







Fully-Scalable MPC Algorithms for Clustering in High Dimension

Artur Czumaj  



Department of Computer Science, University of Warwick, Coventry, UK

Guichen Gao  

School of Computer Science, Peking University, Beijing, China

Shaofeng H.-C. Jiang  

School of Computer Science, Peking University, Beijing, China

Robert Krauthgamer  

Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel

Pavel Veselý  

Computer Science Institute of Charles University, Prague, Czech Republic

Abstract

We design new parallel algorithms for clustering in high-dimensional Euclidean spaces. These algorithms run in the Massively Parallel Computation (MPC) model, and are *fully scalable*, meaning that the local memory in each machine may be n^σ for arbitrarily small fixed $\sigma > 0$. Importantly, the local memory may be substantially smaller than the number of clusters k , yet all our algorithms are fast, i.e., run in $O(1)$ rounds.

We first devise a fast MPC algorithm for $O(1)$ -approximation of uniform FACILITY LOCATION. This is the first fully-scalable MPC algorithm that achieves $O(1)$ -approximation for any clustering problem in general geometric setting; previous algorithms only provide $\text{poly}(\log n)$ -approximation or apply to restricted inputs, like low dimension or small number of clusters k ; e.g. [Bhaskara and Wijewardena, ICML'18; Cohen-Addad et al., NeurIPS'21; Cohen-Addad et al., ICML'22]. We then build on this FACILITY LOCATION result and devise a fast MPC algorithm that achieves $O(1)$ -bicriteria approximation for k -MEDIAN and for k -MEANS, namely, it computes $(1 + \varepsilon)k$ clusters of cost within $O(1/\varepsilon^2)$ -factor of the optimum for k clusters.

A primary technical tool that we introduce, and may be of independent interest, is a new MPC primitive for geometric aggregation, namely, computing for every data point a statistic of its approximate neighborhood, for statistics like range counting and nearest-neighbor search. Our implementation of this primitive works in high dimension, and is based on consistent hashing (aka sparse partition), a technique that was recently used for streaming algorithms [Czumaj et al., FOCS'22].

2012 ACM Subject Classification Theory of computation \rightarrow Massively parallel algorithms; Theory of computation \rightarrow Facility location and clustering; Theory of computation \rightarrow Randomness, geometry and discrete structures

Keywords and phrases Massively parallel computing, high dimension, facility location, k -median, k -means

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.50

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: <https://arxiv.org/abs/2307.07848> [32]

Funding *Artur Czumaj*: Partially supported by the Centre for Discrete Mathematics and its Applications (DIMAP), by EPSRC award EP/V01305X/1, by a Weizmann-UK Making Connections Grant, and by an IBM Award.

Shaofeng H.-C. Jiang: Research partially supported by a national key R&D program of China No. 2021YFA1000900 and a startup fund from Peking University.



© Artur Czumaj, Guichen Gao, Shaofeng H.-C. Jiang, Robert Krauthgamer, and Pavel Veselý;

licensed under Creative Commons License CC-BY 4.0

51st International Colloquium on Automata, Languages, and Programming (ICALP 2024).

Editors: Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson;

Article No. 50; pp. 50:1–50:20



Leibniz International Proceedings in Informatics

LIPIcs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



Robert Krauthgamer: Partially supported by the Israel Science Foundation grant #1336/23, by a Weizmann-UK Making Connections Grant, by the Israeli Council for Higher Education (CHE) via the Weizmann Data Science Research Center, and by a research grant from the Estate of Harry Schutzman.

Pavel Veselý: Partially supported by GA ČR project 22-22997S and by Center for Foundations of Modern Computer Science (Charles Univ. project UNCE 24/SCI/008).

1 Introduction

Clustering large data sets is a fundamental computational task that has been studied extensively due to its wide applicability in data science, including, e.g., unsupervised learning, classification, data mining. Two highly popular and extremely basic problems are k -MEDIAN and k -MEANS. In the geometric (Euclidean) setting, the k -MEDIAN problem asks, given as input an integer k and a set $P \subseteq \mathbb{R}^d$ of n data points, to compute a set $C \subset \mathbb{R}^d$ of k center points, so as to minimize the sum of distances from each point in P to its nearest center. The k -MEANS problem is similar except that instead of the sum of distances, one minimizes the sum of squares of distances. A closely related problem is (uniform) FACILITY LOCATION, which can be viewed as a Lagrangian relaxation of k -MEDIAN, where the number of centers can vary but it adds a penalty to the objective, namely, each center (called open facility) incurs a given cost $f > 0$. Other variants include a similar relaxation of k -MEANS, or generalizing the squaring each distance to raising it to power $z \geq 1$ (see Section 2 for formal definitions), and there is a vast literature on these computational problems.

Clustering is often performed on massive datasets, and it is therefore important to study clustering in the framework of distributed and parallel computations. We consider fundamental clustering problems in the theoretical model of *Massively Parallel Computation (MPC)*, which captures key aspects of modern large-scale computation systems, such as MapReduce, Hadoop, Dryad, or Spark. The MPC model was introduced over a decade ago by Karloff, Suri, and Vassilvitskii [55], and over time has become the standard theoretical model to study data-intensive parallel algorithms (see, e.g., [11, 41, 47]). At a high level, the MPC model consists of many machines that communicate with each other synchronously in a constrained manner, in order to solve a desired task in a few rounds. In more detail, an MPC system has m machines, each with a *local memory* of s words, hence the system's *total memory* is the product $m \cdot s$. Computation takes place in synchronous rounds, where machines perform arbitrary computations on their local memory and then exchange messages with other machines. Every machine is constrained to send and receive at most s words in every round, and every message must be destined to a single machine (not a broadcast). At each round, every machine processes its incoming messages to generate its outgoing messages, usually without any computational restrictions on this processing (e.g., running time). At the end of the computation, the machines collectively output the solution. The efficiency of an algorithm is measured by the *number of rounds* and by the local-memory size s . The total space should be low as well, but this is typically of secondary importance.

The local-memory size s is a key parameter and should be sublinear in the input size N (for if $s \geq N$ then any sequential algorithm can be executed locally by a single machine in one round). We focus on the more challenging regime, called *fully-scalable MPC*, where the local-memory size is an arbitrarily small polynomial, i.e., $s = N^\sigma$ for any fixed $\sigma \in (0, 1)$. This regime is highly desirable because in practice, the local memory of the machines is limited by the hardware available at hand, and the parameter σ can be used to model this limitation. The total memory should obviously be large enough to fit the input, i.e., $m \cdot s \geq N$, and ideally not much larger.

Modern research on MPC algorithms aims to solve fundamental problems in as few rounds as possible, ideally in $O(1)$ rounds, even in the fully-scalable regime, and so far there have been some successes in designing fast (i.e., $O(1)$ -rounds) algorithms for fixed $\sigma < 1$; see e.g., [4, 12, 41]. In contrast, many graph problems, including fundamental ones such as connectivity or even distinguishing between 1 and 2 cycles, seem to require super-constant number of rounds (at the same time, proving a super-constant lower bound would yield a breakthrough in circuit complexity [62]). One can also consider smaller local space s , i.e., $\sigma = o(1)$, and in fact many algorithms work as long as $s \geq \text{polylog}(N)$. In this regime, the best one can hope for is usually not $O(1)$ but rather $O(\log_s N)$ rounds, because even broadcasting a single number to all machines requires $\Omega(\log_s N)$ rounds; see [62] for further discussion.

Clustering of massive datasets has received significant attention recently, with numerous algorithms that span multiple computational models, such as streaming [18, 63] or distributed and parallel models [16, 35, 15, 8, 9, 60, 6, 21, 12, 34]. However, this advance is facing a barrier, first identified by Bhaskara and Wijewardena [13]: State-of-the-art methods for k -clustering (i.e., when the number k of clusters is specified in the input) typically fail to yield fully-scalable MPC algorithms, because when implemented in the MPC model, these methods usually require local-memory size $s \geq \Omega(k)$. For example, many streaming algorithms for k -clustering problems are based on a linear sketch and thus readily applicable to the MPC model; however, all known streaming algorithms require $\Omega(k)$ space, and in fact there are lower bounds to this effect [23] even in low dimension. Another example is coresets [43], an extremely effective method to decrease the size of the dataset, even in high dimension; see [29, 17, 26] for the latest in this long line of research. Coresets can often be merged and/or applied repeatedly via the so-called “merge-and-reduce” framework [43], and can thus be applied successfully in many different settings, including the parallel setting. However, this method suffers from the drawback (which so far seems inherent) that each coreset must be stored in its entirety in a single machine, and it is easy to see that a coreset for k -clustering must be of size at least k ; see also [26].

These shortcomings have led to a surge of interest in MPC algorithms for clustering, with emphasis on fully-scalable and fast algorithms (i.e., taking $O(1)$ rounds even for large k) that work in high dimension d . The first such result for k -clustering was by Bhaskara and Wijewardena [13]; their $O(1)$ -round fully-scalable MPC algorithm achieves polylogarithmic bicriteria approximation for k -MEANS, i.e., it outputs $k \cdot \text{polylog}(n)$ centers (or clusters) whose cost is within $\text{polylog}(n)$ -factor of the optimum for k centers. Recent work by Cohen-Addad et al. [27] achieves for k -MEDIAN a true $\text{polylog}(n)$ -approximation (i.e., without violating the bound k on number of centers); their algorithm actually computes a hierarchical clustering, that is, a single hierarchical structure of centers that induces an approximate solution for k -MEDIAN for every k . These were the best fully-scalable $O(1)$ -round MPC algorithms known for k -MEDIAN and k -MEANS prior to our work, and it remained open to achieve better than $O(\log n)$ -approximation, even as a bicriteria approximation.

Not surprisingly, previous work on MPC algorithms has focused also on achieving improved approximation for restricted inputs. In particular, $(1 + \varepsilon)$ -approximation is achieved in [28] for k -MEANS and k -MEDIAN on inputs that are *perturbation-resilient* [14], meaning that perturbing pairwise distances by (bounded) multiplicative factors does not change the optimal clusters; unfortunately, results for such special cases rarely generalize to all inputs. There are also known fully-scalable MPC algorithms for k -center [10, 30], but they are applicable only in low dimension d and thus less relevant to our focus.

Despite these many advances on fully-scalable MPC algorithms, we are not aware of $O(1)$ -approximation *for any clustering problem* in high dimension.¹ From a technical perspective, the primary difficulty is to distribute the data points across machines, so that points arriving to the same machine are from the same cluster. This task seems to require knowing the clustering, which is actually the desired output, so we run into a chicken-and-egg problem! Fortunately, powerful algorithmic tools can spot points that are close together, even in high-dimensional geometry. Namely, locality sensitive hashing (LSH) is employed in [13] and tree embedding is used in [27], but as mentioned above, these previous results do not achieve $O(1)$ -approximation. We rely instead on the framework of *consistent hashing*, which was first employed for streaming algorithms in [33], although it was originally proposed in [53]; see Section 1.2 for details.

1.1 Our Results

We devise fully-scalable $O(1)$ -round MPC algorithms for a range of clustering problems in \mathbb{R}^d , most notably FACILITY LOCATION, k -MEDIAN, and k -MEANS. We first devise an $O(1)$ -approximation algorithm for FACILITY LOCATION and then exploit the connection between the problems to solve k -clustering.

By convention, we express memory bounds in machine words, where each word can store a counter in the range $[\text{poly}(n)]$, i.e., $O(\log n)$ bits, and/or a coordinate of a point from \mathbb{R}^d with restricted precision (comparable to that of a point from P). For example, the input $P \subset \mathbb{R}^d$, $|P| = n$ fits in nd words. Throughout, the notation $O_\varepsilon(\cdot)$ hides factors that depend on ε . We stress that in the next theorem, the memory bounds are polynomial in the dimension d (and not exponential); this is crucial because the most important case is $d = O(\log n)$, as explained in Remark 1.2f below.

► **Theorem 1.1 (Simplified version).** *Let $\varepsilon, \sigma \in (0, 1)$ be fixed. There is a randomized fully-scalable MPC algorithm that, given a multiset $P \subset \mathbb{R}^d$ of n points distributed across machines with local memory of size $s \geq n^\sigma \cdot \text{poly}(d)$, computes in $O_\sigma(1)$ rounds an $O_\varepsilon(1)$ -approximation for uniform FACILITY LOCATION. The algorithm uses $O(n^{1+\varepsilon}) \cdot \text{poly}(d)$ total space.*

► **Remark 1.2.** This simplified statement omits standard technical details:

- a) We consider here fixed σ , but the algorithm works for any local-memory size $s \geq \text{poly}(d \log n)$, and in that case the number of rounds becomes $O(\log_s n)$.
- b) Similarly, ε can be part of the input, and the approximation factor is in fact $(1/\varepsilon)^{O(1)}$.
- c) The algorithm succeeds with high probability $1 - 1/\text{poly}(n)$.
- d) The input P can be a multiset, in contrast to streaming algorithms for such problems [49, 33], where data points must be distinct.
- e) The algorithm outputs a feasible solution, i.e., a set of facilities $F \subset \mathbb{R}^d$ and an assignment of the input points to facilities. In fact, in all our algorithms $F \subseteq P$. We focus throughout on computing F , since our methods can easily compute a near-optimal assignment given F .
- f) We state here the dependence on d explicitly for sake of completeness, but our result works whenever $s \geq \text{polylog}(dn)$, which is preferable when $d \gg \log n$. This is achieved by reducing to the case $d = O(\log n)$, using a standard dimension reduction, as discussed in Remark 2.4.

¹ Having said that, $(3 + \varepsilon)$ -approximation is known for (non-geometric) correlation clustering [12].

Our full result (in the full version) is more general in two respects. First, it addresses a generalization of FACILITY LOCATION where distances are raised to power $z \geq 1$ (see Section 2 for definitions). Second, it assumes access to consistent hashing with parameters Γ and Λ , and here we plugged in a known construction [33] that achieves a tradeoff $\Gamma = O(1/\varepsilon)$ and $\Lambda = n^\varepsilon$ for any desired $\varepsilon \in (0, 1)$; see Lemma 2.3 and Remark 2.4. The description of the hash function should be included in our MPC algorithm, but it takes only $\text{poly}(d)$ bits, which is easily absorbed in our bounds.

Previously, no fully-scalable MPC algorithm was known for FACILITY LOCATION, although one can apply the algorithm for k -MEDIAN from [27] to obtain an $O(\text{polylog}(n))$ -approximation in $O(\log_s n)$ rounds. Another previous result that one could use is a streaming algorithm that achieves $O(1)$ -approximation [33]. Although many streaming algorithms, including this one, can be implemented in the MPC model, they approximate the optimal value without reporting a solution; see Section 1.3 for details. Thus, from a technical perspective, our chief contribution is to compute an approximately optimal solution, which is a set of facilities F , in the fully-scalable regime.

k -Clustering. Our second result presents MPC algorithms for k -MEDIAN and k -MEANS that achieve an $(O(\mu^{-2}), 1 + \mu)$ -bicriteria approximation, for any desired $\mu \in (0, 1)$. As usual, (α, β) -bicriteria approximation means that for every input, the algorithm outputs at most βk centers whose cost is at most α -factor larger than the optimum cost for k centers. By letting $\mu > 0$ be arbitrarily small but fixed, our algorithm gets arbitrarily close (multiplicatively) to k centers, while still approximating the optimal cost within $O(1)$ -factor; in both respects, this is far stronger than the previous bicriteria approximation for k -MEANS [13]. Our result is incomparable to the previous bound for k -MEDIAN, which is a true $O(\text{polylog}(n))$ -approximation [27]. Nevertheless, our bicriteria approximation breaks a fundamental technical barrier in their tree-embedding approach, which cannot go below $O(\log n)$ ratio, due to known distortion lower bounds, and moreover does not generalize to k -MEANS, because it fails to preserve the squared distance.

Our approach is to tackle k -MEDIAN and k -MEANS via Lagrangian relaxation and rely on our algorithm for FACILITY LOCATION. This approach can inherently handle large k , because our core algorithms for FACILITY LOCATION can even output n clusters. The Lagrangian technique was initiated by Jain and Vazirani [51], who achieved a true $O(1)$ -approximation for k -MEDIAN by leveraging special properties of their algorithm for FACILITY LOCATION (see Section 1.2). However, their primal-dual approach for FACILITY LOCATION seems challenging to implement in fully-scalable MPC. We thus develop an alternative approach that is inherently more parallel, albeit achieves only bicriteria approximation for k -clustering.

► **Theorem 1.3 (Simplified version).** *Let $\varepsilon, \sigma \in (0, 1)$ be fixed. There is a randomized fully-scalable MPC algorithm that, given $\mu \in (0, 1)$, $k \geq 1$, and a multiset $P \subset \mathbb{R}^d$ of n points distributed across machines of memory $s \geq n^\sigma \cdot \text{poly}(d)$, computes in $O_\sigma(1)$ rounds an $(O_\varepsilon(\mu^{-2}), 1 + \mu)$ -bicriteria approximation for k -MEDIAN, or alternatively k -MEANS. The algorithm uses $O(n^{1+\varepsilon}) \cdot d^{O(1)}$ total space.*

This simplified statement omits the same standard details as in Theorem 1.1, and Remark 1.2 applies here as well. In addition, it is known for k -MEDIAN and k -MEANS (but not known for FACILITY LOCATION) that any (true) finite approximation requires $\Omega(\log_s n)$ rounds [13].

1.2 Technical Overview

We overview the main components in our algorithms, and highlight technical ideas that may find more applications in the future. We focus in this overview on FACILITY LOCATION and k -MEDIAN, noting that our algorithm for k -MEANS (and other powers $z \geq 1$) is essentially the same. We further assume that $s = n^\sigma \cdot \text{poly}(d)$ for a fixed $\sigma \in (0, 1)$, and we aim to achieve round complexity $O(\log_s n) = O_\sigma(1)$.

Facility Location. Several different algorithmic approaches have been used in the past to achieve $O(1)$ -approximation for FACILITY LOCATION, including LP-rounding [19, 56], primal-dual [51], greedy [50], and local search [5]. Some of these sequential algorithms were adapted to the PRAM model [16], i.e., to run in polylogarithmic parallel time (RNC algorithms). These algorithms can be implemented in the MPC model, but as some logarithmic factors in the time complexity seem inherent, these fall short of $O(1)$ rounds in the fully-scalable regime. Our starting point is the Mettu-Plaxton (MP) algorithm [61], which is a combinatorial algorithm inspired by [51], that has been previously used to achieve $O(1)$ -approximation in a few related models, particularly streaming, congested clique, and sublinear-time computation [7, 38, 33]. At a high level, this MP algorithm has two steps, it first computes a “radius” $r_p > 0$ for every $p \in P$, and then uses these r_p values to determine which facilities to open. However, implementing these steps in MPC is technically challenging:

- Computing r_p (approximately) can be reduced to counting the number of points in P within a certain distance from p [7], which is non-trivial to compute in $O(1)$ rounds, because many geometric techniques are ineffective in high dimension, for instance quadrees/tree embeddings incur large approximation error and grids/nets require large memory. We overcome this issue by devising a new MPC primitive for geometric aggregation (in high dimension), that can handle a wide range of statistics, including the counting mentioned above.
- The MP algorithm determines which facilities to open by scanning the points in P in order of non-decreasing r_p value and deciding greedily whether to open each point as a facility. We design a new algorithm that avoids any sequential decision-making and decides whether to open each facility locally and in parallel. Our new algorithm may thus be useful also in other models.

Let us discuss these two new ideas in more detail.

MPC Primitive for Geometric Aggregation in High Dimension. We propose a new MPC primitive for aggregation tasks in high-dimensional Euclidean spaces (see Theorem 3.1 for details). Given a radius $r > 0$, this primitive outputs, for every data point $p \in P$, a certain statistic of the ball $B_P(p, r) := P \cap \{y \in \mathbb{R}^d : \text{dist}(x, y) \leq r\}$. Our implementation can handle any statistic that is defined by a *composable* function f , which means that $f(\cup_i S_i)$ for disjoint sets $S_i \subset \mathbb{R}^d$ can be evaluated from the values $\{f(S_i)\}_i$. Composable functions include counting the number of points, or finding the smallest label (when data points are labeled). In fact, this primitive can even be used for approximate nearest-neighbor search in parallel for all points (see Section 3.1). This natural aggregation tool plays a central role in all our MPC algorithms, and we expect it to be useful for other MPC algorithms in high dimension.

Technically, exact computation of such statistics may be difficult, and our algorithm estimates the statistic by evaluating it (exactly) on a set $A_P(p, r)$ that approximates the ball $B_P(p, r)$ in the sense that it is sandwiched between $B_P(p, r)$ and $B_P(p, \beta r)$ for some error parameter $\beta \geq 1$. To implement this algorithm in MPC, a natural idea is to “collect” all

data points in the ball $B_P(p, r)$ at a single machine, however this set might be too large to fit in one or even few machines. A standard technique to resolve this issue in low dimension is to impose a grid of fine resolution, say εr , and move each data point to its nearest grid point, which provides a decent approximation (e.g., $\beta = 1 + \sqrt{d\varepsilon}$). However, a ball $B_P(p, r)$ might contain $\varepsilon^{-\Theta(d)}$ grid points, which for high dimension might still not fit in one machine.

Our approach is to use *consistent hashing* (see Definition 2.2), which was first introduced in [53] under the name sparse partition, and was recently employed in the streaming setting for FACILITY LOCATION in high dimension [33]. Roughly speaking, consistent hashing is a partition of the space \mathbb{R}^d , such that each part has diameter bounded by βr , and every ball $B_P(p, r)$ intersects at most $n^{1/\beta}$ parts.² Our algorithm moves each data point $p \in P$ to a fixed representative point inside its own part, then computes the desired statistic on each part (namely, on the data points moved to the same representative), and finally aggregates, for each $p \in P$, the $n^{1/\beta}$ statistics of the parts that intersect $B_P(p, r)$. This algorithm can be implemented in $O(\log_s n)$ rounds on MPC, albeit with slightly bigger total space $n^{1+1/\beta}$. An interesting feature of this algorithm is that its core is deterministic and hence leads to new *deterministic* MPC algorithms, including for approximate nearest-neighbor search; see Section 3.1.

Computing A Solution for Facility Location. Our algorithm is based on the MP algorithm, where a key notion is the “radius” $r_p > 0$ defined for each data point $p \in P$. Formally, it takes the value r such that serving all points in the ball $B(p, r)$ by p , i.e., opening a facility at p and assigning points to p , incurs a cost of $|B_P(p, r)| \cdot r$. This value r always exists and is unique. It is known that a constant-factor approximation \hat{r}_p of r_p satisfies that $|B_P(p, \hat{r}_p)| \approx 1/\hat{r}_p$, hence computing $|B_P(p, r)|$ for $O(\text{poly } \log n)$ different values of r suffices to compute an $O(1)$ -approximation of r_p [7, Lemma 1]. However, it is not easy to estimate $|B_P(p, r)|$ in MPC (simultaneously for all p and r), and the abovementioned geometric aggregation only estimates $|A_P(p, r)|$ for some $A_P(p, r)$ that is sandwiched between $B(p, r)$ and $B(p, \beta r)$. Nonetheless, we show this weaker estimate suffices for approximating r_p within $O(\beta)$ factor. Thus, our aggregation primitive yields an $O(1)$ -approximation for all the r_p values in $O(1)$ rounds.

An $O(1)$ -factor estimate of the optimal cost OPT can be computed from an $O(1)$ -approximation of the r_p values for all $p \in P$, because $\sum_{p \in P} r_p = \Theta(\text{OPT})$ [7, 38]. Moreover, the r_p values can be used to compute an $O(1)$ -approximate *solution* for FACILITY LOCATION. Specifically, the MP algorithm [61] scans the points in P in order of non-decreasing r_p value, and opens a facility at each point p if so far no facility was opened within distance $2r_p$ from p . The sequential nature of this algorithm makes it inadequate for MPC, and we therefore design a new algorithm that makes decisions in parallel. It has two separate rules to decide whether to open a facility at each point $p \in P$:

- (P1) open a facility at p with probability $\Theta(r_p)$, independently of other points; and
- (P2) open a facility at p if it has the smallest label among $B_P(p, r_p)$, where each point $q \in P$ is assigned independently a random label $h(q) \in [0, 1]$.

Let us give some intuition for these rules. Rule (P1) is a straightforward way to use the r_p values so that the expected opening cost is $O(\sum_p r_p) = O(\text{OPT})$, but it is not sufficient by itself because the connection cost might be too large. Indeed, if a cluster of points is very far from all other points, say, a cluster of t points all with the same $r_p = 1/t$, then with constant

² This tradeoff between βr and $n^{1/\beta}$ is just one specific choice of known parameters. Our theorem works with any possible parameters of consistent hashing, see Lemma 2.3 and Remark 2.4.

probability, (P1) does not open any facility inside this cluster, and the closest open facility is prohibitively far. However, (P2) guarantees that at least one facility is opened inside this cluster, at the point that has the smallest label. Rule (P2) is not sufficient by itself as well. Indeed, let $x_1 \in P$ be the minimizer from the viewpoint of p , i.e., have the smallest label in $B_P(p, r_p)$. Then it may happen that there is no facility at x_1 because $B_P(x_1, r_{x_1})$ has another point x_2 with an even smaller label. The same may happen also to x_2 , and we may potentially get a long “assignment” sequence $(p = x_0, x_1, x_2, x_3, \dots, x_t)$, where each x_{i+1} is the minimizer from the viewpoint of x_i and only the last point x_t is an open facility. In this case, the connection cost of p can be as large as $\sum_{i=0}^{t-1} r_{x_i}$ (i.e., matching the bound obtained by the triangle inequality), which might be unbounded relative to r_p . This might happen not only for one point p but actually for many points, and the total connection cost would be prohibitive.

We deal with this assignment issue using rule (P1), and in effect use both rules together. Given such an assignment sequence, we prove that for every $i \geq 0$, either (i) $B_P(x_i, r_{x_i})$ contains an open facility with constant probability, or (ii) $r_{x_{i+1}} \leq r_{x_i}/2$, i.e., the r_p value drops significantly in the next step. Property (i) means that the sequence stops at x_i with constant probability and thus, unless (ii) occurs, the expected length of the assignment sequence is $O(1)$. Property (ii) implies that the connection cost of the next step drops significantly as it is proportional to its r_p value, and hence, we just sum up a subsequence of geometrically decreasing r_p values. Combining the two properties, we obtain that the expected connection cost for every point p is $O(r_p)$. Hence, the expected total connection cost is $O(\sum_p r_p) = O(\text{OPT})$.

The main technical challenge is to show property (i) when (ii) does not occur. Specifically, we need to show that, given a partial reassignment sequence $(p = x_0, \dots, x_{i-1})$, the probability that the sequence does not terminate at x_i is bounded by a small constant. This event can be broken into two sub-events: (a) rule (P1) does not open any facility at the “new” points of $B_P(x_i, r_{x_i})$, where a point is considered new if it is not in $\cup_{j < i} B_P(x_j, r_{x_j})$; and (b) the smallest label appears at a new point (and thus rule (P2) does not open a facility at x_i). Let $t \in [0, 1]$ be the fraction of points that are new in $B_P(x_i, r_{x_i})$. Then the probability of (a) is roughly $(1 - r_{x_i})^{\Omega(t/r_{x_i})} \approx e^{-\Omega(t)}$, where this calculation crucially uses that (ii) does not occur, which means all new points have a similar r_p value as x_i , and that for every point x , the ball $B_P(x, r_x)$ roughly contains $\Omega(1/r_x)$ points. Since the two events are independent and since (b) happens with probability t by symmetry, we conclude that the probability we need to bound is at most $\exp(-O(t)) \cdot t \leq O(1)$. Here, one can observe that (P1) and (P2) are “complementing” each other to make the said probability small: when t is large, the probability $\exp(-O(t))$ of (a), which comes from rule (P1), is small; otherwise, the probability t of (b), which comes from rule (P2), is small.

The idea of opening facilities using random labels and analyzing the cost by constructing an assignment sequence was previously used in [3]. The context there is of a dynamically changing input, and this technique is used to limit changes in the solution over time, while our goal is to have a fast parallel implementation. Although the high level idea is similar, the setup is quite different, as their algorithm needs a solution to a linear-programming (LP) relaxation for FACILITY LOCATION, while ours only needs the r_p values; and consequently also the analysis is different, as their analysis uses the LP constraints and bounds the cost relative to LP value, while our analysis crucially uses basic properties of the r_p values.

It remains to bound the opening cost which is the easy part. We show that the number of open facilities is $O(\sum_p r_p)$ in expectation. Indeed, for points selected by rule (P1) this is clear. For rule (P2), since $B_P(p, r_p)$ contains at least $1/r_p$ points, the probability that p has the smallest label in $B_P(p, r_p)$ is at most r_p .

When implementing our algorithm in the MPC model, the only non-trivial part (except for estimating the r_p values) is to check that p has the minimum label in the ball $B(p, r_p)$. Nevertheless, it is sufficient to look for the minimum label in a larger set that approximates the ball, such as a set sandwiched between this ball and a ball whose radius is larger by $O(1)$ -factor. Therefore, the MPC primitive for geometric aggregation is sufficient to execute rule (P2).

Overall, these rules compute a set of open facilities. To compute also an assignment of each point $p \in P$ to an open facility, we use our approximate nearest-neighbor search algorithm, to find for each point $p \in P$ its $O(1)$ -approximately closest facility (see Section 3.1). Notice that the assignment algorithm searches for the closest facility, but the analysis is still based on the assignment sequence as above, even though the connection cost of this sequence may substantially exceed the distance to the closest facility.

k -Clustering. Our algorithm for k -MEDIAN follows the Lagrangian-relaxation framework established by Jain and Vazirani [51] (and used implicitly earlier by Garg [37]). They managed to obtain a true $O(1)$ -approximation for k -MEDIAN by leveraging a special property guaranteed by their algorithm for FACILITY LOCATION, namely, its output solution has opening cost cost_O and connection cost cost_C that satisfy $\alpha \cdot \text{cost}_O + \text{cost}_C \leq \alpha \cdot \text{OPT}$ for a certain $\alpha = O(1)$. Unfortunately, this stronger property is obtained via a highly sequential primal-dual approach, and seems difficult to implement efficiently in MPC, particularly because it is too sensitive for the known toolkit for high dimension, like locality sensitive hashing (LSH) and our geometric aggregation via consistent hashing.

We therefore take another approach of relying on a *generic* γ -approximation algorithm for FACILITY LOCATION, and using it in a black-box manner to obtain bicriteria approximation for k -MEDIAN. Our algorithm can output $(1 + \mu)k$ centers whose cost is at most $O(1/\mu^2)$ -factor larger than the optimum cost for k centers, for any desired $0 < \mu < 1$. This type of tradeoff, where the number of centers is arbitrarily close to k , was relatively less understood, as previous work has focused mostly on a smaller $O(1)$ -factor in the cost, but using significantly more than k centers [57, 58, 1, 64, 46]. The result of [59] does give $(1 + \mu)k$ centers, and is thus the closest to ours in terms of bicriteria bounds, however it relies explicitly on LP rounding, which seems difficult to implement in MPC. To the best of our knowledge, obtaining $(1 + \mu)k$ centers for k -clustering by a black-box reduction to FACILITY LOCATION was not known before. We believe this new reduction, and the smooth tradeoff it offers, may be of independent interest.

In more detail, the black-box reduction from k -MEDIAN to FACILITY LOCATION goes as follows: Assume momentarily that we know (an approximation of) the optimal clustering cost OPT , and consider the clustering instance as an input for FACILITY LOCATION with opening cost $\mathfrak{f} := \text{OPT}/k$. The optimal cost of this instance is at most $k \cdot \mathfrak{f} + \text{OPT} = 2 \text{OPT}$ as the optimal k -MEDIAN solution is also a feasible solution for FACILITY LOCATION with at most k facilities; note that the choice of \mathfrak{f} balances the opening and connection costs in this solution. We then run (any) γ -approximation algorithm for FACILITY LOCATION, and it will find a solution whose cost is at most $\gamma \cdot 2 \text{OPT}$. The choice of \mathfrak{f} implies that the number of open facilities in this solution is at most $\gamma \cdot 2 \text{OPT} / \mathfrak{f} = 2\gamma \cdot k$. Hence, we obtain a $(2\gamma, 2\gamma)$ -bicriteria approximation of k -MEDIAN. Finally, we remove the assumption of knowing OPT by running this procedure in parallel for a logarithmic number of guesses for OPT , and taking the cheapest solution that uses at most $O(\gamma \cdot k)$ centers. Using our MPC algorithm for FACILITY LOCATION, this gives an efficient MPC algorithm for clustering with $(O(\gamma), O(\gamma))$ -bicriteria approximation guarantees.

In this approach, the number of centers (in the solution) might exceed k by a large constant factor. We now use a different method to reduce it to be arbitrarily close to k . First, we observe that the $(O(\gamma), O(\gamma))$ -approximate solution can be used to get a *weak coreset*, namely, a weighted set of at most $O(\gamma \cdot k)$ distinct points, such that any α -approximate set of centers for the coreset is also an $O(\alpha \cdot \gamma)$ -approximate solution for the original instance. To obtain the coreset, we just move every point to its approximately nearest facility in the approximate solution, which is a standard step in the clustering literature (see e.g. [42]). Then, weight $w(p)$ of a point p is the number of original points rounded to p . We note that the problem becomes trivial if the coreset fits in one machine, and thus the interesting case is when k is very large.

Given this weak coreset, we find an approximate solution with at most, say, $2k$ centers and cost increased by another factor of $O(\gamma)$, using the following simple but sequential algorithm: Process the coreset points in order of non-increasing weights $w(\cdot)$, and open a center at point p if so far there is no center within distance $\text{OPT} / (k \cdot w(p))$ from p ; here we again need a guess for OPT . In a nutshell, the analysis of this algorithm is based on averaging arguments; intuitively, only few points in the weak coreset can have a relatively large connection cost in the optimal solution.

We then convert this sequential algorithm into a parallel one using similar ideas as for FACILITY LOCATION. That is, for every point in parallel, we add it to the set of centers using two separate rules: (i) independently with probability $1/\gamma$; or (ii) if there is no point with larger weight within distance $\text{OPT} / (k \cdot w(p))$. To implement (ii) in MPC, we need to ensure a consistent tie-breaking, for which a small random perturbation of the weights is sufficient. Finally, to efficiently find the point of maximum (perturbed) weight in the neighborhood of every point, we again employ our MPC primitive for geometric aggregation.

1.3 Related Work

Parallel and Distributed Algorithms. A more general metric setting of FACILITY LOCATION has been studied earlier in the distributed Congest and CongestedClique models (see, e.g., [38, 44, 45]), and these results immediately transfer into MPC algorithms with $O(n)$ local memory and $O(n^2)$ total space. In particular, in combination with the recent result in [20], these results yield an $O(1)$ -round MPC algorithm for metric FACILITY LOCATION. In this general-metric setting of FACILITY LOCATION, instances have size $O(n^2)$, which makes the problem significantly different from our geometric setting, e.g., instances in \mathbb{R}^d are trivial if the local memory is $O(n)$. Furthermore, those results for general metrics rely on computing $O(1)$ -ruling sets, and by the conditional lower bounds in [40, 31], this seems to require $\omega(1)$ rounds on a fully-scalable MPC. Our algorithms bypass the obstacle of ruling sets by leveraging the geometric structure in \mathbb{R}^d and one can view our high-level contribution as proposing a setting avoiding that obstacle.

Clustering problems (e.g., FACILITY LOCATION, k -MEANS, k -MEDIAN) have been also studied in the PRAM model of parallel computation [16, 15]. These algorithms can be implemented in the MPC model, but the logarithmic factors in the running time or the approximation ratio seem inherent, and they fall short of achieving $O(1)$ rounds in the fully-scalable regime.

Connections to Streaming. A closely related model is the streaming model, which mainly focuses on sequential processing of large datasets by a single machine with a limited (sublinear) memory. In general, if a streaming algorithm is storing only a linear sketch and uses space $O(s^{1-\varepsilon})$ for a fixed $\varepsilon > 0$, then it can be simulated on MPC in $O(\log_s N)$ rounds (recall

s is the local memory per machine and N is the input size); this was also observed and mentioned in, e.g., [24]. Thus, the various recent results on streaming algorithms in high dimension can be readily applied in MPC, and we briefly discuss the most relevant ones in the following. We note that streaming algorithms typically assume the input is discrete, i.e., $P \subseteq [\Delta]^d$ with $\Delta = \text{poly}(n)$.

For both k -MEDIAN [18] and k -MEANS [63], it is possible to find $(1 + \varepsilon)$ -approximate solution with k centers using space $\text{poly}(\varepsilon^{-1}kd \log \Delta)$. However, these algorithms are not directly applicable in our setting, since to simulate these algorithms it requires local memory size $s = \Omega(k)$ which is not fully scalable. For FACILITY LOCATION, [49] gave $O(d \log \Delta)$ -approximation (along with several other problems including minimum spanning tree and matching), using space $\text{poly}(d \log \Delta)$. Later on, an $O(d/\log d)$ -approximation for FACILITY LOCATION was obtained using similar space $\text{poly}(d \log \Delta)$, and alternatively $O(1/\varepsilon)$ -approximation using space $O(n^\varepsilon)$ [33]. However, these results for FACILITY LOCATION can only estimate the optimal cost, as storing the solution requires linear space (which is too costly since streaming algorithms aim to use sublinear space). Hence, simulating these results only leads to estimating the optimal cost in MPC, while our result for FACILITY LOCATION can indeed find an approximate solution.

MPC Algorithms for MST. A similar issue of cost estimation versus finding approximate solutions is present also for the minimum spanning tree (MST) problem. The classical streaming algorithm of [49] was recently improved to an $O(\log n)$ -approximation using the same space regime $\text{poly}(d \log \Delta)$ [25] and to an $O(1/\varepsilon^2)$ -approximation for the n^ε space regime [24]. However, these results are for estimating the optimal MST cost, and it is still open to find an $O(1)$ -approximate solution for high-dimensional MST in $O(\log_s n)$ rounds of MPC. Indeed, the currently best MPC algorithm finds an $O(1)$ -approximate MST in $\tilde{O}(\log \log n)$ rounds [52] (using local space polynomial in n), while in $O(\log_s n)$ rounds, one can only find an $O(1)$ -approximation in a low dimension [4] (see also [22] for an exact MST algorithm for $d = 2$) or a $\text{poly}(\log n)$ -approximation in a high dimension [2]. For a related problem of single-linkage clustering in a low dimension, there is a $(1 + \varepsilon)$ -approximation on MPC in $O(\log n)$ rounds [65].

2 Preliminaries

For integer n , let $[n] := \{1, 2, \dots, n\}$. For a function $\varphi : X \rightarrow Y$, the image of a subset $S \subseteq X$ is defined $\varphi(S) := \{\varphi(x) : x \in S\}$, and the preimage of $y \in Y$ is defined as $\varphi^{-1}(y) := \{x \in X : \varphi(x) = y\}$. A Euclidean ball centered at $x \in \mathbb{R}^d$ with radius $r \geq 0$ is defined as $B(x, r) := \{y \in \mathbb{R}^d : \text{dist}(x, y) \leq r\}$, where $\text{dist}(x, y) := \|x - y\|_2$ refers throughout to Euclidean distance. For a set $P \subset \mathbb{R}^d$, we define $B_P(x, r) := B(x, r) \cap P$, which is also a metric ball inside P . Let $\text{diam}(S)$ denote the diameter of $S \subseteq \mathbb{R}^d$. For two sequences S, T , denote their concatenation by $S \circ T$. The aspect ratio of a point set $S \subset \mathbb{R}^d$ is the ratio between the maximum and minimum inter-point distance of S , i.e., $\frac{\max_{x, y \in S} \text{dist}(x, y)}{\min_{x \neq y \in S} \text{dist}(x, y)}$.

► **Fact 2.1** (Generalized triangle inequality). *Let (V, ρ) be a metric space, and let $z \geq 1$. Then*

$$\forall x, x', y \in V, \quad \rho^z(x, y) \leq 2^{z-1}(\rho^z(x, x') + \rho^z(x', y)).$$

Power- z (Uniform) Facility Location. Given a set of data points $P \subset \mathbb{R}^d$, a (uniform) opening cost $\mathfrak{f} > 0$ and some $z \geq 1$, the objective function of POWER- z (UNIFORM) FACILITY LOCATION for a set of facilities $F \subset \mathbb{R}^d$ is defined as

$$\mathfrak{fl}_z(P, F) := |F| \cdot \mathfrak{f} + \sum_{p \in P} \text{dist}^z(p, F),$$

where again $\text{dist}(x, y) := \|x - y\|_2$ and $\text{dist}(x, S) := \min_{y \in S} \text{dist}(x, y)$. From now on, we omit “uniform” from the name of the problem, and simply use POWER- z FACILITY LOCATION. We denote the minimum value of a solution for POWER- z FACILITY LOCATION by $\text{OPT}_z^{\text{fl}}(P) := \min_{F \subseteq \mathbb{R}^d} \text{fl}_z(P, F)$; we omit P if it is clear from the context.

(k, z) -Clustering. Given a set of data points $P \subset \mathbb{R}^d$, an integer $k \geq 1$ and some $z \geq 1$, the objective function of (k, z) -CLUSTERING for a center set $C \subset \mathbb{R}^d$ with $|C| \leq k$ is defined as

$$\text{cl}_z(P, C) := \sum_{p \in P} \text{dist}^z(p, C).$$

Notice that the special cases $z = 1$ and $z = 2$ are called k -MEDIAN and k -MEANS, respectively. We denote the minimum value of a solution for (k, z) -CLUSTERING by $\text{OPT}_z^{\text{cl}}(P) := \min_{C \subseteq \mathbb{R}^d: |C| \leq k} \text{cl}_z(P, C)$; we again omit P when it is clear from the context.

Consistent Hashing. As mentioned above, our MPC primitive for geometric aggregation relies on *consistent hashing*. We define it below, and then state the best known bounds for its parameters, which are near-optimal [36].

► **Definition 2.2** ([33, Definition 1.6]). *A mapping $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is called a Γ -gap Λ -consistent hash with diameter bound $\ell > 0$, or simply (Γ, Λ) -hash,³ if it satisfies:*

1. *Diameter: for every image $z \in \varphi(\mathbb{R}^d)$, we have $\text{diam}(\varphi^{-1}(z)) \leq \ell$; and*
2. *Consistency: for every $S \subseteq \mathbb{R}^d$ with $\text{diam}(S) \leq \ell/\Gamma$, we have $|\varphi(S)| \leq \Lambda$.*

► **Lemma 2.3** ([33, Theorem 5.1]). *For every $\Gamma \in [8, 2d]$, there exists a (deterministic) (Γ, Λ) -hash $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}^d$ where $\Lambda = \exp(8d/\Gamma) \cdot O(d \log d)$. Furthermore, φ can be described using $O(d^2 \log^2 d)$ bits and one can evaluate $\varphi(x)$ for any point $x \in \mathbb{R}^d$ in space $O(d^2 \log^2 d)$.*

► **Remark 2.4.** Our main results also hold under the assumption that $s \geq \text{polylog}(dn)$, which is preferable when $d \gg \log n$. It follows by the well-known idea of applying a JL transform π (named after Johnson and Lindenstrauss [54]) with target dimension $O(\log n)$ as a preprocessing of the input $P \subset \mathbb{R}^d$, i.e., running our algorithm on $\pi(P)$ instead of P . Then, with probability $1 - 1/\text{poly}(n)$, all the guarantees in our results would suffer only an $O(1)$ factor. To implement this preprocessing in MPC using $\text{polylog}(dn)$ words of local memory, we use a bounded-space version of the JL transform [48], that requires only $\text{polylog}(dn)$ words to specify π (in comparison, a naive implementation of π requires $O(d \log n)$ words). One machine can randomly generate this specification of π and broadcast it, and then all machines can apply π locally in parallel. This requires additional $O(\text{polylog}(dn))$ local memory and $O(\log_s n)$ rounds, and these additional costs are easily absorbed in our bounds. Hence, in all our results we can assume without loss of generality that P is replaced by $\pi(P)$. Furthermore, after this preprocessing, we can use $d = O(\log n)$ also in the consistent hashing bounds in Lemma 2.3 to obtain a tradeoff $\Gamma = O(1/\varepsilon)$ and $\Lambda = O(n^\varepsilon)$, for any desired fixed $\varepsilon \in (0, 1)$, and also the hash can be described using $\text{polylog}(n)$ bits.

3 MPC Primitive for Geometric Aggregation in High Dimension

In our MPC algorithms, we often face a scenario where we want to compute something for each input point $p \in P$. That computation is a relatively simple problem, like computing the number of points in the ball $B_P(p, r)$ for some global value $r > 0$. A more general

³ Note that this definition is scale invariant with respect to ℓ , i.e., a scaling of \mathbb{R}^d will scale ℓ but not affect the parameters Γ and Λ . Thus, upper and lower bounds can restrict attention to the case $\ell = 1$.

version is to allow different radii r (local for each p); another generalization is to compute some function f over the points in $B_P(p, r)$, like finding the point with smallest identifier or smallest distance to p . A naive approach is to collect (copies of) all the points in $B_P(p, r)$ to the same machine, say the one holding p , and then compute f there. This is very challenging and our solution is to approximate these balls by generating sets $A_P(p, r) \approx B_P(p, r)$, and evaluate f on these sets instead of on the balls. The approximation here just means that the set $A_P(p, r)$ is sandwiched between a ball of radius r and one of larger radius, see (1). Informally, we thus compute $f(A_P(p, r)) \approx f(B_P(p, r))$, but of course the approximation here need not be multiplicative.

Our MPC algorithm derives these sets $A_P(p, r)$ from consistent hashing (see Definition 2.2), and thus our description below requires access to such a hash function φ , and moreover the final guarantees depend on the parameters Γ and Λ of the consistent hashing. The theorem below is stated in general, i.e., for any (Γ, Λ) -hash, but we eventually employ known constructions with $\Gamma = O(1/\varepsilon)$ and $\Lambda = n^\varepsilon$, for any desired $\varepsilon > 0$ (see Remark 2.4 and Lemma 2.3 for details). Obviously, running this algorithm in MPC requires an implementation of consistent hashing, which might require additional memory; but this memory requirement is typically much smaller than \sqrt{s} , and thus the hash function can be easily stored in each machine.

Yet another challenge is that the entire set $A_P(p, r)$ might not fit in a single machine, and we thus impose on f another requirement. We say that a function f is *composable* if for every disjoint $S_1, \dots, S_t \subseteq \mathbb{R}^d$, one can evaluate $f(S_1 \cup \dots \cup S_t)$ from the values of $f(S_1), \dots, f(S_t)$.⁴ In our context, f maps finite subsets of \mathbb{R}^d to \mathbb{R} . For instance, $f(S) = |S|$ is clearly composable. For a few more interesting examples, suppose every $x \in \mathbb{R}^d$ is associated with a value $h(x) \in \mathbb{R}$. Now if $h(x)$ represents the weight of x , then $f(S) = \sum_{x \in S} h(x)$ is the total weight of S ; and if $h(x)$ represents an identifier (perhaps chosen at random), then $f(S) = \min_{x \in S} h(x)$ is the smallest identifier in S .

► **Theorem 3.1** (Geometric Aggregation in MPC). *There is a deterministic fully-scalable MPC algorithm with the following guarantees. Suppose that*

- *the input is $r \geq 0$ and a multiset $P \subset \mathbb{R}^d$ of n points distributed across machines with local memory $s \geq \text{poly}(d \log n)$; and*
- *the algorithm has access to a composable function f (mapping finite subsets of \mathbb{R}^d to \mathbb{R}) and to a (Γ, Λ) -hash $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}^d$.*

Then the algorithm uses $O(\log_s n)$ rounds and $O(\Lambda \cdot \text{poly}(d)) \cdot \tilde{O}(n)$ total space, and computes for each $p \in P$ a value $f(A_P(p, r))$, where $A_P(p, r)$ is an arbitrary set that satisfies

$$B_P(p, r) \subseteq A_P(p, r) \subseteq B_P(p, 3\Gamma \cdot r). \quad (1)$$

(In fact, the set $A_P(p, r)$ is determined by φ .)

Proof. Our algorithm makes use of the following standard subroutines in MPC, and we note that they are deterministic. In the broadcast procedure, to send a message of length at most \sqrt{s} from some machine \mathcal{M}_0 to every other ones, one can build an \sqrt{s} -ary broadcasting tree whose nodes are the machines (with \mathcal{M}_0 as the root), and send/replicate the message level-by-level through the tree (starting from the root). Observe that the height of the tree is $O(\log_s n)$ and hence the entire process runs in $O(\log_s n)$ rounds. The reversed procedure defined on the same broadcast tree, sometimes called converge-cast [39], can be used to

⁴ We use general t here because of our intended application, but obviously it follows from the special case $t = 2$.

aggregate messages with size at most \sqrt{s} distributed across machines to some root machine \mathcal{M}_0 (for instance, to aggregate the sum of vectors of length \sqrt{s} distributed across machines), in $O(\log_s n)$ rounds. In particular, it can be used to evaluate the composable function f on a (distributed) set S , where each machine \mathcal{M} evaluates $f(S_{\mathcal{M}})$ for its own part $S_{\mathcal{M}} \subseteq S$, and aggregate using converge-cast.

► **Lemma 3.2** (Sorting in MPC [41]). *There is a deterministic MPC algorithm that given a set X of N comparable items distributed across machines with local memory s , sorts X such that each $x \in X$ knows its rank and $\forall x < y \in X$, it holds that the machine that holds x has an ID no larger than that of y . The algorithm uses $O(\log_s N)$ rounds and total space of $O(N)$ words.*

We give an outline of our algorithm in Algorithm 1; the implementation details in MPC are discussed below. The algorithm starts with “partitioning” \mathbb{R}^d into buckets with respect to the (Γ, Λ) -hash φ , and approximates each ball $B(p, r)$ by the union of buckets that this ball intersects. This distorts the radius by at most an $O(\Gamma)$ -factor. Then, we evaluate the f value on each bucket, and the approximation to $f(B_P(p, r))$ is obtained by “aggregating” the f value for the intersecting buckets of $B(p, r)$, where the composability of f is crucially used.

Implementation Details. Here we discuss how each step of Algorithm 1 is implemented efficiently in MPC. In line 3, since after the sorting, for each $u \in \varphi(P)$ the points in P_u , i.e., the set of points p such that $u = \varphi(p)$, span a (partial) segment of machines with contiguous IDs, one can use a converge-cast in parallel in each segment to aggregate $f(P_u)$. In line 4, although the total space is sufficient to hold all tuples, we may not have enough space to store the $O(\Lambda)$ tuples for a point p in a single machine. Instead, we allocate for every point p a (partial) segment of machines whose total space is $O(\Lambda)$ (which can be figured out via sorting), replicate p 's to them (via broadcast), and generate $\varphi(B(p, r))$ in parallel on those machines. Specifically, each machine in the segment is responsible for generating a part of $\varphi(B(p, r))$, and a part can be generated locally without further communication since every machine holds the same deterministic φ . Line 6 and line 7 can be implemented similarly by broadcast and converge-cast, respectively, in parallel on each segment of machines.

■ **Algorithm 1** MPC algorithm for evaluating $f(A_P(p, r))$ for $p \in P$, for given $P \subset \mathbb{R}^d, r > 0$.

- 1: each machine imposes the same (Γ, Λ) -hash $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}^d$ with diameter bound $\ell := 2\Gamma r$
 ▷ notice that φ is deterministic, hence no communication is required
- 2: sort P with respect to $\varphi(p)$ for $p \in P$ (using Lemma 3.2)
- 3: for $u \in \varphi(P)$, evaluate and store $f(P_u)$ where $P_u := \varphi^{-1}(u) \cap P$
- 4: for each $p \in P$ and $u \in \varphi(B(p, r))$, create and store a tuple (p, u)
 ▷ as $|\varphi(B(p, r))| = O(\Lambda)$ by Definition 2.2, the total space is enough to hold all tuples
- 5: sort the tuples with respect to u (using Lemma 3.2)
- 6: let $\mathcal{T}_u = \{(\cdot, u)\}$, append $f(P_u)$ to all tuples in \mathcal{T}_u
 ▷ $f(P_u)$ is already evaluated and stored, as in line 3
- 7: sort the tuples with respect to p , and evaluate $f(A_P(p, r))$ for each p , where

$$A_P(p, r) := \bigcup_{u \in \varphi(B(p, r))} P_u$$

Round Complexity and Total Space. The round complexity is dominated by the sorting, broadcast and converge-cast procedures, which all take $O(\log_s n)$ rounds to finish and are invoked $O(1)$ times in total. Therefore, the algorithm runs in $O(\log_s n)$ rounds. The total space is asymptotically dominated by $\text{poly}(d \log n)$ times the total number of tuples, which is $O(\Lambda) \cdot n$ by Definition 2.2.

Correctness. Observe that the algorithm is deterministic, and hence there is no failure probability. It remains to show that $A_P(p, r)$ satisfies that $B_P(p, r) \subseteq A_P(p, r) \subseteq B_P(p, 3\Gamma r)$. Recall that $P_u = \varphi^{-1}(u) \cap P$ as defined in line 3, and that $A_P(p, r) = \bigcup_{u \in \varphi(B(p, r))} P_u$ as in line 7. Hence, we have

$$A_P(p, r) = P \cap \varphi^{-1}(\varphi(B(p, r))).$$

Therefore, the first inequality is straightforward as $B_P(p, r) \subseteq \varphi^{-1}(\varphi(B(p, r))) \cap P = A_P(p, r)$ for any mapping φ .

To prove the second inequality, fix some $p \in P$. For a point $q \in A_P(p, r)$, by definition there is a $u_q \in \varphi(B_P(p, r))$ such that $q \in \varphi^{-1}(u_q)$. Then by Definition 2.2, we have $\text{diam}(\varphi^{-1}(u_q)) \leq \ell = 2\Gamma r$ which implies that any point $x \in \varphi^{-1}(u_q)$ satisfies $\text{dist}(x, q) \leq 2\Gamma r$. Now, pick a point $x \in B_P(p, r)$ such that $\varphi(x) = u_q$; such a point exists as $u_q \in \varphi(B_P(p, r))$. Then by definition, $x \in B_P(p, r) \cap \varphi^{-1}(u_q)$ and $\text{dist}(p, x) \leq r$. Hence, by triangle inequality, we have that $\text{dist}(p, q) \leq \text{dist}(p, x) + \text{dist}(x, q) \leq r + 2\Gamma r \leq 3\Gamma r$, which implies that $A_P(p, r) \subseteq B_P(p, 3\Gamma r)$. This finishes the proof. \blacktriangleleft

3.1 Application to Nearest Neighbor Search

Given a set $X \subseteq \mathbb{R}^d$ of *terminals* and a set $P \subseteq \mathbb{R}^d$ of data points, the ρ -approximate nearest neighbor search problem asks to find for every $p \in P$ a terminal $x \in X$, such that $\text{dist}(p, x) \leq \rho \cdot \text{dist}(p, X)$. This process is useful in clustering and facility location problems, since one can find an assignment of every data point to its approximately nearest center/facility. We show how to solve this problem using Theorem 3.1, provided the knowledge of the *aspect ratio* Δ of $X \cup P$.

Pick an arbitrary point $x \in X \cup P$, compute in $O(\log_s n)$ rounds the maximum distance $M := \max_{y \in X \cup P} \text{dist}(x, y)$ from x to every other point (via broadcast and converge-cast). Since M is a 2-approximation to $\text{diam}(X \cup P)$, we conclude that for every $x \neq y \in X \cup P$, $M/\Delta \leq \text{dist}(x, y) \leq 2M$. Rescale the instance by dividing M/Δ , then the distances are between 1 and $O(\Delta)$. Then, let $Z := \{2^i : 1 \leq 2^i \leq O(\Delta)\}$. We apply Theorem 3.1 in parallel for $r \in Z$ and f such that $f(Y)$ for $Y \subseteq X$ finds the terminal with the smallest ID in Y (where the ID of a point can be defined arbitrarily as long as it is consistent), and f returns \perp if $Y = \emptyset$. This f is clearly composable. After we obtain the result of Theorem 3.1, i.e., $f(A_X(p, r))$ for $p \in P$ and $r \in Z$, we find in parallel for each $p \in P$ the smallest $r \in Z$ such that $f(A_X(p, r)) \neq \perp$. This way, we explicitly get for each point p an approximately nearest facility in X . This algorithm has approximation factor $\rho = O(\Gamma)$, using total space by an $O(\log \Delta)$ -factor larger than that of Theorem 3.1, while the round complexity remains $O(\log_s n)$.

We remark that techniques based on locality sensitive hashing (LSH) can also be applied in MPC to solve the approximate nearest neighbor problem [13, 28]. LSH in fact achieves a slightly better tradeoff, namely, an $O(c)$ -approximation using total space $n^{1/c^2} \cdot \tilde{O}(n)$, while our approach requires total space $n^{1/c} \cdot \tilde{O}(n)$, by plugging in Lemma 2.3 and assuming the dimension reduction in Remark 2.4 is performed. Alternatively, if $d \leq O(\log n)$, one can

obtain the same $n^{1/c} \cdot \tilde{O}(n)$ bound without applying the randomized dimension reduction in Remark 2.4, which leads to a deterministic algorithm, whereas approaches based on LSH are inherently randomized.

References

- 1 Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for k -means clustering. In *Proceedings of the 12th International Workshop on Approximation Algorithms for Combinatorial Optimization, and of the 13th International Workshop on Randomization and Approximation Techniques in Computer Science (APPROX/RANDOM)*, pages 15–28, 2009.
- 2 AmirMohsen Ahanchi, Alexandr Andoni, MohammadTaghi Hajiaghayi, Marina Knittel, and Peilin Zhong. Massively parallel tree embeddings for high dimensional spaces. In *Proceedings of the 35th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 77–88, 2023. doi:10.1145/3558481.3591096.
- 3 Hyung-Chan An, Ashkan Norouzi-Fard, and Ola Svensson. Dynamic facility location via exponential clocks. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 708–721, 2015.
- 4 Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 574–583, 2014. doi:10.1145/2591796.2591805.
- 5 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k -median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.
- 6 Olivier Bachem, Mario Lucic, and Andreas Krause. Scalable k -means clustering via lightweight coresets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 1119–1127, 2018.
- 7 Mihai Bădoiu, Artur Czumaj, Piotr Indyk, and Christian Sohler. Facility location in sublinear time. In *Proceedings of the 32nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 866–877, 2005. doi:10.1007/11523468_70.
- 8 Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable K -Means++. *Proc. VLDB Endow.*, 5(7):622–633, 2012.
- 9 Maria-Florina Balcan, Steven Ehrlich, and Yingyu Liang. Distributed k -means and k -median clustering on general communication topologies. In *NIPS*, pages 1995–2003, 2013.
- 10 MohammadHossein Bateni, Hossein Esfandiari, Manuela Fischer, and Vahab S. Mirrokni. Extreme k -center clustering. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pages 3941–3949, 2021. doi:10.1609/aaai.v35i5.16513.
- 11 Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. *Journal of the ACM*, 64(6):40:1–40:58, 2017. doi:10.1145/3125644.
- 12 Soheil Behnezhad, Moses Charikar, Weiyun Ma, and Li-Yang Tan. Almost 3-approximate correlation clustering in constant rounds. In *Proceedings of the 63rd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 720–731, 2022. doi:10.1109/FOCS54457.2022.00074.
- 13 Aditya Bhaskara and Maheshakya Wijewardena. Distributed clustering via LSH based data partitioning. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 569–578. PMLR, 2018. URL: <https://proceedings.mlr.press/v80/bhaskara18a.html>.
- 14 Yonatan Bilu and Nathan Linial. Are stable instances easy? *Combinatorics, Probability & Computing*, 21(5):643–660, 2012. doi:10.1017/S0963548312000193.
- 15 Guy E. Blelloch, Anupam Gupta, and Kanat Tangwongsan. Parallel probabilistic tree embeddings, k -me network design. In *Proceedings of the 24th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 205–213, 2012.

- 16 Guy E. Blelloch and Kanat Tangwongsan. Parallel approximation algorithms for facility-location problems. In *Proceedings of the 22nd Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 315–324, 2010. doi:10.1145/1810479.1810535.
- 17 Vladimir Braverman, Vincent Cohen-Addad, Shaofeng H.-C. Jiang, Robert Krauthgamer, Chris Schwiegelshohn, Mads Bech Tofttrup, and Xuan Wu. The power of uniform sampling for coresets. In *Proceedings of the 63rd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 462–473, 2022. doi:10.1109/FOCS54457.2022.00051.
- 18 Vladimir Braverman, Gereon Frahling, Harry Lang, Christian Sohler, and Lin F. Yang. Clustering high dimensional dynamic data streams. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 576–585. PMLR, 2017.
- 19 Jaroslaw Byrka and Karen I. Aardal. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *SIAM Journal on Computing*, 39(6):2212–2231, 2010.
- 20 Mélanie Cambus, Fabian Kuhn, Shreyas Pai, and Jara Uitto. Time and space optimal massively parallel algorithm for the 2-ruling set problem. In *Proceedings of the 37th International Symposium on Distributed Computing (DISC)*, pages 11:1–11:12, 2023. doi:10.4230/LIPICS.DISC.2023.11.
- 21 Matteo Ceccarello, Andrea Pietracaprina, and Geppino Pucci. Solving k -center clustering (with outliers) in MapReduce and streaming, almost as accurately as sequentially. *Proc. VLDB Endow.*, 12(7):766–778, 2019.
- 22 Yi-Jun Chang and Da Wei Zheng. Fully scalable massively parallel algorithms for embedded planar graphs. In *Proceedings of the 35th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4410–4450, 2024. doi:10.1137/1.9781611977912.155.
- 23 Jiecao Chen, He Sun, David P. Woodruff, and Qin Zhang. Communication-optimal distributed clustering. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 3720–3728, 2016. URL: <https://proceedings.neurips.cc/paper/2016/hash/7503cfacd12053d309b6bed5c89de212-Abstract.html>.
- 24 Xi Chen, Vincent Cohen-Addad, Rajesh Jayaram, Amit Levi, and Erik Waingarten. Streaming Euclidean MST to a Constant Factor. In *Proceedings of the 55th Annual Symposium on Theory of Computing (STOC)*, pages 156–169, 2023. doi:10.1145/3564246.3585168.
- 25 Xi Chen, Rajesh Jayaram, Amit Levi, and Erik Waingarten. New streaming algorithms for high dimensional EMD and MST. In *Proceedings of the 54th Annual Symposium on Theory of Computing (STOC)*, pages 222–233, 2022. doi:10.1145/3519935.3519979.
- 26 Vincent Cohen-Addad, Kasper Green Larsen, David Saulpic, and Chris Schwiegelshohn. Towards optimal lower bounds for k -median and k -means coresets. In *Proceedings of the 54th Annual Symposium on Theory of Computing (STOC)*, pages 1038–1051, 2022. doi:10.1145/3519935.3519946.
- 27 Vincent Cohen-Addad, Silvio Lattanzi, Ashkan Norouzi-Fard, Christian Sohler, and Ola Svensson. Parallel and efficient hierarchical k -median clustering. In *NeurIPS*, pages 20333–20345, 2021. URL: <https://proceedings.neurips.cc/paper/2021/hash/aa495e18c7e3a21a4e48923b92048a61-Abstract.html>.
- 28 Vincent Cohen-Addad, Vahab S. Mirrokni, and Peilin Zhong. Massively parallel k -means clustering for perturbation resilient instances. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 4180–4201. PMLR, 2022. URL: <https://proceedings.mlr.press/v162/cohen-addad22b.html>.
- 29 Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. In *Proceedings of the 53rd Annual Symposium on Theory of Computing (STOC)*, pages 169–182, 2021.
- 30 Sam Coy, Artur Czumaj, and Gopinath Mishra. On parallel k -center clustering. In *Proceedings of the 35th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 65–75, 2023. doi:10.1145/3558481.3591075.

- 31 Artur Czumaj, Peter Davies, and Merav Parter. Component stability in low-space massively parallel computation. In *Proceedings of the 40th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 481–491, 2021. doi:10.1145/3465084.3467903.
- 32 Artur Czumaj, Guichen Gao, Shaofeng H.-C. Jiang, Robert Krauthgamer, and Pavel Veselý. Fully scalable MPC algorithms for clustering in high dimension. *CoRR*, abs/2307.07848, 2023. doi:10.48550/arXiv.2307.07848.
- 33 Artur Czumaj, Shaofeng H.-C. Jiang, Robert Krauthgamer, Pavel Veselý, and Mingwei Yang. Streaming facility location in high dimension via geometric hashing. In *Proceedings of the 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 450–461, 2022. The latest version can be found at arXiv:2204.02095 and it has additional results. doi:10.1109/FOCS54457.2022.00050.
- 34 Mark de Berg, Leyla Biabani, and Morteza Monemizadeh. k -center clustering with outliers in the MPC and streaming model. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium, (IPDPS)*, pages 853–863, 2023. doi:10.1109/IPDPS54959.2023.00090.
- 35 Alina Ene, Sungjin Im, and Benjamin Moseley. Fast clustering using MapReduce. In *17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '11*, pages 681–689. ACM, 2011. doi:10.1145/2020408.2020515.
- 36 Arnold Filtser. Scattering and sparse partitions, and their applications. In *47th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 47:1–47:20, 2020. doi:10.4230/LIPIcs.ICALP.2020.47.
- 37 Naveen Garg. A 3-approximation for the minimum tree spanning k vertices. In *Proceedings of the 37th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 302–309, 1996.
- 38 Joachim Gehweiler, Christiane Lammersen, and Christian Sohler. A distributed $O(1)$ -approximation algorithm for the uniform facility location problem. *Algorithmica*, 68(3):643–670, 2014. doi:10.1007/s00453-012-9690-y.
- 39 Mohsen Ghaffari. Massively parallel algorithms, 2019. Lecture Notes from ETH Zurich. URL: <http://people.csail.mit.edu/ghaffari/MPA19/Notes/MPA.pdf>.
- 40 Mohsen Ghaffari, Fabian Kuhn, and Jara Uitto. Conditional hardness results for massively parallel computation from distributed lower bounds. In *Proceedings of the 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1650–1663, 2019. doi:10.1109/FOCS.2019.00097.
- 41 Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the MapReduce framework. In *Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC)*, pages 374–383, 2011. doi:10.1007/978-3-642-25591-5_39.
- 42 Sudipto Guha, Nina Mishra, Rajeev Motwani, and Liadan O’Callaghan. Clustering data streams. In *Proceedings of the 41st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 359–366, 2000.
- 43 Sariel Har-Peled and Soham Mazumdar. On coresets for k -means and k -median clustering. In *Proceedings of the 36th Annual Symposium on Theory of Computing (STOC)*, pages 291–300, 2004. doi:10.1145/1007352.1007400.
- 44 James W. Hegeman and Sriram V. Pemmaraju. Sub-logarithmic distributed algorithms for metric facility location. *Distributed Computing*, 28(5):351–374, 2015. doi:10.1007/s00446-015-0243-x.
- 45 James W. Hegeman, Sriram V. Pemmaraju, and Vivek Sardeshmukh. Near-constant-time distributed algorithms on a congested clique. In *Proceedings of the 28th International Symposium on Distributed Computing (DISC)*, pages 514–530, 2014. doi:10.1007/978-3-662-45174-8_35.
- 46 Daniel J. Hsu and Matus Telgarsky. Greedy bi-criteria approximations for k -medians and k -means. *CoRR*, abs/1607.06203, 2016. arXiv:1607.06203.

- 47 Sungjin Im, Ravi Kumar, Silvio Lattanzi, Benjamin Moseley, and Sergei Vassilvitskii. Massively parallel computation: Algorithms and applications. *Foundations and Trends in Optimization*, 5(4):340–417, 2023.
- 48 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *Proceedings of the 41st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 189–197, 2000. doi:10.1109/SFCS.2000.892082.
- 49 Piotr Indyk. Algorithms for dynamic geometric problems over data streams. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 373–380, 2004. doi:10.1145/1007352.1007413.
- 50 Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, 50(6):795–824, 2003. doi:10.1145/950620.950621.
- 51 Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k -median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM*, 48(2):274–296, 2001. doi:10.1145/375827.375845.
- 52 Rajesh Jayaram, Vahab Mirrokni, Shyam Narayanan, and Peilin Zhong. Massively parallel algorithms for high-dimensional Euclidean minimum spanning tree. In *Proceedings of the 35th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3960–3996, 2024. doi:10.1137/1.9781611977912.139.
- 53 Lujun Jia, Guolong Lin, Guevara Noubir, Rajmohan Rajaraman, and Ravi Sundaram. Universal approximations for TSP, Steiner tree, and set cover. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 386–395, 2005. doi:10.1145/1060590.1060649.
- 54 W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Proceedings of the Conference in Modern Analysis and Probability (New Haven, Connecticut, 1982)*, pages 189–206. American Mathematical Society, 1984.
- 55 Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for MapReduce. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 938–948, 2010. doi:10.1137/1.9781611973075.76.
- 56 Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Inf. Comput.*, 222:45–58, 2013.
- 57 Jyh-Han Lin and Jeffrey Scott Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, 44(5):245–249, 1992.
- 58 Jyh-Han Lin and Jeffrey Scott Vitter. ϵ -approximations with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC)*, pages 771–782, 1992. doi:10.1145/129712.129787.
- 59 Konstantin Makarychev, Yury Makarychev, Maxim Sviridenko, and Justin Ward. A bi-criteria approximation algorithm for k -means. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 14:1–14:20, 2016. doi:10.4230/LIPICS.APPROX-RANDOM.2016.14.
- 60 Gustavo Malkomes, Matt J. Kusner, Wenlin Chen, Kilian Q. Weinberger, and Benjamin Moseley. Fast distributed k -center clustering with outliers on massive data. In *NIPS*, pages 1063–1071, 2015.
- 61 Ramgopal R. Mettu and C. Greg Plaxton. The online median problem. *SIAM Journal on Computing*, 32(3):816–832, 2003. doi:10.1137/S0097539701383443.
- 62 Tim Roughgarden, Sergei Vassilvitskii, and Joshua R. Wang. Shuffles and circuits (On lower bounds for modern parallel computation). *Journal of the ACM*, 65(6):41:1–41:24, 2018. doi:10.1145/3232536.
- 63 Zhao Song, Lin F. Yang, and Peilin Zhong. Sensitivity sampling over dynamic geometric data streams with applications to k -clustering. *CoRR*, abs/1802.00459, 2018. arXiv:1802.00459.

50:20 Fully-Scalable MPC Algorithms for Clustering in High Dimension

- 64 Dennis Wei. A constant-factor bi-criteria approximation guarantee for k -means++. In *NIPS*, volume 29, pages 604–612, 2016. URL: <https://proceedings.neurips.cc/paper/2016/hash/357a6fdf7642bf815a88822c447d9dc4-Abstract.html>.
- 65 Grigory Yaroslavtsev and Adithya Vadapalli. Massively parallel algorithms and hardness for single-linkage clustering under ℓ_p distances. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 5596–5605. PMLR, 2018. URL: <http://proceedings.mlr.press/v80/yaroslavtsev18a.html>.