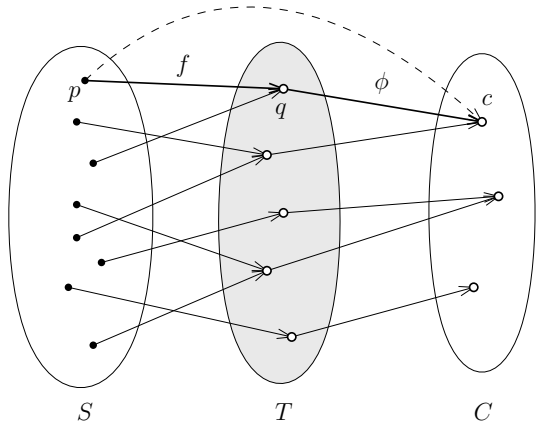

Figure 2: A balanced k-clustering of T transformed to a balanced k-clustering of S.

$p \in S$ and $q \in T$ where $q = f(p)$, we add an edge (p, q) with weight $w(p)$ to G. Moreover, for each pair of points $q \in T$ and $c \in C$ with $\phi(q, c) > 0$, we add an edge (q, c) with weight $\phi(q, c)$ to G. (See Figure 2.) Note that for each $q \in T$, the total weight of edges entering q is equal to $w_T(q)$ by equality (1), and also, the total weight of edges leaving q is $w_T(q)$ by the demand constraint. Moreover, for each vertex $c \in C$, the total weight of edges entering c is at most L by the capacity constraint.

Now, take a path from S to C in G, composed of two edges (p, q) and (q, c), and let W be the minimum of the weights of these two edges. We add a new edge (p, c) of weight W to the graph, and decrease the weights of (p, q) and (q, c) by W. Note that by this modification, the total weight of edges leaving p remains $w(p)$, and the total weight of edges entering c remains unchanged. Moreover, the total weight of edges entering q remains equal to the total weight of edges leaving q. Now, we remove any edge whose weight becomes zero, and repeat the above process until there remains no edge entering (and leaving) T. The graph G induced by $S \cup C$ represents a balanced k-clustering of S as it satisfies the capacity and demand constraints. Moreover, for each edge (p, c) in this graph, there has been a vertex q in the initial graph such that $d(p, c) \leq d(p, q) + d(q, c)$ by the triangle inequality. Since $d(p, q)$ is at most $\delta \cdot \mathrm{OPT}$, and $d(q, c) \leq \mathrm{cost}_T(C)$, we have

$$\mathrm{cost}_S(C) \leq \mathrm{cost}_T(C) + \delta \cdot \mathrm{OPT}. \tag{4}$$

Combining (3) and (4) implies $\mathrm{cost}_S(C) \leq (\alpha(2\delta + 1) + \delta) \cdot \mathrm{OPT}$, which completes the proof. □

The following lemma shows how a δ-coreset, for $\delta = 2$, can be computed efficiently in any metric space.

Lemma 2. *Let P be a point set in a metric space, and let $T = \mathrm{GMM}(P, k)$. Then T is a 2-coreset of P with respect to (balanced) k-center.*

Proof. Let $r = \max_{p \in P} d(p, T)$, and x be a point in P with $d(x, T) = r$. By construction of T in the GMM algorithm, the distance between any pair of points in T is at least r. As such, $Q = T \cup \{x\}$ forms a set of $k + 1$ points with minimum pairwise distance r. Fix an arbitrary superset S of P, and let C^* be an optimal balanced k-clustering of S with cost OPT. By the pigeonhole principle, there are at least two points u and v in Q which are assigned to a same center c^* in C^*. Now, by the triangle inequality, we have

$$d(u, v) \leq d(u, c^*) + d(c^*, v) \leq 2 \cdot \mathrm{OPT},$$

where the last inequality holds because both distances $d(u, c^*)$ and $d(v, c^*)$ are at most OPT. Combined with the fact that $r \leq d(u, v)$, we get $r \leq 2 \cdot \text{OPT}$. Moreover, by the definition of r, we know that for any point $p \in P$, there is a point $q \in T$ such that $d(p, q) \leq r$. Therefore, $d(p, q) \leq 2 \cdot \text{OPT}$, and hence, T is a 2-coreset of P of size k as desired. Note that our proof is independent of the value of capacity, L, and hence, T is also a 2-coreset with respect to the (classical) k-center problem. □

The following lemma provides an upper bound on the storage requirement for representing a clustering, which in turn is crucial for analyzing the space and communication complexity of our algorithm.

Lemma 3. *Let (C, ϕ) be a balanced k-clustering of a point set (T, w). A compact copy of the clustering can be stored in $O(|T| + |C|)$ memory.*

Proof. Consider an undirected bipartite graph $G = (T \cup C, E)$, where for each pair of vertices (q, c) with $\phi(q, c) > 0$, there is an edge (q, c) in E with weight $\phi(q, c)$. By the capacity and demand constraints, for each vertex $c \in C$, the total weight of edges incident to c is at most L, and for each vertex $q \in T$, the total weight of edges incident to q is $w_T(q)$.

In general, the number of edges in G, which determines the complexity of the clustering, is $O(|T| \cdot |C|)$. In the following, we show how the number of edges can be reduced to $O(|T| + |C|)$, without increasing the cost of the clustering. If G is acyclic, then the number of edges in G is at most $|T| + |C|$, and we are done. Otherwise, take a cycle $\mathcal{C}$ in G, and let e be the cheapest edge in the cycle, with weight W. Now, starting from e, we walk along the cycle, and increase the weight of each edge alternately by $-W$ and W. After this modification, the total weight of edges incident to each vertex of the graph does not change, and hence, the corresponding clustering remains feasible. Moreover, as we add no new edge during this process, the cost of clustering does not increase. Now, by this process, the weight of edge e becomes zero, while the weight of other edges remains non-negative. Removing edge e from G destroys the cycle $\mathcal{C}$. We repeat this process until there remains no cycle in the graph, and hence, the number of edges becomes $O(|T| + |C|)$. □

Now, we have all ingredients needed for presenting the main result of this section.

Theorem 4. *There is a two-round MPC algorithm that computes a $(5\alpha + 2)$-approximation to the (weighted) balanced k-center problem in any metric space, where α is the approximation factor of a sequential algorithm for the problem. The space requirement of each machine is $O(n/m + mk)$, where m is the number of machines, and n is the total input size. The average communication per machine is $O(k)$.*

Proof. By Theorem 1, Algorithm 2 has an approximation factor of $\alpha(2\delta + 1) + \delta$. If we use GMM for extracting δ-coresets in the algorithm, we get $\delta = 2$ by Lemma 2. As such, the approximation factor of the algorithm becomes $5\alpha + 2$. Each of the m local machines stores a partition of S of size $O(n/m)$. Moreover, the central machine uses $O(mk)$ memory to store m coresets of size k. The space required by each machine is therefore bounded by $O(n/m + mk)$.

In the first round of communication, each machine sends a coreset of size k to the central machine, amounting to a total communication of $O(mk)$ in the first round. The clustering sent back to each i-th machine is an assignment function ϕ restricted to $T_i \cup C$, which has size $O(|T_i| + |C|)$ by Lemma 3. Since $|T_i| = |C| = k$, the total communication in the second round is $O(mk)$. As such, the average communication per machine is $O(k)$. □

As stated in Theorem 4, the average communication per machine in our algorithm is $O(k)$. We show in the following theorem that this amount of communication is indeed optimal, under the composable coreset framework.

Theorem 5. *Any parallel/distributed algorithm for the balanced k-center problem with a bounded approximation guarantee under the composable coreset framework requires $\Omega(k)$ communication per machine.*

Proof. Suppose that the point set stored in the i-th machine, S_i, consists of exactly k points of pairwise distance M, for a large constant M. Consider the case where all other points in other machines are at a small distance ε from a single point $p \in S_i$. If the i-th machine does not send all of its k points to the central machine, then the cost of the final solution will be M, while the cost of an optimal solution is ε, leading to an unbounded approximation ratio. Therefore, all of the k points in S_i must be sent to the central machine. □

4 Metric Spaces with Bounded Doubling Dimension

In this section, we improve the approximation factor of the MPC algorithm presented in Section 3 for metric spaces with bounded doubling dimension, including all $\mathbb{R}^d$ spaces under any ℓ_p metric. Recall that the *doubling constant* D of a metric space denotes the minimum number of balls of radius $r/2$ needed to cover a ball of radius r. The *doubling dimension* of the metric space is defined as $d = \log_2 D$.

The core idea behind our improved algorithm is to enhance the precision of the δ-coresets extracted by local machines, similar to the technique used in [8]. To this end, we introduce a new notion of "*covering dimension*" of a metric space as follows.

Definition 1. *The* covering constant *of a metric space (X, d) is the minimum constant H such that for any point set $S \subseteq X$ and any integer $k \geq 1$,*

$$\mathrm{OPT}(S, Hk) \leq \frac{1}{2}\mathrm{OPT}(S, k), \tag{5}$$

where $\mathrm{OPT}(S, k)$ denotes the radius of an optimal k-center of S. The covering dimension *of a metric space is defined as $h = \log_2 H$.*

Informally, the covering constant H of a metric space denotes the minimum factor by which we need to increase the number of centers in order to reduce the optimal radius by a factor of at least $1/2$. Not all metric spaces have a bounded covering constant. For example, consider a metric space where the distance between any pair of distinct points is 1. If the covering constant of this space is bounded by H, then for a point set S of size $H+1$, we have $\mathrm{OPT}(S, H) = \mathrm{OPT}(S, 1) = 1$, which contradicts the definition of covering constant. Despite this fact, the following theorem shows that for any metric space with a bounded doubling dimension, the covering dimension is also bounded.

Lemma 6. *Any metric space with doubling dimension d has covering dimension at most $2d$.*

Proof. Consider a metric space with doubling dimension d, and let Q be the minimum number of balls of radius $r/4$ that cover a ball of radius r. By definition of doubling dimension, $Q \leq (2^d)^2 = 2^{2d}$. Now, consider a point set S, and let C^* be an optimal k-center of S. Fix a center $c \in C^*$, and let B be a ball of radius $\mathrm{OPT}(S, k)$ centered at c. The ball B can be covered by Q balls of radius $\mathrm{OPT}(S, k)/4$. Therefore, the area covered by c can be partitioned into

Q regions so that the pairwise distance in each region is at most $\mathrm{OPT}(S,k)/2$. For each non-empty region R, we select an arbitrary point as a center covering all the points in $S \cap R$. This yields a feasible solution with at most Qk centers having cost at most $\mathrm{OPT}(S,k)/2$. Therefore, $\mathrm{OPT}(S,Qk) \leq \mathrm{OPT}(S,k)/2$. This implies that the covering constant of the space is at most Q, and hence, the covering dimension is at most $\log_2 Q \leq 2d$. □

Now, we show how a bounded doubling dimension can help us achieve a better approximation factor for our problem.

Lemma 7. *Let P be a point set in a metric space with bounded doubling dimension d. Then, for any $\delta > 0$, a δ-coreset of P of size $O(k/\delta^{2d})$ can be computed in polynomial time.*

Proof. Let H denote the covering constant of our metric space. By Lemma 6, $H \leq 2^{2d}$. If we repeatedly apply inequality (5) for t times on a set S, we get

$$\mathrm{OPT}(S, H^t k) \leq \frac{1}{2^t}\mathrm{OPT}(S,k).$$

Given a point set P, let $T = \mathrm{GMM}(P,r)$, for $r = H^t k$, and let S be an arbitrary superset of P. By Lemma 2, T is a 2-coreset of P (with respect to r-center), i.e., for any point $p \in P$, there is a point $q \in T$ such that

$$d(p,q) \leq 2 \cdot \mathrm{OPT}(S,r) \leq 2 \cdot \frac{1}{2^t}\mathrm{OPT}(S,k).$$

Setting $t = \lceil \log_2(1/\delta) \rceil + 1$ implies $d(p,q) \leq \delta \cdot \mathrm{OPT}(S,k)$. Hence, T is a δ-coreset of P, as desired. The size of T is $H^t k = O(k2^{2dt}) = O(k/\delta^{2d})$. Moreover, T can be computed by the GMM algorithm in $O(|P| \cdot r) = O(nk/\delta^{2d})$ time, which is polynomial for any fixed d. □

Theorem 8. *There is a two-round MPC algorithm that computes a $(5+\varepsilon)$–approximation to the (weighted) balanced k-center problem in any metric space with bounded doubling dimension d, using $O(k/\varepsilon^{2d})$ average communication per machine. The space requirement for each machine is bounded by $O(n/m + mk/\varepsilon^{2d})$, where n is the total input size and m is the number of machines.*

Proof. By Theorem 1, Algorithm 2 yields an approximation factor of $\alpha + (2\alpha+1)\delta$ for the weighted balanced k-center problem. By Lemma 7, we can set δ arbitrary close to 0. Moreover, the sequential algorithm of Khuller and Sussmann [17] gives $\alpha = 5$. Therefore, by setting $\delta = \varepsilon/11$, the approximation factor of Algorithm 2 becomes $5+\varepsilon$. However, getting an $O(\varepsilon)$-coreset comes at the expense of increasing the coreset size to $O(k/\varepsilon^{2d})$ by Lemma 7. As such, the average communication per machine is $O(k/\varepsilon^{2d})$. The space complexity is analogously implied. □

5 Streaming Model

In this section, we show how our results from the previous sections can be extended to the streaming model, where input points arrive one by one over time, and the algorithm has only a limited memory, which is sub-linear in the size of the stream. The following theorem provides a generic streaming algorithm for the problem, based on the "merge-and-reduce" framework of Bentley and Saxe [6].

Theorem 9. *Suppose there is a sequential algorithm for computing a δ-coreset of size λ in polynomial time. Then, given a stream S of n points, we can maintain an $(\alpha + \ell\delta(2\alpha+1))$-approximation to the (weighted) balanced k-center of S using $O(\ell b \lambda)$ space, where $b \geq 2$ is an input integer, $\ell = \log_b(n/\lambda) - 1$, and α is the approximation factor of a sequential algorithm for the problem.*

Proof. The streaming algorithm is as follows. We start by $\ell + 1$ empty sets $T_0, T_1, \ldots, T_\ell$, each having capacity for storing $b\lambda$ points. When a point p in the stream S arrives, we simply put p in the set T_0. Whenever a set T_i (for $0 \leq i < \ell$) is full (i.e., $|T_i| = b\lambda$), we take a δ-coreset of T_i, add the coreset to T_{i+1}, and discard all the points in T_i by setting $T_i = \emptyset$.

By our algorithm, a point in T_i is discarded only when a coreset point representing it is added to T_{i+1}. Therefore, any point in the input stream is eventually represented by a point in some T_i. But, the points in T_i are obtained by repeatedly taking δ-coresets for i times. Therefore, the distance of a point p in the stream to its representing point $q \in T_i$ is at most $i\delta \cdot \mathrm{OPT}$. Since $i \leq \ell$, the set $T = \bigcup_{i=0}^{\ell} T_i$ forms an $(\ell\delta)$-coreset of S.

Upon query time, we run a sequential α-approximation algorithm on the set T to compute a balanced k-clustering of S and return it as output. By Theorem 1, an α-approximate balanced k-clustering of a δ-coreset of a set S yields an $(\alpha + \delta(2\alpha + 1))$-approximate solution to balanced k-center of S. If we use an $(\ell\delta)$-coreset instead of a δ-coreset in the theorem, we give an approximation factor of $\alpha + \ell\delta(2\alpha + 1)$.

Now, we prove an upper bound on the memory requirement. Let n_i denote the number of points added to the set T_i during the whole algorithm, including those points discarded from T_i at some point. By our algorithm, for each $b\lambda$ points added to T_i, we add a coreset of size λ to T_{i+1}. As such, $n_i \geq b \cdot n_{i+1}$, which implies $n_0 \geq b^\ell \cdot n_\ell$. The last set T_ℓ is only full when $n_\ell = b\lambda$, and hence, $n_0 \geq b^{\ell+1}\lambda$. Thus, if we set $\ell = \log_b(\frac{n}{\lambda}) - 1$, we have $n_0 \geq n$, and as such, we have enough memory for processing n points. The algorithm only keeps $\ell + 1$ sets T_i, each of size at most $b\lambda$. The total space used by the algorithm is therefore $O(\ell b\lambda)$. □

Theorem 10. *For any constant $0 < \varepsilon < 1$, there is a streaming algorithm for the balanced k-center problem in general metric spaces with approximation factor $(\frac{4}{\varepsilon} - 3)\alpha + (\frac{2}{\varepsilon} - 2)$ using $O(n^\varepsilon k^{1-\varepsilon})$ memory, where α is the approximation factor of a sequential algorithm for the problem.*

Proof. We use the streaming algorithm in Theorem 9, with parameter $b = (\frac{n}{\lambda})^\varepsilon$. The number of levels is thus $\ell = \log_b\left(\frac{n}{\lambda}\right) - 1 = \frac{1}{\varepsilon} - 1$, which is a constant. By Lemma 2, any point set in a metric space has a 2-coreset of size k. As such, we have $\delta = 2$ and $\lambda = k$. The final approximation factor by Theorem 9 is therefore $\alpha + 2(\frac{1}{\varepsilon} - 1)(2\alpha + 1) = (\frac{4}{\varepsilon} - 3)\alpha + \frac{2}{\varepsilon} - 2$. The space usage is also $O(\ell b\lambda) = O\left((\frac{n}{k})^\varepsilon k\right) = O\left(n^\varepsilon k^{1-\varepsilon}\right)$. □

Corollary 11. *Given a stream of n points in a general metric space, a $(5\alpha + 2)$-approximation to balanced k-center can be maintained using $O(\sqrt{nk})$ memory.*

Proof. This is a direct corollary of Theorem 10 by setting $\varepsilon = 1/2$. □

While Theorem 10 and Corollary 11 provide streaming algorithms with sub-linear space, it is usually preferred to have a space poly-logarithmic in n. The following theorem shows how such a poly-logarithmic dependency can be achieved at the expense of increasing the approximation factor by an $O(\log n)$ factor.

Theorem 12. *There is a streaming algorithm for the balanced k-center problem in any metric space with approximation factor $O(\log n)$ using $O(k \log n)$ memory.*

Proof. We use the same streaming algorithm presented in Theorem 9 with $b = 2$. Since input points are in a metric space, we can set $\delta = 2$ and $\lambda = k$ by Lemma 2. As such, we have $\ell = \log_b\left(\frac{n}{k}\right) - 1 \leq \log_2 n$. Therefore, the memory usage is $O(\ell bk) = O(k \log n)$. By Theorem 9, the approximation factor of the algorithm is therefore $\alpha + 2(2\alpha + 1)\log_2 n = O(\log n)$. □

In metric spaces with bounded doubling dimension, we can achieve a constant approximation factor using poly-logarithmic space, as shown below.

Theorem 13. *In metric spaces with bounded doubling dimension, we can maintain a* $(5+\varepsilon)$*-approximation to the balanced* k*-center problem in the streaming model using* $O(k \cdot \mathrm{polylog}(n))$ *memory.*

Proof. We use the streaming algorithm presented in Theorem 9, with parameter $b = 2$. It implies $\ell = \log_b\left(\frac{n}{\lambda}\right) - 1 \leq \log_2 n$. By Lemma 7, in all metric spaces with a bounded doubling dimension d, a δ-coreset of size $O(k/\delta^{2d})$ can be computed efficiently, for any $\delta > 0$. By setting $\delta = \varepsilon/((2\alpha+1)\log_2 n)$, the approximation factor of the algorithm becomes $\alpha + \ell\delta(2\alpha+1) \leq \alpha + \varepsilon$ by Theorem 9, which is at most $5+\varepsilon$, using the sequential algorithm available with $\alpha = 5$ [17]. Moreover, the coreset size is $\lambda = O(k/\delta^{2d}) = O(k(\log n/\varepsilon)^{2d})$. As such, the total space used by the algorithm is $O(\ell b \lambda) = O(k(\log n)^{2d+1}/\varepsilon^{2d})$, which is $O(k \cdot \mathrm{polylog}\,(n))$ for any fixed d and ε. □

6 Conclusion

In this paper, we presented new approximation algorithms for the weighted balanced k-center problem in the MPC and streaming models. In particular, we provided an MPC algorithm for the problem with an approximation factor of $5\alpha + 2$, substantially improving over the best previous factor of 32α, where α is the approximation factor of the corresponding sequential algorithm. The ideas used in this paper seem applicable to other variants of the center-based clustering, including k-median and k-means. It remains open whether the approximation factors provided in this paper can be further improved either in the MPC model, or in the one-pass streaming model.

Acknowledgments. The authors would like to thank Emad Aghajani for his valuable discussions during the early stages of this work.

References

[1] H.-C. An, A. Bhaskara, C. Chekuri, S. Gupta, V. Madan, and O. Svensson. Centrality of trees for capacitated k-center. *Math. Program.*, 154(1-2):29–53, 2015.

[2] J. Bar-Ilan, G. Kortsarz, and D. Peleg. How to allocate network centers. *J. Algorithms*, 15(3):385–415, 1993.

[3] M. Bateni, A. Bhaskara, S. Lattanzi, and V. Mirrokni. Distributed balanced clustering via mapping coresets. In *Proc. 27th Annu. Conf. Neural Info. Processing Systems*, pages 2591–2599, 2014.

[4] M. Bateni, H. Esfandiari, M. Fischer, and V. Mirrokni. Extreme k-center clustering. In *Proc. 35th AAAI Conf. Artif. Intell.*, volume 35, pages 3941–3949, 2021.

[5] P. Beame, P. Koutris, and D. Suciu. Communication steps for parallel query processing. *J. ACM*, 64(6), 2017.

[6] J. L. Bentley and J. B. Saxe. Decomposable searching problems I: static-to-dynamic transformation. *J. Algorithms*, 1(4):301–358, 1980.

[7] S. K. Bera, S. Das, S. Galhotra, and S. S. Kale. Fair k-center clustering in MapReduce and streaming settings. In *Proc. 31st Internat. Conf. World Wide Web*, pages 1414–1422, 2022.

[8] M. Ceccarello, A. Pietracaprina, and G. Pucci. Solving k-center clustering (with outliers) in MapReduce and streaming, almost as accurately as sequentially. *Proc. VLDB Endow.*, 12(7):766–778, 2019.

[9] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. *SIAM J. Comput.*, 33(6):1417–1440, 2004.

[10] M. Cygan, M. Hajiaghayi, and S. Khuller. LP rounding for k-centers with non-uniform hard capacities. In *Proc. 53rd Annu. IEEE Sympos. Found. Comput. Sci.*, FOCS '12, pages 273–282, 2012.

[11] A. Ene, S. Im, and B. Moseley. Fast clustering using MapReduce. In *Proc. 17th ACM SIGKDD Internat. Conf. Knowledge Discovery and Data Mining*, pages 681–689, 2011.

[12] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoret. Comput. Sci.*, 38:293–306, 1985.

[13] S. Guha. Tight results for clustering and summarizing data streams. In *Proc. 12th Internat. Conf. Database Theory*, pages 268–275, 2009.

[14] D. S. Hochbaum and D. B. Shmoys. A best possible heuristic for the k-center problem. *Math. Oper. Res.*, 10(2):180–184, 1985.

[15] P. Indyk, S. Mahabadi, M. Mahdian, and V. S. Mirrokni. Composable core-sets for diversity and coverage maximization. In *Proc. 33rd Sympos. Principles of Database Systems*, pages 100–108, 2014.

[16] H. Karloff, S. Suri, and S. Vassilvitskii. A model of computation for MapReduce. In *Proc. 21st ACM-SIAM Sympos. Discrete Algorithms*, SODA '10, pages 938–948, 2010.

[17] S. Khuller and Y. J. Sussmann. The capacitated k-center problem. *SIAM J. Discrete Math.*, 13(3):418–403, 2000.

[18] G. Malkomes, M. J. Kusner, W. Chen, K. Q. Weinberger, and B. Moseley. Fast distributed k-center clustering with outliers on massive data. In *Proc. 28th Annu. Conf. Neural Info. Processing Systems*, pages 1063–1071, 2015.

[19] J. McClintock and A. Wirth. Efficient parallel algorithms for k-center clustering. In *Proc. 45th Internat. Conf. Parallel Processing*, pages 133–138, 2016.

[20] R. M. McCutchen and S. Khuller. Streaming algorithms for k-center clustering with outliers and with anonymity. In *Proc. 11th Internat. Workshop Approx. Algorithms*, volume 5171, pages 165–178, 2008.

[21] K. Mirjalali and H. Zarrabi-Zadeh. Improved algorithms for distributed balanced clustering. In *Proc. 3rd Internat. Conf. Topics Theoret. Comput. Sci.*, volume 12281 of *Lecture Notes Comput. Sci.*, pages 72–84, 2020.

[22] S. Muthukrishnan. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1(2):117–236, 2005.

[23] H.-L. Nguyen, Y.-K. Woon, and W. K. Ng. A survey on data stream clustering and classification. *Knowledge Info. Syst.*, 45(3):535–569, 2015.

[24] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. de Carvalho, and J. a. Gama. Data stream clustering: A survey. *ACM Computing Surveys*, 46(1):13:1–13:31, 2013.

[25] A. Zubaroğlu and V. Atalay. Data stream clustering: a review. *Artif. Intell. Rev.*, 54(2):1201–1236, 2021.