Fully Scalable MPC Algorithms for Euclidean k-Center

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

Guichen Gao
School of Computer Science, Peking University, Beijing, China

Mohsen Ghaffari
Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, USA

Shaofeng H.-C. Jiang
School of Computer Science, Peking University, Beijing, China

Abstract

The *k-center* problem is a fundamental optimization problem with numerous applications in machine learning, data analysis, data mining, and communication networks. The k-center problem has been extensively studied in the classical sequential setting for several decades, and more recently there have been some efforts in understanding the problem in parallel computing, on the Massively Parallel Computation (MPC) model. For now, we have a good understanding of k-center in the case where each local MPC machine has sufficient local memory to store some representatives from each cluster, that is, when one has $\Omega(k)$ local memory per machine. While this setting covers the case of small values of k, for a large number of clusters these algorithms require undesirably large local memory, making them poorly scalable. The case of large k has been considered only recently for the *fully scalable* low-local-memory MPC model for the Euclidean instances of the k-center problem. However, the earlier works have been considering only the constant dimensional Euclidean space, required a super-constant number of rounds, and produced only $k(1+o(1))$ centers whose cost is a super-constant approximation of k-center.

In this work, we significantly improve upon the earlier results for the k-center problem for the fully scalable low-local-memory MPC model. In the low dimensional Euclidean case in $\mathbb{R}^d$, we present the first constant-round fully scalable MPC algorithm for $(2+\varepsilon)$-approximation. We push the ratio further to $(1+\varepsilon)$-approximation albeit using slightly more $(1+\varepsilon)k$ centers. All these results naturally extends to slightly super-constant values of d. In the high-dimensional regime, we provide the first fully scalable MPC algorithm that in a constant number of rounds achieves an $O(\log n/\log\log n)$-approximation for k-center.

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

Keywords and phrases Massively Parallel Computing, Euclidean Spaces, k-Center Clustering

Digital Object Identifier 10.4230/LIPIcs.ICALP.2025.64

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: arXiv:2504.16382 [16]

Funding *Artur Czumaj*: Research supported in part by the Centre for Discrete Mathematics and its Applications (DIMAP), by EPSRC award EP/V01305X/1, by a Weizmann-UK Making Connections Grant, by an IBM Award.
Shaofeng H.-C. Jiang: Research partially supported by a national key R&D program of China No. 2021YFA1000900.

52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025).
Editors: Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis
Article No. 64; pp. 64:1–64:20

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Clustering and the k-Center Problem. Clustering is a fundamental task in data analysis and machine learning. We consider a well-known clustering problem, called k-Center, in Euclidean spaces. In this problem, given an integer parameter $k \geq 1$ and a dataset $P \subset \mathbb{R}^d$, the goal is to find a *center set* $C \subset \mathbb{R}^d$ of k points, such that the following clustering objective is minimized

$$\text{cost}(P, C) := \max_{p \in P} \text{dist}(p, C). \tag{1}$$

Here, $\text{dist}(x, y) := \|x - y\|_2$ for any two points $x, y \in \mathbb{R}^d$, and $\text{dist}(p, C) := \min_{c \in C} \text{dist}(p, c)$ for any point $p \in \mathbb{R}^d$ and any set of points $C \subset \mathbb{R}^d$.

Solving k-Center on massive data sets introduces outstanding scalability issues. To meet this scalability challenge, practical approaches usually use several interconnected computers to solve the problem, i.e., they resort to distributed computing. Accordingly, there has been significant recent interest in scalable algorithms with provable guarantees for k-Center [1, 5, 7, 9, 11, 14, 20, 31, 37, 44, 46], primarily in the Massively Parallel Computing (MPC) model, which has nowadays become the de-facto standard theoretical model for such large-scale distributed computation settings (see, e.g., [6, 29, 36]).

Massively Parallel Computation (MPC) Model. The MPC model, introduced in [43], provides a theoretical abstraction for widely-used practical frameworks such as MapReduce [19], Hadoop [52], Spark [53], and Dryad [38]. In this model, we are given a set of machines, each with some given memory of size s (also known as *local memory*). At the beginning of computation, the input (which in our case is a set of n data points from $\mathbb{R}^d$) is arbitrarily distributed among these machines, with the constraint that it must fit within each machine's local memory. (Hence we will require that the number of machines is $\Omega(n/s)$, for otherwise the input would not fit the system.) The MPC computation proceeds in *synchronous rounds*. In each round, first, each machine processes its local data and performs an arbitrary computation on its data without communicating with other machines. Then, at the end of each round, machines can communicate by exchanging messages, subject to the constraint that for every machine, the total size of the messages it sends or receives is $O(s)$. When the algorithm terminates, MPC machines collectively output the solution. The goal is to finish the computational task using as small as possible number of rounds.

Local Memory Regimes and Full Scalability. The central parameter determining the computational model is the size of the local memory s. Unlike the input size n, local memory is defined by the hardware provided and as such, one would like the relation between s and n to be as flexible as possible. Therefore an ideal MPC algorithm should be *fully scalable*, meaning that it should work with $s = n^\sigma$ for any constant $\sigma \in (0, 1)$. The importance of designing fully scalable MPC algorithms has been recently observed (cf. [8]) for clustering problems like k-Center (and also k-means and k-median), where the prior research (see below) demonstrates that the problem's difficulty changes radically depending on whether $s = \Omega(k)$ or not, i.e., whether one machine can hold the entire set of proposed centers or not. It is furthermore desirable for the algorithm to use a near-linear total memory.

Prior Work with High Local Memory Requirements. In the non-fully scalable regime, when $s = \Omega(k)$, a classical technique of *coresets* [32, 33] can be applied to reduce the n input points into a $(1 + \varepsilon)$-approximate proxy with only $O(k)$ points (ignoring $f(d) \cdot \text{poly}(\log n)$

factors). For k-Center, provided that $s = \Omega(kn^\gamma)$ for some constant $\gamma \in (0, 1)$ [7], this small proxy can be computed in $O(1)$ MPC rounds and moved to a single machine. The clustering problem (in fact, its approximate version because of the approximation caused by the use of coresets) can then be solved locally without further communication in a single MPC round. However, the coreset approach is not applicable when $s = o(k)$, because coresets suffer a trivial size lower bound of $\Omega(k)$ making the approach sketched above unsuitable. Hence, new techniques must be developed for *fully scalable algorithms when local memory is sub-linear in k*.

Several other works for k-Center, although not using a coreset directly, also follow a similar paradigm of finding a sketch of $\text{poly}(k)$ points in each machine [1, 20, 31, 46], and therefore they still require $s = \Omega(k)$. We remark that these results work under general metrics with distance oracle, which is a setting very different from our Euclidean setting. This fundamental difference translates to different flavor of studies: in general metrics not much more can be done than using the triangle inequality, whereas in $\mathbb{R}^d$ we need to investigate what Euclidean properties are useful for fully scalable algorithms.

Toward Fully Scalable MPC Algorithms. To combat these technical challenges, several recent works have designed fully scalable MPC algorithms for k-Center and for related clustering problems, including k-Median and k-Means [5, 8, 12, 13, 14, 17]. These MPC algorithms are fully scalable in the sense that they can work with $s = n^\sigma$ for any constant $\sigma \in (0, 1)$. Indeed, they usually work with any $s \geq f(d) \operatorname{poly} \log n$ regardless of k (albeit many of these results output more than k centers, only achieving a bi-criteria approximation). Therefore, in particular, despite the inspiring recent progress, fully scalable MPC algorithms for k-Center are poorly understood. The state-of-the-art algorithm (for geometric k-Center and only for $d = O(1)$) is by Coy, Czumaj, and Mishra [14]: it achieves a *super-constant* approximation ratio of $O(\log^* n)$ (improving over an $O(\log \log \log n)$-rounds bound from an earlier work of Batteni et al. [5]), using $k + o(k)$ centers; this violates the constraint of using at most k centers and works in a *super-constant* $O(\log \log n)$ number of rounds. These bounds, which are stated for the standard regime of $s = n^\sigma$ with a constant $\sigma \in (0, 1)$, show a drastic gap to the above-mentioned bounds achieved in the fully scalable $s = \Omega(k)$ regime.

Challenges of High Dimensionality. Apart from the fully-scalability, the high dimensionality of Euclidean spaces is another challenge in designing MPC algorithms. Indeed, there has been emerging works that address high dimensionality in MPC [4, 8, 12, 13, 17, 21, 39]. Unfortunately, all previous fully-scalable MPC algorithms for k-Center only work for low-dimensional regime since it requires $\exp(d)$ dependence in d in local space, and fully-scalable MPC algorithms suitable for high dimension, especially those with $\text{poly}(d)$ dependence, constitutes an open area of research. Overall, there has been a large gap in fully-scalable algorithms for k-Center under both low- and high-dimensional regime.

1.1 Our Results

We give new fully scalable MPC algorithms for Euclidean k-Center and we systematically address both the low-dimensional and high-dimensional regimes. Our results in low dimension significantly improve previous results simultaneously in various aspects (i.e., approximation ratio, round complexity, etc.). We also obtain the first results for high dimension, and our approximation ratio bypasses several natural barriers.

Low-Dimensional Regime. In low dimension, we provide the first fully scalable MPC algorithm that achieves a constant approximation to k-Center, running in a constant number of rounds (Theorem 1.1). This result significantly improves the previous fully scalable algorithms for k-Center [5, 14] in several major aspects: by achieving constant round complexity, better approximation factor, and true approximation (without bi-criteria considerations that allow slightly more than k centers).

▶ **Theorem 1.1.** *There exists an MPC algorithm that given $\varepsilon \in (0,1)$, $k \geq 1$, and a dataset $P \subset \mathbb{R}^d$ of n points distributed across MPC machines with local memory $s \geq (\Omega(d\varepsilon^{-1}))^{\Omega(d)} \operatorname{poly}\log n$, with probability at least $1 - 1/n$ computes a $(2+\varepsilon)$-approximate solution to k-Center, using $O(\log_s n)$ rounds and $O(n \cdot \operatorname{poly}\log n \cdot (O(d\varepsilon^{-1}))^{O(d)})$ total memory.*

Furthermore, we can improve the $(2+\varepsilon)$ approximation bound if we allow bi-criteria approximations: we design a $(1+\varepsilon)$-approximation k-Center algorithm that uses $(1+\varepsilon)k$ centers (Theorem 1.2). This bi-criteria approximation is almost optimal, and is the first of its kind in the literature.

▶ **Theorem 1.2.** *There exists an MPC algorithm that given $\varepsilon \in (0,1)$, $k \geq 1$, and a dataset $P \subset \mathbb{R}^d$ of n points distributed across MPC machines with local memory $s \geq (\Omega(d\varepsilon^{-1}))^{\Omega(d)} \operatorname{poly}\log n$, with probability at least $1-1/n$ computes a $(1+\varepsilon, 1+\varepsilon)$-approximate solution[1] to k-Center, using $O(\log_s n)$ rounds and $O(n \cdot \operatorname{poly}\log n \cdot (O(d\varepsilon^{-1}))^{O(d)})$ total memory.*

Observe that for the memory regime of $s = n^\sigma$ with a constant $\sigma \in (0,1)$, both Theorems 1.1 and 1.2 run in a constant number of rounds. Moreover, $\Theta(\log_s n)$ is the complexity of far more rudimentary tasks, e.g., outputting the summation of n numbers (see, e.g., [49]). The dependence on d in the memory bounds is $2^{\Theta(d \log d)}$. Thus, for any constant $\delta \in (0,1)$, if $d = o(\log n / \log\log n)$, then the algorithms work in a constant number of rounds with local memory $s \geq n^\delta$ and total memory $n^{1+o(1)}$, which is the regime studied in the previous works on fully scalable algorithms for k-Center [5, 14]. Furthermore, if $d = o(\log\log n / \log\log\log n)$, then the total memory is in the desirable regime of $O(n \operatorname{poly}\log(n))$.

High-Dimensional Regime. Next, we go beyond the low-dimensional regime and explore the high dimension case where d can be as large as $O(\log n)$.[2] For this regime, we provide the first fully scalable MPC algorithm for k-Center, and it achieves an $O(\log n / \log\log n)$-approximation, running in a constant number of rounds (Theorem 1.3).

▶ **Theorem 1.3.** *There exists an MPC algorithm that given $\varepsilon \in (0,1)$, $k \geq 1$, and a dataset $P \subset \mathbb{R}^d$ of n points distributed across MPC machines with local memory $s \geq \operatorname{poly}(d \log n)$, with probability at least $1 - 1/n$ computes an $O(\varepsilon^{-1} \log n / \log\log n)$-approximate solution to k-Center, using $O(\log_s n)$ rounds and $O(n^{1+\varepsilon} \operatorname{poly}(d \log n))$ total memory.*

We are not aware of any earlier fully scalable MPC algorithms for k-Center in high dimension that we could compare with. In fact, fully scalable MPC algorithms for clustering problems in high dimension are generally less understood, and the only known result is

[1] An (α, β)-*approximate* solution to k-Center (also called a *bi-criteria* solution) is a center set $C \subset \mathbb{R}^d$ that has at most βk centers and has cost at most α times the optimal cost of using at most k centers.

[2] As also observed in e.g., [12, 17], one can assume $d = O(\log n)$ without loss of generality by a Johnson-Lindenstrauss transform.

an $O(\log^2 n)$-approximation for k-Median [12] (and in fact this ratio may be improved to $O(\log^{1,.5} n)$ using the techniques from [2] although it is not explicitly mentioned), and as far as we know, nothing is known for k-Center and k-Means. These existing results for k-Median rely on the tree embedding technique, and currently only an $O(\log^{1.5} n)$-distortion is known [2] (which translates to the ratio). As a result, even if these techniques could be adapted for k-Center, it would only provide an $O(\log^{1.5} n)$-approximation, which falls short of the $O(\log n/\log\log n)$-approximation achieved by our method; in fact, our bound is even better than the fundamental lower bound of $\Omega(\log n)$-approximation of tree embedding. This is not to mention the added technical difficulty of using the tree embedding: its expected distance distortion guarantee is too weak to be useful for the "max" aggregation of distances in k-Center.

1.2 Technical Overview

Our algorithms for k-Center rely on variants of the classic reductions to geometric versions of the *ruling set* (RS) and *minimum dominating set* (MDS) problems, which are fundamental problems in distributed computing. We briefly describe the reductions in Section 1.2.1, particularly to mention our exact setup of RS and MDS, and state the results we obtain for each. Then in Section 1.2.2 and Section 1.2.3, we provide a technical overview of our proposed MPC algorithms for these results, focusing on the key challenges and core techniques. While the relation between k-Center and RS/MDS is well-known and has also been used in MPC algorithms for k-Center in general metrics [1, 31], it has not been studied for Euclidean k-Center in the fully scalable setting. This is a key technical difference to previous fully scalable algorithms for Euclidean k-Center [5, 14] which employ successive uniform sampling to find centers.

Both our low dimension and high dimension results rely on geometric hashing techniques (in Section 3), through which we utilize the Euclidean structure. For low dimension, our algorithms are based on natural parallel algorithms where similar variants were also considered in graph MPC algorithms, and the geometric hashing is the key to achieve the new bounds. For high dimension, our algorithm is a variant of the one-round version of Luby's algorithm. It has been known that the one-round Luby's algorithm yields a $\Theta(\log n)$ bound for RS in general graphs (see e.g. [24, Exercise 1.12]). However, our new variant crucially makes use of the Euclidean structure via geometric hashing, and it breaks the mentioned $\Theta(\log n)$ bound in general graphs, improving it by an $O(\log\log n)$ factor in the Euclidean setting; See Section 1.2.3 for a more formal discussion. Due to the space limit, proofs for claims made in this section may be omitted in this version and can be found in the full version [16].

1.2.1 Reductions and Results for Geometric RS and MDS

To establish Theorems 1.1–1.3, we begin with introducing the definitions and reductions for RS and MDS. Let $\tau, \alpha > 0$ be parameters, and let OPT be the minimum cost of the solution for k-Center.

Geometric RS and MDS. A subset $S \subseteq P$ is called a τ-independent set (τ-IS) for P, if for every $x \neq y \in S$, $\mathrm{dist}(x, y) > \tau$, and we say $S \subseteq \mathbb{R}^d$ is a τ-dominating set (τ-DS) for P, if for every $x \in P$, $\mathrm{dist}(x, S) \leq \tau$. A subset $S \subseteq P$ is a (τ, α)-ruling set ((τ, α)-RS) for P if S is both a τ-IS and α-DS for P. A τ-MDS is a τ-DS with the minimum size, denoted as $\mathrm{MDS}_\tau(P)$. A related well known notion is maximal independent set (MIS), where a τ-MIS is (τ, τ)-RS.

Table 1 RS and MDS results, where $\tilde{O}$ hides $\text{poly}(d \log n)$ factor, all run in $O(\log_s n)$ rounds.

guarantee	local space	total space	reference
$(\tau, (1+\varepsilon)\tau)$-RS	$(\varepsilon^{-1}d)^{O(d)} \cdot \tilde{O}(1)$	$\tilde{O}(n) \cdot (\varepsilon^{-1}d)^{O(d)}$	Lemma 4.1
$(1+\varepsilon)\tau$-DS of size $(1+\varepsilon)\lvert \text{MDS}_\tau(P)\rvert$	$(\varepsilon^{-1}d)^{O(d)} \cdot \tilde{O}(1)$	$\tilde{O}(n) \cdot (\varepsilon^{-1}d)^{O(d)}$	Lemma 4.2
$(\tau, O\left(\varepsilon^{-1} \frac{\log n}{\log\log n}\right)\tau)$-RS	$\tilde{O}(1)$	$\tilde{O}(n^{1+\varepsilon})$	Lemma 4.3

Reductions. It is known (see, e.g., [35]) that any τ-IS of P for $\tau \geq 2\,\text{OPT}$ must have at most k points. Therefore, an MPC algorithm that computes a $(2\,\text{OPT}, \alpha)$-RS of P would immediately yield α-approximation for k-Center on P. On the other hand, for MDS, since the optimal solution for k-Center itself is a candidate for a τ-MDS and has at most k points for $\tau = \text{OPT}$, a $(1+\varepsilon)$-approximation to τ-MDS would yield $(1+\varepsilon)k$ centers. This relation helps to obtain the desired bi-criteria approximation. Compared with the setting of RS which could only leads to some $O(1)$-approximation, MDS operates on the entire $\mathbb{R}^d$. This is necessary for the $(1+\varepsilon)$ ratio since centers need to be picked from $\mathbb{R}^d$ instead of only from P.

Results for RS and MDS. We obtain the following results for RS and MDS, in both low and high dimension. Combining with the above-mentioned reductions, these results readily imply Theorems 1.1–1.3. All results run in $O(\log_s n)$ rounds which is constant in the typical setup of $s = n^\sigma$ for constant $0 < \sigma < 1$. In low dimension, we obtain $(\tau, (1+\varepsilon)\tau)$-RS and a $(1+\varepsilon)\tau$-DS whose size is at most $(1+\varepsilon)$ times the τ-MDS (which in a sense is a "bi-criteria" approximation). Both results use $(\varepsilon^{-1}d)^{O(d)} \cdot \text{poly}(\log(n))$ local space, and $n\,\text{poly}(d \log n) \cdot (\varepsilon^{-1}d)^{O(d)}$ total space. In high dimension, we obtain $(\tau, (\varepsilon^{-1} \log n / \log\log n)\tau)$-RS, using (ideal) $\text{poly}(d \log n)$ local space and $n^{1+\varepsilon}\,\text{poly}(d \log n)$ total space. These results are summarized in Table 1.

We remark that MIS, RS, and MDS are fundamental yet notoriously challenging problems in MPC. Existing studies on these problems are mostly under (general) graphs or general metric spaces, and they achieve worse bounds than ours, e.g., they need to use a super-constant number of rounds [25, 26, 27, 30, 40, 47], and/or are not fully scalable [10, 27, 31]. However, our results seem to suggest that these problems in Euclidean spaces behave very differently than in graphs/general metrics. On the one hand, we obtain fully-scalable algorithms in both low and high dimension, but on the other hand, our algorithms are only "approximations" to MIS and MDS; for instance, in low dimension, both our RS and MDS results have $(1+\varepsilon)$ factor off in the dominating parameter to MIS and MDS. For RS/MIS and MDS without violating dominating parameter, we are only aware of a line of research in distributed computing for growth-bounded graphs, see Schneider and Wattenhofer [51], which indirectly lead to $O(\log^* n)$-rounds fully scalable algorithms for MIS/MDS in $\mathbb{R}^d$ for constant d. It is still open to design fully scalable algorithms for MIS, even in 2D, in *constant* number of rounds. In fact, this problem is already challenging on a 2D input set with diameter $O(\tau)$. Nevertheless, our RS and MDS results suffice for approximations for k-Center.

1.2.2 RS and MDS in Low Dimension

The RS and MDS in low dimension starts with a rounding to a $\varepsilon\tau/\sqrt{d}$-grid. Specifically, we move each data point to the nearest $\varepsilon\tau/\sqrt{d}$-grid point, whose coordinates are multiples of $\varepsilon\tau/\sqrt{d}$, denoted as P'.[3] Then we show that any $(\tau, \alpha\tau)$-RS to P' yields a $(\tau, (1+\varepsilon)\alpha\tau)$-RS

[3] In our proof we actually need to use a slightly different rounding to make sure the image also belongs to P.

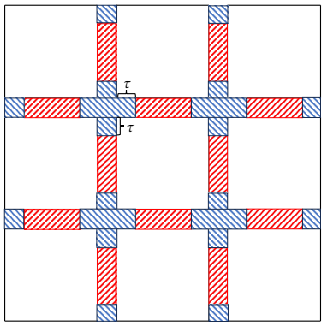

Figure 1 A space partition in 2D with $T = 3$ and $\alpha = 5$. The first group is squares with side-length $5\tau/\sqrt{2}$ (the blank space), the second is the red-shaded rectangles, and the third is the cross-like structures in blue shades.

to the original dataset P, and similarly, any τ-DS to P' yields a $(1+\varepsilon)\tau$-DS to P. Hence, we can assume without loss of generality that P is a subset of the said grid, and find RS and MDS on it. This rounding is useful, since it ensures that in any subset of $\mathbb{R}^d$ of diameter $\gamma \cdot \tau$ ($\gamma \geq 1$), the number of the grid points is at most $(O(d\gamma/\varepsilon))^d$. Hence, as long as $s \geq \Omega(d/\varepsilon)^d$, we can afford to bring all grid points in a small area to a single machine and solve RS/MDS on it locally.

An Overview for the Proof of RS. In fact, on the rounded dataset P, we find a (τ, τ)-RS which is as well τ-MIS on P. A standard way of finding MIS in a graph is a greedy algorithm: start with the entire vertex set, and if the current vertex set is nonempty, then find any vertex x, add it to the output (MIS) set, remove all vertices that are adjacent to x from the current vertex set, and repeat. In the geometric setting, we can use an improved version where in each iteration we identify a large number of vertices that are independent, instead of only one. Specifically, we partition the entire $\mathbb{R}^d$ into some T *groups* $\mathcal{W}_1, \ldots, \mathcal{W}_T$, such that each group consists of *regions* that are τ apart from each other. Furthermore, each region has a bounded diameter $\alpha\tau$ for some $\alpha \geq 1$.[4] For $d = 2$, there is a simple way to partition with $T = O(1)$ and $\alpha = O(1)$, as illustrated in Figure 1. For general d, we give a partition that achieves $T = O(d)$ and $\alpha = O(d^{1.5})$. See Lemma 3.1 for the precise statement.

A key property of this decomposition is that the MIS computation in each region can be done independently within each group, as regions are τ-separated. This yields an $O(T \log_s n)$-round MPC algorithm: iterating over the T groups $\mathcal{W}_1, \ldots, \mathcal{W}_T$, computing an MIS in each region in parallel, and removing points within τ from selected MIS points before proceeding to the next group. Since each region in any group is of diameter $\alpha\tau$ which is at most $d^{1.5}\tau$, there are at most $O(\varepsilon^{-1}d)^{O(d)}$ data points in each region, using the assumption of the rounded instance. Hence, as long as $s \geq \Omega(\varepsilon^{-1}d)^{\Omega(d)}$, the data points in each region can be stored in a single machine. Each such iteration can be implemented in $O(\log_s n)$ MPC rounds, and thus the overall scheme which has T iterations takes $O(T \log_s n)$ MPC rounds.

To reduce this to $O(\log_s n)$ rounds, we exploit the locality of the above procedure. Instead of iterating sequentially over groups, each region $R \in \mathcal{W}_i$ for $1 \leq i \leq T$ directly determines its final subset R' by identifying the influence from earlier groups $\bigcup_{j<i} \mathcal{W}_j$ that are within a

[4] The reader may notice the reminiscence with the widely-used notion of network decomposition in graphs [3, 50]. In that notion, the node set V of the graph is partitioned into $T = O(\log n)$ groups $V_1, \ldots, V_T$, such that in the subgraph induced by each V_i, each connected component has diameter at most $\alpha = O(\log n)$.

bounded range $O(i\tau + (i-1)\alpha\tau) \leq \text{poly}(d) \cdot \tau$. Since an MIS selection only affects a τ-radius per iteration, and each region has a diameter at most $\alpha\tau$, the cumulative affected area over i iterations remains bounded. Leveraging the rounding property again, the number of relevant regions remains at most $O(\varepsilon^{-1}d)^{O(d)}$, allowing all necessary regions to be replicated locally. This ensures that each region can compute its MIS in parallel in $O(\log_s n)$ rounds, provided $s \geq \Omega(\varepsilon^{-1}d)^{\Omega(d)} \operatorname{poly}(\log n)$.

An Overview for the Proof of MDS. We start with a weaker local space bound that requires a $2^{\Omega(d^2)}$ dependence of d in s, instead of the claimed $2^{\Omega(d \log d)}$ bound. Our approach starts with a simple algorithm: partition $\mathbb{R}^d$ into hypercubes of side-length $\alpha\tau$, where $\alpha \geq 1$ is a parameter to be determined. Then send the data points in each hypercube to a single machine, and solve locally the exact MDS of each hypercube. Taking the union of these MDSs forms the final dominating set. Clearly, this returns a dominating set for the entire dataset, but it may not be of small size. Intuitively, the major gap in this algorithm is when the optimal solution uses points located at the boundary of the hypercubes to dominate the adjacent hypercubes, while the algorithm described here only uses a point to dominate points in a single hypercube.

To address this issue, we observe that bridging the gap between the algorithm's solution and the optimal solution requires only bounding the number of points in the outer "τ-extended region" of each hypercube R, denoted as $U_\tau^\infty(R)$, that intersect with the optimal solution. Specifically, it suffices to show that this number is at most an ε-fraction of the optimal solution size. To establish this bound, we apply an averaging argument: If we shift the entire hypercube partitioning by some multiple of $O(\tau)$, there must exist a shift that satisfies our requirement, provided that the hypercube side-length $\alpha\tau$ is sufficiently large. This may be visualized more easily in 1D: each $U_\tau^\infty(R)$ is simply two intervals of length τ to the left and right of R (which itself is also an interval, whose length is $\alpha\tau$). Then by shifting a multiple of $O(\tau)$, the two intervals of U_τ^∞, after these shifts, form a partition of the entire $\mathbb{R}$. Unfortunately, this simple shifting of hypercubes only leads to $\alpha = 2^{O(d)}$, which translates to a $2^{O(d^2)}$ dependence of d in the local space s. The main reason for this $\alpha = 2^{O(d)}$ bound is that the same point in the optimal solution may belong to up to $2^{O(d)}$ sets $U_\tau^\infty(R)$ (for some hypercube R).

To further reduce $\alpha = \text{poly}(d)$, which leads to the $d^{O(d)}$ local space bound, we need to employ a more sophisticated geometric hashing. We state this in Lemma 3.1 and Fact 3.2. This hash maps each point in $\mathbb{R}^d$ into some bucket, and we would replace the hypercubes in the above-mentioned algorithm with such buckets. An important property of this bucketing is that any point in $\mathbb{R}^d$ (hence any point in the optimal solution) can intersect at most $\text{poly}(d)$ number of sets $U_\tau^\infty(R)$ over all buckets R, instead of $2^{O(d)}$ as in the simple hypercube partition. However, the use of this new bucketing also introduces additional issues. Specifically, since the buckets are of complicated structure, it is difficult to analyze the even more complex set $U_\tau^\infty(R)$ for the averaging argument. To this end, we manage to show (in Lemma 3.1, third property) that the union of $\bigcup_R U_\tau^\infty(R)$ is contained in the complement of a Cartesian power of (1D) intervals (e.g., $([1,2] \cup [3,4])^d$). This structure of Cartesian power is similar enough to the U_τ^∞ annulus of hypercubes (which may be handled by projecting to each dimension), albeit taking the complement. This eventually enables us to use a modified averaging argument to finish the proof.

1.2.3 RS in High Dimension

Our $(\tau, O(\varepsilon^{-1} \log n / \log\log n)\tau)$-RS in high dimension is a modification of the well-known Luby's algorithm. In this discussion, we assume $\varepsilon = \Theta(1)$ and ignore this parameter. The first modification is that, unlike the standard Luby's algorithm which runs for $O(\log n)$

iterations [45], our algorithm only runs Luby's for one iteration: for every data point $x \in P$, generate a uniform random value $h(x) \in [0,1]$, and then for each $x \in P$, include x in the RS if x has the smallest h value in $B_P(x,\tau)$ (the ball centered at x with radius τ, intersecting points in P). This one-round Luby's algorithm achieves $(\tau, O(\log n)\tau)$-RS (with high probability), and we also show that this is tight in general graphs.[5] However, this is worse than the $O(\log n/\log\log n)$ factor that we can achieve.

A New Preprocessing Step Based on Geometric Hashing. Hence, we need to introduce the second important modification to this one-round Luby's algorithm in order to bypass the $O(\log n)$ factor. Specifically, before running the one-round Luby's algorithm, we run a preprocessing step to map the data points to the buckets of a geometric hashing. The geometric hashing that we use is the *consistent hashing* [18] (see Lemma 3.4), and the concrete guarantee in our context is that, each hash bucket has diameter $\ell := O(\log n/\log\log n)\tau$, and for any subset $S \subseteq \mathbb{R}^d$ with diameter at most $O(\tau)$, the number of buckets that S intersects is at most $\Lambda := \text{poly}(\log n)$. We pick an arbitrary point in P from each (non-empty) bucket, denoting the resultant set as P', and we run the one-round Luby's on P'. Clearly, this hashing step only additively increases the dominating parameter by $\ell = O(\log n/\log\log n)\tau$ which we can afford. At a high level, the use of this rounding is to limit the size of the $O(\tau)$-neighborhood for every point, which is analogue to the degree of a graph. In a sense, what we prove is that one-round Luby's on graphs with degree bound $\text{poly}(\log n)$ yields an $O(\log n/\log\log n)$-ruling set.

Next, we explain in more detail why this hashing step helps to obtain $O(\log n/\log\log n)\tau$ dominating parameter. This requires us to do a formal analysis to one-round Luby's algorithm (which did not seem to appear in the literature), and utilize the property of hashing that for every $x \in P'$, $|B_{P'}(x,\tau)| \le \Lambda = \text{poly}(\log n)$.

A Re-Assignment Argument. Let R be the resultant set found by running one-round Luby's on P'. Fix a point $p \in P'$, and we need to upper bound $\text{dist}(x,R)$. To this end, we interpret the algorithm as defining an auxiliary sequence $S = (x_0 := p, x_1, \ldots, x_T)$ (where T is also random). Specifically, S is formed in the following process: we start with $i = 0$, whenever x_i is not picked into R we define x_{i+1} as the point with the smallest h value in $B_{P'}(x,\tau)$, and we terminate the procedure otherwise. Clearly, $\text{dist}(x,R) \le T\tau$, and it suffices to upper bound T (which is a random stopping time). Indeed, a similar plan of re-assignment argument has been employed in [17], but the definition and analysis of S is quite different since they focus on facility location.

Now, a crucial step is to bound the probability of the event that, given the algorithm picks a prefix $(x_0, \ldots, x_i)$ of sequence S, the algorithm picks some new $x_{i+1} \in P'$ instead of terminating. We call this *extension* probability. Ideally, if we can give this extension probability a universal upper bound $\gamma < 1$, then for $t \ge 1$, the probability that $T > t$ is at most γ^t, which decreases exponentially with respect to t. However, this seemingly good bound may not immediately be sufficient, since one still needs to take a union bound over sequences of length t, because the sequence S is random. A naïve bound is n^t since each x_i may be picked from any point in P', and this is positively large (i.e., γ^t cannot "cancel it out".). Moreover, an added difficulty is that the "ideal" case of having universal upper bound γ for the extension probability may not be possible in the first place.

[5] Similar bounds were also mentioned without proof in the literature, see e.g. [24, Exercise 1.12]. We give a proof (sketch) for this tight bound for completeness.

Our analysis resolves both difficulties. We show that the extension probability is roughly upper bounded by $|B_{P'}(x_i,\tau)|/|\bigcup_{j=1}^{i} B_{P'}(x_j,\tau)|$ (recalling that $(x_0,\ldots,x_i)$ is given). For the union bound, since the hashing guarantees that $|B_{P'}(x,\tau)| \le \Lambda = \text{poly}(\log n)$ for any $x \in P'$, we can approximate the size $|B_{P'}(x_i,\tau)|$ by rounding to the next power of 2, denoted as $\lceil |B_{P'}(x_i,\tau)| \rceil_2$, and this yields only $O(\log\log n)$ possibilities after rounding. For a (fixed) sequence $S' := (x_0,\ldots,x_m)$, we define its *configuration* as the rounded value of $(\lceil |B_{P'}(x_1,\tau)| \rceil_2, \ldots, \lceil |B_{P'}(x_m,\tau)| \rceil_2)$. We can do a union bound with respect to the configuration of the sequences, and there are at most $O(\log\log n)^t$ number of configurations for length-t sequences.

Finally, given a sequence $S' = (x_0,\ldots,x_t)$ with a given configuration (for some t), to upper bound the extension probability, we need to analyze $\prod_i \frac{|B_{P'}(x_i,\tau)|}{|\sum_{j=1}^{i} B_{P'}(x_j,\tau)|}$, and we show in a key lemma that this is upper bounded by $\exp(-\Omega(t)\log(t/\log\Lambda))$. Therefore, we can pick $t = \log n/\log\log n$, apply the union bound and conclude that $\Pr[T > t] \le (\log\log n)^t \cdot \exp(-\Omega(t)\log(t/\log\Lambda)) \le 1/\text{poly}(n)$.

We also provide a (sketch) of how to slightly modify our analysis to show the one-round Luby's algorithm yields $O(\tau, O(\log n)\tau)$-RS. As mentioned, this did not seem to appear in the literature. We also provide a tight instance for this one-round Luby's algorithm.

1.3 Related Work

The k-Center problem, as one of the fundamental clustering problems [28, 34, 35], has been studied extensively in sequential setting, and more recently, also in parallel setting. For MPC algorithms for k-Center, most of prior works have focused on non-fully scalable algorithms that have a dependence of $\Omega(k)$ in the local memory s and can generally achieve $O(1)$ rounds. In general metric space, [20] obtains a large-constant approximation, using $O(1/\sigma)$ rounds if the local memory is $\text{poly}(k)n^\sigma$, which offers a tradeoff between the number of rounds and the local memory. More recent works aim to achieve a smaller constant ratio, down to factor 2, at the cost of having local memory size with a fixed polynomial dependence in n and/or a polynomial dependence in the number of machines [1, 31, 37, 46]. There has been also some k-Center studies on doubling metrics instances (which is a generalization of $\mathbb{R}^d$) [7, 11], where in the bi-criteria (or with outliers) setting, not only a constant but even a $(1+\varepsilon)$ ratio can be achieved due to the low-dimensional structure [7].

Fully-scalable MPC algorithms have been also studied for related clustering problems of k-Median and k-Means in $\mathbb{R}^d$. [12] describes a hierarchical clustering algorithm that in a constant number of rounds returns a $\text{poly}\log(n)$-approximation for k-Median using $s = \text{poly}(d)\cdot n^\sigma$ local memory; we are not aware of any similar approximation for k-Means (even when $d = O(1)$ and with $\text{poly}\log n$ ratio). Furthermore, since the techniques used in [12] rely critically on the approach of hierarchically well-separated trees, they are unlikely to lead to sub-polylogarithmic approximation ratio algorithms. On the other hand, bi-criteria approximation are known for both k-Median and k-Means [8, 17], and in a constant number of rounds and with $s = \text{poly}(d)\cdot n^\sigma$ local memory, one can approximate k-median and k-means with $O(1)$ ratio using $(1+\varepsilon)k$ centers [17]. In the same setting, it is also possible to achieve a $(1+\varepsilon)$-approximation for special inputs [13].

2 Preliminaries

For integer $n \ge 1$, let $[n] := \{1,\ldots,n\}$. For a mapping $f : X \to Y$, denote $f^{-1}(y) := \{x \in X : f(x) = y\}$ as the pre-image of f. For some $d \ge 1$, a set $S \subseteq \mathbb{R}^d$ and $x \in \mathbb{R}^d$, write $S + x$ to denote $\{x + y : y \in S\}$. For $t > 0$, let $\mathcal{G}_t \subseteq \mathbb{R}^d$ be the set of t-grid points, that

is, points whose coordinates take values as integer multiples of t. For a subset $S \subset \mathbb{R}^d$, let $\mathrm{diam}(S) := \max_{x,y\in S} \mathrm{dist}(x,y)$ be its diameter. Similarly define dist_∞ and diam_∞ as the ℓ_∞ version.

Neighborhoods. Let $B(x,r) := \{y \in \mathbb{R}^d : \mathrm{dist}(x,y) \le r\}$ be a ball centered at $x \in \mathbb{R}^d$ with radius $r \ge 0$. For a point set $X \subseteq \mathbb{R}^d$, let $B_X(x,r) := B(x,r) \cap X$ denote a ball inside X. For a point set $S \subset \mathbb{R}^d$ and $\gamma > 0$, let $N^\infty_\gamma(S)$ be the γ-neighborhood of S under the ℓ_∞ distance, i.e., $N^\infty_\gamma(S) := \{x \in \mathbb{R}^d : \exists s \in S, ||x-s||_\infty \le \gamma\}$, and let $U^\infty_\gamma(S) := N^\infty_\gamma(S) \setminus S$ be the γ-annulus of S under ℓ_∞ distance.

Geometric Independent Set, Ruling Set and Dominating Set. Let $\tau > 0$ be some threshold, $\alpha > 0$ be some parameter and $P \subset \mathbb{R}^d$ be a dataset. A subset $S \subseteq P$ is called a τ-independent set (τ-IS) for P, if for every $x \ne y \in S$, $\mathrm{dist}(x,y) > \tau$, and we say $S \subseteq \mathbb{R}^d$ is a τ-dominating set (τ-DS) for P, if for every $x \in P$, $\mathrm{dist}(x,S) \le \tau$. A subset $S \subseteq P$ is a (τ,α)-ruling set ((τ,α)-RS) for P if S is both a τ-IS and α-DS for P. A τ-MDS is a τ-DS with the minimum size, denoted as $\mathrm{MDS}_\tau(P)$. A related well known notion is maximal independent set (MIS), where a τ-MIS for P, denoted as $\mathrm{MIS}_\tau(P)$, is a (τ,τ)-RS for P.

***k*-Center.** Recall that the objective of k-Center is defined in Section 1 as $\mathrm{cost}(P,C)$ for a dataset P and center set C. Let $\mathrm{OPT}(P)$ be the minimum value of the solution for k-Center, i.e., $\mathrm{OPT}(P) := \min_{C\subset\mathbb{R}^d} \mathrm{cost}(P,C)$. When the context is clear, we simply write OPT for $\mathrm{OPT}(P)$.

Standard MPC Primitives. In our algorithms we frequently use several basic primitives on MPC with local memory $s \ge \mathrm{poly}\log(N)$ using total memory $O(N \,\mathrm{poly}\log N)$ and number of rounds $O(\log_s N)$, where N is the size of a generic input. This includes standard procedures of broadcast and converge-cast (of a message that is of size $\le \sqrt{s}$), see e.g. [23]. Goodrich et al. [29] show that the task of sorting N numbers can also be performed deterministically in the above setting.

▶ **Lemma 2.1** (Packing property, cf. [48, Lemma 4.1]). *For a point set $S \subset \mathbb{R}^d$ such that $\forall x \ne y \in S, dist(x,y) \ge \rho$, we have that $|S| \le (\frac{3\,\mathrm{diam}(S)}{\rho})^d$.*

3 Geometric Hashing

We present our new geometric hashing in Lemma 3.1. This hashing is crucially used in our low dimension results. The construction of this hashing is the same as [18, Theorem 5.3], and the first two properties have also been established in [18]. However, the third property is new, and is based on a careful analysis that utilizes the structure of this specific construction.

▶ **Lemma 3.1.** *For every $\beta > 0$, $\ell \ge \Theta(d^{1.5}\beta)$, there is a hash function $f : \mathbb{R}^d \to \mathbb{R}^d$ such that the following holds.*

1. *Each bucket has diameter at most ℓ, namely, for every image $u \in f(\mathbb{R}^d)$, $\mathrm{diam}(f^{-1}(u)) \le \ell$.*
2. *The bucket set $\{f^{-1}(u) : u \in f(\mathbb{R}^d)\}$ can be partitioned into $d+1$ groups $\{\mathcal{W}_i\}_{i=0}^d$, such that every two buckets $S \ne S'$ in the same group $\mathcal{W}_i$ $(0 \le i \le d)$ has $\mathrm{dist}(S,S') \ge \mathrm{dist}_\infty(S,S') > \beta$.*

3. *For every* $0 < \tau \le \beta$, $\left(\bigcup_{u\in f(\mathbb{R}^d)} U_\tau^\infty(f^{-1}(u))\right) \cap L(z, 2b)^d = \emptyset$,[6] *where* $z := \ell/\sqrt{d}$, $b := d\beta + \tau$ *and* $L(p, q) := \bigcup_{a\in\mathbb{Z}}[ap + q, (a + 1)p - q)$ *for* $p > 2q$.

Furthermore, it takes $\mathrm{poly}(d)$ *space to store* f *and to evaluate* $f(x)$ *for every* $x \in \mathbb{R}^d$.

Proof. The proof can be found in Section A. ◀

Lemma 3.1 readily implies the following property. Roughly speaking, it ensures that for any point set S with small enough ℓ_∞ diameter, the number of intersected buckets in the hash is bounded.

▶ **Fact 3.2.** *The second property of Lemma 3.1 implies that for every* $S \subset \mathbb{R}^d$ *such that* $\mathrm{diam}_\infty(S) \le \beta$, *it holds that* $|f(S)| \le d + 1$.

Geometric hash functions that has similar guarantee as in Fact 3.2 has been studied under the notion of consistent hashing [18], or sparse partitions which interpret the hashing as a space partition [22, 41]. Specifically, the definition of consistent hashing is stated as follows. The consistently guarantee of Fact 3.2 is slightly stronger in the sense that the diameter bound of S is in ℓ_∞ instead of ℓ_2.

▶ **Definition 3.3** ([18, Definition 1.6]). *A mapping* $\varphi : \mathbb{R}^d \to \mathbb{R}^d$ *is called a* Γ*-gap* Λ*-consistent hash with diameter bound* $\ell > 0$, *or simply* (Γ, Λ)*-hash , if it satisfies:*

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

Lemma 3.4 gives a space-efficient consistent hashing with near-optimal parameter tradeoffs. We rely on this parameter tradeoff in a preprocessing step of our high dimension ruling set result.

▶ **Lemma 3.4** ([18, Theorem 5.1]). *For every* $\Gamma \in [8, 2d]$, *there exists a (deterministic)* (Γ, Λ)*-hash* $\varphi : \mathbb{R}^d \to \mathbb{R}^d$ *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* $O(d^2 \log^2 d)$.

4 MPC Algorithms for RS and MDS: Formal Statements

In this section, we present the formal statements of our results for RS and MDS. The proofs for the low-dimensional RS and MDS results (Lemmas 4.1 and 4.2) closely follow the arguments outlined in the technical overview (see Section 1.2.2) and are therefore omitted due to space limitation (and can be found in the full version [16]). We nonetheless provide a proof outline for the high-dimensional RS result (Lemma 4.3) in Section 5.

▶ **Lemma 4.1.** *There is a deterministic MPC algorithm that given threshold* $\tau > 0$, *constant* $\varepsilon \in (0, 1)$ *and dataset* $P \subseteq \mathbb{R}^d$ *of* n *points distributed across MPC machines with local memory* $s \ge \Omega(\varepsilon^{-1}d)^{\Omega(d)} \cdot \mathrm{poly}(\log n)$, *computes a* $(\tau, (1 + \varepsilon)\tau)$*-RS for* P *in* $O(\log_s n)$ *rounds, using* $O(n\,\mathrm{poly}(d \log n)) \cdot O(\varepsilon^{-1}d)^{O(d)}$ *total memory.*

▶ **Lemma 4.2.** *There is a deterministic MPC algorithm that given threshold* $\tau > 0$, *parameter* $\varepsilon \in (0, 1)$ *and dataset* $P \subseteq \mathbb{R}^d$ *of* n *points distributed across MPC machines with local memory* $s \ge \Omega(\varepsilon^{-1}d)^{\Omega(d)}\,\mathrm{poly}(\log n)$, *computes a* $(1 + \varepsilon)\tau$*-DS* $S \subset \mathbb{R}^d$ *for* P *such that* $|S| \le (1 + \varepsilon)|\,\mathrm{MDS}_\tau(P)|$ *in* $O(\log_s n)$ *rounds, using* $O(n\,\mathrm{poly}(\log n)) \cdot O(\varepsilon^{-1}d)^{O(d)}$ *total memory.*

[6] Recall that the notation $L(\cdot, \cdot)^d$ denotes the d-th Cartesian power of $L(\cdot, \cdot)$.

▶ **Lemma 4.3.** *There is an MPC algorithm that given threshold $\tau > 0$, $0 < \varepsilon < 1$ and a dataset P of n points in $\mathbb{R}^d$ distributed across MPC machines with local memory $s \geq \text{poly}(d \log n)$, with probability at least $1 - 1/n$ computes a $(\tau, O(\varepsilon^{-1} \frac{\log n}{\log \log n})\tau)$-ruling set for P in $O(\log_s n)$ rounds, using $O(n^{1+\varepsilon} \text{poly}(d \log n))$ total memory.*

5 Proof Outline of Lemma 4.3: RS in High Dimension

Our algorithm may be viewed as a Euclidean version of one-round Luby's, with two major differences. One is an additional preprocessing algorithm Algorithm 1, and this step is crucially needed to improve the ruling set parameter from $O(\log n)$ (which is what one-round Luby's algorithm can achieve in general [24]) to $(\log n / \log \log n)$. The other is that it is not immediate to exactly simulate the one-round Luby's in high dimensional Euclidean spaces, as the neighborhood around a data point can be huge and is not easy to handle in MPC. To this end, we need to use some approximate ball $A_P^\beta(p, \tau)$ that is "sandwiched" between $B_P(p, \tau)$ and $B_P(p, \beta\tau)$ for some β and every point $p \in P$, and an MPC procedure for this has been suggested in [17] (restated in Lemma 5.7). We would address this inaccuracy introduced by $A_P^\beta(\cdot, \cdot)$ in the analysis of the offline algorithm. In the remainder of this section, we use the notations from Algorithms 1 and 2.

Algorithm 1 Preprocessing, with input $P \subseteq \mathbb{R}^d$ of n points, $\beta \geq 1, \tau > 0$.

1: let φ be a consistent hashing with parameter $\Gamma \leftarrow O(d/\log \log n)$ $\Lambda \leftarrow \text{poly}(d \log n)$ and diameter $\ell \leftarrow O(\beta\Gamma\tau)$, using Lemma 3.4
2: for every $z \in \varphi(P)$, pick an arbitrary representative point $x \in \varphi^{-1}(z) \cap P$, denoted as $\text{rep}(z)$
3: return $P' \leftarrow \text{rep}(\varphi(P))$

Algorithm 2 Local algorithm for RS on P' resultant from Algorithm 1, same $\beta \geq 1, \tau > 0$.

1: for each $p \in P'$, pick a uniformly random label $h(p) \in [0, 1]$
2: initialize $R \leftarrow \emptyset$ and for every $p \in P'$, $R \leftarrow R \cup \{p\}$ if p has the smallest label in $A_{P'}^\beta(p, \tau)$
3: return R

▶ **Lemma 5.1.** *R is a τ-independent set for P' (with probability 1).*

▶ **Lemma 5.2.** *For any $\alpha \geq 1$, any $(\tau, \alpha\tau)$-ruling set for P' is an $(\tau, (\alpha + \beta\Gamma)\tau)$-ruling set for P.*

▶ **Fact 5.3.** *For every $p \in P'$, $|A_{P'}^\beta(x, \tau)| \leq \Lambda$.*

▶ **Lemma 5.4.** *For every $t \geq \Omega(\log^2 \Lambda)$, the set R returned by Algorithm 2 satisfies*

$$\forall p \in P', \quad \text{dist}(p, R) \leq O(\beta t)\tau$$

with probability at least $1 - n \cdot \exp(-\Omega(t) \log(t/\log \Lambda))$.

Proof. It suffices to show that for every $p \in P'$, with probability $1 - \exp(-\Omega(t) \log(t/\log \Lambda))$, $\text{dist}(p, R) \leq O(\beta t)\tau$, since one can conclude the proof using a union bound for all points $p \in P'$.

Now, fix a point $p \in P'$ and consider $\mathrm{dist}(p, R)$. In order to analyze $\mathrm{dist}(p, R)$, we construct a sequence $S := (x_0, x_1, \ldots, x_T)$ using an auxiliary algorithm, stated in Algorithm 3, where the sequence S starts at the point $x_0 := p$ and denote the length of S as $T \geq 0$ which is random. We emphasize that Algorithm 3 is defined with respect to the internal states of a specific (random) run of Algorithm 2, and it does not introduce new randomness.

Algorithm 3 Finding an assignment sequence $S = (x_0 = p, \ldots, x_T)$, for a given $p \in P'$.

let $i \leftarrow 0$, $x_0 \leftarrow p$, $S \leftarrow (x_0)$
while Algorithm 2 does not add x_i at line 2 **do**
    let x_{i+1} be the point from $A^{\beta}_{P'}(x_i, \tau)$ with the smallest label
    let $S \leftarrow S \circ x_{i+1}$ and $i \leftarrow i + 1$
end while
return S

Observe that in Algorithm 3, for every $i \geq 0$, $x_{i+1} \in A^{\beta}_{P'}(x_i, \tau)$. Since for every $p \in P'$, $A^{\beta}_{P'}(p, \tau) \subseteq B_{P'}(p, \beta\tau)$, then we have that

$$\mathrm{dist}(p, R) \leq \sum_{i=1}^{T} \mathrm{dist}(x_{i-1}, x_i) \leq T\beta\tau$$

by triangle inequality. Hence, it remains to give an upper bound for T, and we show this in the following Lemma 5.5 which is the main technical lemma. Its proof can be found in the full version due to space limit.

▶ **Lemma 5.5.** *For $t \geq \Omega(\log^2 \Lambda)$, we have $\Pr[T \geq t] \leq \exp(-\Omega(t)\log(t/\log\Lambda))$.*

Finally, as mentioned, applying Lemma 5.5 with a union bound finishes the proof of Lemma 5.4. ◀

Proof of Lemma 4.3. Our MPC algorithm for Lemma 4.3 is obtained via an MPC implementation of the offline Algorithms 1 and 2. However, before we do so, our algorithm requires to run the Johnson-Lindenstrauss (JL) transform [42] on the dataset as preprocessing to reduce d to $d = O(\log n)$, using parameter $\varepsilon = O(1)$, restated as follows with our notations.

▶ **Lemma 5.6** (JL transform [42])**.** *For some $0 < \varepsilon < 1$, let $M \in \mathbb{R}^{d' \times d}$ be a random matrix such that every entry is an independent standard Gaussian variable $N(0, 1)$ where $d' = O(\varepsilon^{-2}\log n)$, then the mapping $g : x \mapsto 1/\sqrt{d'} \cdot Mx$ satisfies that with probability $1 - 1/\mathrm{poly}(n)$, $\forall x, y \in P$, $\mathrm{dist}(g(x), g(y)) \in (1 \pm \varepsilon)\,\mathrm{dist}(x, y)$.*

This JL transform only needs $O(1)$ rounds to run in MPC, since one can generate the matrix M in a lead machine, which uses space $O(d \log n)$, and then broadcast to every other machines. Since the pairwise distance between data points is preserved, and that our algorithms only use the distance between data points, it is immediate that any $(\tau, \alpha\tau)$-ruling set on $g(P)$ is as well a $(\Theta(\tau), \Theta(\alpha\tau))$-ruling set on P. Hence, it suffices to work on $g(P)$ which is of dimension $d' = O(\log n)$. Without loss of generality, we can simply assume P is of dimension $O(\log n)$.

Now, we turn to implement Algorithms 1 and 2 in MPC. For Algorithm 1, since the hash in Lemma 3.4 that we use is data oblivious and only requires $\mathrm{poly}(d) = \mathrm{poly}(\log n)$ space, all machines can generate the same hash without communication. The other steps in Algorithm 1 may be implemented using standard MPC procedures including sorting, broad-casting and converge-casting. For Algorithm 2, the the only nontrivial step is to

implement $A^{\beta}_{P'}(\cdot,\cdot)$. To this end, we make use of the following lemma from [17]. In particular, our MPC implementation of Algorithm 2 applies Lemma 5.7 with $\beta = O(\varepsilon^{-1})$. This finishes the description of the algorithm for Lemma 4.3.

▶ **Lemma 5.7** ([17, Theorem 3.1]). *There is a deterministic MPC algorithm that takes as input $0 < \varepsilon < 1$, $\tau \geq 0$, $P \subset \mathbb{R}^d$ of n points and for each $p \in P$ a value $h(p) \in \mathbb{R}$, distributed across machines with local memory $s \geq \text{poly}(d \log n)$, computes for every $p \in P$ a value $\min(h(A_P(p,\tau)))$, where $A_P(p,\tau)$ is an arbitrary set that satisfies*

$$B_P(p,\tau) \subseteq A_P(p,\tau) \subseteq B_P(p, O(\varepsilon^{-1}\tau)), \tag{2}$$

in $O(\log_s n)$ rounds and $O(n^{1+\varepsilon}\,\text{poly}(d \log n))$ total memory.

Now we turn to the analysis. Observe that the assumptions regarding $A_{P'}(\cdot,\cdot)$ underlying the analysis of Algorithms 1 and 2 remain valid. By Lemma 5.1, the found set R is also an τ-independent for P as $R \subseteq P' \subseteq P$. Moreover, since we assume $d = O(\log n)$, the parameters $\Gamma = \log n / \log\log n$, and $\Lambda = \text{poly}(\log n)$. Therefore, applying Lemma 5.4 with $t = O(\log n / \log\log n)$, we conclude that $\forall p \in P'$, $\text{dist}(p,R) \leq O(\varepsilon^{-1} \log n / \log\log n)\tau$, with probability $1 - 1/\text{poly}(n)$. Combining with Lemma 5.2, we conclude that R is $(\tau, O(\varepsilon^{-1} \log n / \log\log n)\tau)$-ruling set for P, with probability $1 - 1/\text{poly}(n)$.

Finally, for the round complexity, local memory and total memory, these are dominated by the parallel invocations of Lemma 5.7, which are $O(\log_s n)$, $\text{poly}(d \log n)$ and $O(n^{1+\varepsilon}\,\text{poly}(d \log n))$, respectively. Therefore, we complete the proof of Lemma 4.3. ◀

References

1 Sepideh Aghamolaei and Mohammad Ghodsi. A 2-approximation algorithm for data-distributed metric k-center. *arXiv*, 2309.04327, 2023. doi:10.48550/arXiv.2309.04327.

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 ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 77–88. ACM, 2023. doi:10.1145/3558481.3591096.

3 Baruch Awerbuch, Andrew V Goldberg, Michael Luby, and Serge A Plotkin. Network decomposition and locality in distributed computation. In *30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 364–369. IEEE Computer Society, 1989. doi:10.1109/SFCS.1989.63504.

4 Amir Azarmehr, Soheil Behnezhad, Rajesh Jayaram, Jakub Lacki, Vahab Mirrokni, and Peilin Zhong. Massively parallel minimum spanning tree in general metric spaces. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 143–174. SIAM, 2025. doi:10.1137/1.9781611978322.5.

5 MohammadHossein Bateni, Hossein Esfandiari, Manuela Fischer, and Vahab S. Mirrokni. Extreme k-center clustering. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, (AAAI)*, pages 3941–3949. AAAI Press, 2021. doi:10.1609/AAAI.V35I5.16513.

6 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.

7 Mark de Berg, Leyla Biabani, and Morteza Monemizadeh. k-center clustering with outliers in the MPC and streaming model. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 853–863. IEEE, 2023. doi:10.1109/IPDPS54959.2023.00090.

8 Aditya Bhaskara and Maheshakya Wijewardena. Distributed clustering via LSH based data partitioning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 569–578. PMLR, 2018. URL: http://proceedings.mlr.press/v80/bhaskara18a.html.

9 Leyla Biabani and Ami Paz. k-center clustering in distributed models. In *Structural Information and Communication Complexity - 31st International Colloquium (SIROCCO)*, volume 14662 of *Lecture Notes in Computer Science*, pages 83–100. Springer, 2024. doi:10.1007/978-3-031-60603-8_5.

10 Mélanie Cambus, Fabian Kuhn, Shreyas Pai, and Jara Uitto. Time and space optimal massively parallel algorithm for the 2-ruling set problem. In *37th International Symposium on Distributed Computing (DISC)*, volume 281 of *LIPIcs*, pages 11:1–11:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.DISC.2023.11.

11 Matteo Ceccarello, Andrea Pietracaprina, and Geppino Pucci. Solving k-center clustering (with outliers) in mapreduce and streaming, almost as accurately as sequentially. *Proceedings of the VLDB Endowment*, 12(7):766–778, 2019. doi:10.14778/3317315.3317319.

12 Vincent Cohen-Addad, Silvio Lattanzi, Ashkan Norouzi-Fard, Christian Sohler, and Ola Svensson. Parallel and efficient hierarchical k-median clustering. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 20333–20345, 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/aa495e18c7e3a21a4e48923b92048a61-Abstract.html.

13 Vincent Cohen-Addad, Vahab S. Mirrokni, and Peilin Zhong. Massively parallel k-means clustering for perturbation resilient instances. In *International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pages 4180–4201. PMLR, 2022. URL: https://proceedings.mlr.press/v162/cohen-addad22b.html.

14 Sam Coy, Artur Czumaj, and Gopinath Mishra. On parallel k-center clustering. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 65–75. ACM, 2023. doi:10.1145/3558481.3591075.

15 Artur Czumaj, Arnold Filtser, Shaofeng H.-C. Jiang, Robert Krauthgamer, Pavel Veselý, and Mingwei Yang. Streaming facility location in high dimension via geometric hashing. *arXiv preprint arXiv:2204.02095*, 2022. The latest version has additional results compared to the preliminary version in [18]. arXiv:2204.02095.

16 Artur Czumaj, Guichen Gao, Mohsen Ghaffari, and Shaofeng H. C. Jiang. Fully scalable mpc algorithms for euclidean k-center. *arXiv*, 2504.16382, 2025. doi:10.48550/arXiv.2504.16382.

17 Artur Czumaj, Guichen Gao, Shaofeng H.-C. Jiang, Robert Krauthgamer, and Pavel Veselý. Fully-scalable MPC algorithms for clustering in high dimension. In *51st International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 297 of *LIPIcs*, pages 50:1–50:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ICALP.2024.50.

18 Artur Czumaj, Shaofeng H.-C. Jiang, Robert Krauthgamer, Pavel Veselý, and Mingwei Yang. Streaming facility location in high dimension via geometric hashing. In *63rd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 450–461. IEEE, 2022. doi:10.1109/FOCS54457.2022.00050.

19 Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008. doi:10.1145/1327452.1327492.

20 Alina Ene, Sungjin Im, and Benjamin Moseley. Fast clustering using MapReduce. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 681–689. ACM, 2011. doi:10.1145/2020408.2020515.

21 Alessandro Epasto, Mohammad Mahdian, Vahab S. Mirrokni, and Peilin Zhong. Massively parallel and dynamic algorithms for minimum size clustering. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1613–1660. SIAM, 2022. doi:10.1137/1.9781611977073.66.

22 Arnold Filtser. Scattering and sparse partitions, and their applications. *ACM Transactions on Algorithms*, 20(4):30:1–30:42, 2024. doi:10.1145/3672562.

23 Mohsen Ghaffari. Massively parallel algorithms, 2019. Lecture Notes from ETH Zürich. URL: http://people.csail.mit.edu/ghaffari/MPA19/Notes/MPA.pdf.

24 Mohsen Ghaffari. Distributed graph algorithms, 2022. Lecture Notes from MIT. URL: https://people.csail.mit.edu/ghaffari/DA22/Notes/DGA.pdf.

25 Mohsen Ghaffari, Christoph Grunau, and Ce Jin. Improved MPC algorithms for MIS, matching, and coloring on trees and beyond. In *34th International Symposium on Distributed Computing (DISC)*, volume 179 of *LIPIcs*, pages 34:1–34:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.DISC.2020.34.

26 Mohsen Ghaffari and Jara Uitto. Sparsifying distributed algorithms with ramifications in massively parallel computation and centralized local computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1636–1653. SIAM, 2019. doi:10.1137/1.9781611975482.99.

27 Jeff Giliberti and Zahra Parsaeian. Massively parallel ruling set made deterministic. In *38th International Symposium on Distributed Computing (DISC)*, volume 319 of *LIPIcs*, pages 29:1–29:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.DISC.2024.29.

28 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. doi:10.1016/0304-3975(85)90224-5.

29 Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the MapReduce framework. In *Algorithms and Computation - 22nd International Symposium (ISAAC)*, volume 7074 of *Lecture Notes in Computer Science*, pages 374–383. Springer, 2011. doi:10.1007/978-3-642-25591-5_39.

30 Chetan Gupta, Rustam Latypov, Yannic Maus, Shreyas Pai, Simo Särkkä, Jan Studený, Jukka Suomela, Jara Uitto, and Hossein Vahidi. Fast dynamic programming in trees in the MPC model. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 443–453. ACM, 2023. doi:10.1145/3558481.3591098.

31 Alireza Haqi and Hamid Zarrabi-Zadeh. Almost optimal massively parallel algorithms for k-center clustering and diversity maximization. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 239–247. ACM, 2023. doi:10.1145/3558481.3591077.

32 Sariel Har-Peled. No, coreset, no cry. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 3328 of *Lecture Notes in Computer Science*, pages 324–335. Springer, 2004. doi:10.1007/978-3-540-30538-5_27.

33 Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 291–300. ACM, 2004. doi:10.1145/1007352.1007400.

34 Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985. doi:10.1287/MOOR.10.2.180.

35 Dorit S. Hochbaum and David B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33(3):533–550, 1986. doi:10.1145/5925.5933.

36 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. doi:10.1561/2400000025.

37 Sungjin Im and Benjamin Moseley. Brief announcement: Fast and better distributed mapreduce algorithms for k-center clustering. In *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 65–67. ACM, 2015. doi:10.1145/2755573.2755607.

38 Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: Distributed data-parallel programs from sequential building blocks. In *Proceedings of the 2007 EuroSys Conference*, pages 59–72. ACM, 2007. doi:10.1145/1272996.1273005.

39 Rajesh Jayaram, Vahab Mirrokni, Shyam Narayanan, and Peilin Zhong. Massively parallel algorithms for high-dimensional Euclidean minimum spanning tree. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3960–3996. SIAM, 2024. doi:10.1137/1.9781611977912.139.

40 Hongyan Ji, Kishore Kothapalli, Sriram V. Pemmaraju, and Ajitanshu Singh. Fast deterministic massively parallel ruling sets algorithms. In *Proceedings of the 26th International Conference on Distributed Computing and Networking (ICDCN)*, pages 152–160. ACM, 2025. doi:10.1145/3700838.3700872.

41 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. ACM, 2005. doi:10.1145/1060590.1060649.

42 William Johnson and Joram Lindenstrauss. Extensions of Lipschitz maps into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984. doi:10.1090/conm/026/737400.

43 Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for MapReduce. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 938–948. SIAM, 2010. doi:10.1137/1.9781611973075.76.

44 Ting Liang, Qilong Feng, Xiaoliang Wu, Jinhui Xu, and Jianxin Wang. Improved approximation algorithm for the distributed lower-bounded k-center problem. In *Theory and Applications of Models of Computation - 18th Annual Conference (TAMC)*, volume 14637 of *Lecture Notes in Computer Science*, pages 309–319. Springer, 2024. doi:10.1007/978-981-97-2340-9_26.

45 Michael Luby. A simple parallel algorithm for the maximal independent set problem. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10. ACM, 1985. doi:10.1145/22145.22146.

46 Gustavo Malkomes, Matt J. Kusner, Wenlin Chen, Kilian Q. Weinberger, and Benjamin Moseley. Fast distributed k-center clustering with outliers on massive data. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1063–1071, 2015. URL: https://proceedings.neurips.cc/paper/2015/hash/8fecb20817b3847419bb3de39a609afe-Abstract.html.

47 Krzysztof Onak. Round compression for parallel graph algorithms in strongly sublinear space. *arXiv*, 1807.08745, 2018. doi:10.48550/arXiv.1807.08745.

48 David Pollard. *Empirical Processes: Theory and Applications*, chapter 4: Packing and Covering in Euclidean Spaces, pages 14–20. IMS, 1990. doi:10.1214/cbms/1462061091.

49 Tim Roughgarden, Sergei Vassilvitski, 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.

50 Václav Rozhoň and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 350–363. ACM, 2020. doi:10.1145/3357713.3384298.

51 Johannes Schneider and Roger Wattenhofer. An optimal maximal independent set algorithm for bounded-independence graphs. *Distributed Computing*, 22(5-6):349–361, 2010. doi:10.1007/S00446-010-0097-1.

52 Tom White. *Hadoop: The Definitive Guide*. O'Reilly, 4th edition, 2015. URL: http://www.oreilly.de/catalog/9781491901632/index.html.

53 Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud'10*. USENIX Association, 2010. URL: https://www.usenix.org/conference/hotcloud-10/spark-cluster-computing-working-sets.

A Proof of Lemma 3.1: Geometric Hashing

▶ **Lemma 3.1.** *For every $\beta > 0$, $\ell \geq \Theta(d^{1.5}\beta)$, there is a hash function $f : \mathbb{R}^d \to \mathbb{R}^d$ such that the following holds.*

1. *Each bucket has diameter at most ℓ, namely, for every image $u \in f(\mathbb{R}^d)$, $\operatorname{diam}(f^{-1}(u)) \leq \ell$.*

2. *The bucket set* $\{f^{-1}(u) : u \in f(\mathbb{R}^d)\}$ *can be partitioned into* $d+1$ *groups* $\{\mathcal{W}_i\}_{i=0}^{d}$*, such that every two buckets* $S \neq S'$ *in the same group* $\mathcal{W}_i$ $(0 \leq i \leq d)$ *has* $\mathrm{dist}(S, S') \geq \mathrm{dist}_\infty(S, S') > \beta$.
3. *For every* $0 < \tau \leq \beta$, $\left(\bigcup_{u \in f(\mathbb{R}^d)} U_\tau^\infty(f^{-1}(u))\right) \cap L(z, 2b)^d = \emptyset$,[7] *where* $z := \ell/\sqrt{d}$, $b := d\beta + \tau$ *and* $L(p, q) := \bigcup_{a \in \mathbb{Z}}[ap + q, (a+1)p - q)$ *for* $p > 2q$.

Furthermore, it takes $\mathrm{poly}(d)$ *space to store* f *and to evaluate* $f(x)$ *for every* $x \in \mathbb{R}^d$.

The construction of the hash $f : \mathbb{R}^d \to \mathbb{R}^d$ is the same as that in [15, Theorem 5.3]. Their proof already implies the space bound, as well as the first two properties; In fact, our second property is stronger than what they state, but the stronger version was indeed proved in their paper, albeit implicitly. We would restate their algorithm/construction, briefly sketch the proof for the first two properties, and focus on the third property.

Hash Construction. We start with reviewing their algorithm/construction of f. Partition $\mathbb{R}^d$ into hypercubes of side length z of the form $\bigtimes_{i=1}^{d}[a_i z, (a_i+1)z)$ where $a_1, \ldots, a_d \in \mathbb{Z}$. Notice that each of hypercubes is of diameter ℓ. Let $\ell_i := (d-i)\beta$ where $i \in \{0, \ldots, d-1\}$. For every $i \in \{0, \ldots, d\}$, let $\mathcal{F}_i$ be the set of i-dimensional faces of the above-mentioned hypercubes, and let A_i be the set of points that belong to these faces, i.e., $A_i := \bigcup_{Q \in \mathcal{F}_i} Q$. For each $i \in \{0, \ldots, d-1\}$, let $B_i := N_{\ell_i}^\infty(A_i)$ be ℓ_i-neighborhood of A_i (with respect to ℓ_∞ distance). Let $B_{-1} := \emptyset$ and $B_{\leq i} := \bigcup_{j \leq i} B_j$. The main procedure for the construction of f goes as follows.

- For every $i \in \{0, \ldots, d-1\}$, for every $Q \in \mathcal{F}_i$, let $\widehat{Q} := N_{\ell_i}^\infty(Q) \setminus B_{\leq i-1}$, and for every $x \in \widehat{Q}$ assign $f(x) = q$ where $q \in \widehat{Q}$ is an arbitrary but fixed point. Observe that $\widehat{Q} \subseteq B_i$ and thus $x \in \widehat{Q}$ will not be assigned again in later iterations.
- For every $Q \in \mathcal{F}_d$, let $\widehat{Q} := Q \setminus B_{\leq d-1}$ be the remaining part of Q whose $f(x)$ has not been assigned, and assign $f(x) = q$ for every $x \in \widehat{Q}$, where $q \in \widehat{Q}$ is arbitrary but fixed point.

First Two Properties. The first property is immediate from this construction since every bucket is a subset of a hypercube of side-length z whose diameter is $\sqrt{d}z = \ell$. To establish the second property, we need to define the $d+1$ groups. For $0 \leq i \leq d-1$, define $\mathcal{W}_i := \{N_{\ell_i}^\infty(Q) \setminus B_{\leq i-1} : Q \in \mathcal{F}_i\}$ and let $\mathcal{W}_d := \{Q \setminus B_{\leq d-1} : Q \in \mathcal{F}_d\}$. Then by the construction of f, $\{f^{-1}(u) : u \in f(\mathbb{R}^d)\} = \bigcup_{i=0}^{d} \mathcal{W}_i$, hence the $\mathcal{W}_i$ is a proper partition of buckets. These $\mathcal{W}_i$'s are implicitly analyzed in [15, Lemma 5.5 and Lemma 5.8], restated and adapted (as Lemma A.1) to our notation as follows. This lemma readily implies our second property.

▶ **Lemma A.1** ([15, Lemma 5.5 and Lemma 5.8]). *For every* $0 \leq i \leq d$ *and every* $S \neq S' \in \mathcal{W}_i$, $\mathrm{dist}(S, S') \geq \mathrm{dist}_\infty(S, S') > \beta$.

Third Property. We proceed to prove the third property. The third property, particularly $L(z, 2b)$ is well-defined, since by the choice of parameters it can be verified that $z > 4b$. The proof starts with the following claim. It relates the complicated $\bigcup_u U_\tau^\infty(f^{-1}(u))$ to a much simpler object $N_b^\infty(A_{d-1})$, which is merely the b-neighborhood of $(d-1)$-dimensional faces.

▷ Claim A.2. $\bigcup_{u \in f(\mathbb{R}^d)} U_\tau^\infty(f^{-1}(u)) \subseteq N_b^\infty(A_{d-1})$.

[7] Recall that the notation $L(\cdot, \cdot)^d$ denotes the d-th Cartesian power of $L(\cdot, \cdot)$.

Proof. Recall that $\{f^{-1}(u) : u \in f(\mathbb{R}^d)\} = \bigcup_{i=0}^{d} \mathcal{W}_i$. Therefore, it suffices to prove that for every $i \in \{0, \ldots, d\}$ and for every $S \in \mathcal{W}_i$, $U_\tau^\infty(S) \subseteq N_b^\infty(A_{d-1})$. We prove this separately for the case of $i \leq d-1$ and $i = d$.

Case I: $i \leq d-1$. We first analyze the case that $0 \leq i \leq d-1$. Fix some $0 \leq i \leq d-1$ and some $S := N_{\ell_i}^\infty(Q) \setminus B_{\leq i-1} \in \mathcal{W}_i$ (for some $Q \in \mathcal{F}_i$). Observe that $S = N_{\ell_i}^\infty(Q) \setminus B_{\leq i-1} \subseteq N_{\ell_i}^\infty(Q) \subseteq N_{\ell_i}^\infty(A_i)$. Then we have that $N_\tau^\infty(S) \subseteq N_\tau^\infty(N_{\ell_i}^\infty(A_i)) = N_{\ell_i+\tau}^\infty(A_i)$. Observe that $\ell_i = (d-i)\beta \leq d\beta$. Hence, $U_\tau^\infty(S) = N_\tau^\infty(S) \setminus S \subseteq N_\tau^\infty(S) \subseteq N_{\ell_i+\tau}^\infty(A_i) \subseteq N_{d\beta+\tau}^\infty(A_{d-1}) = N_b^\infty(A_{d-1})$. This finishes the case of $i \leq d-1$.

Case II: $i = d$. Next, we analyze the case that $i = d$. By Lemma A.1, for every $S \neq S' \in \mathcal{W}_d$, $\mathrm{dist}_\infty(S, S') > \beta \geq \tau$, and this implies

$$N_\tau^\infty(S) \cap S' = \emptyset, \tag{3}$$

as ℓ_∞ distance between any two points is at most their ℓ_2 distance. Now fix some $S \in \mathcal{W}_d$. By the construction, $A_d = \mathbb{R}^d$ and $B_{\leq d-1} \cap \bigcup_{S \in \mathcal{W}_d} S = \emptyset$. By (3), $N_\tau^\infty(S)$ has no intersection with $\mathcal{W}_d$ other than on S. Hence,

$$U_\tau^\infty(S) = N_\tau^\infty(S) \setminus S \subseteq B_{\leq d-1}. \tag{4}$$

To further relate $B_{\leq d-1}$ with A_{d-1}, we have

$$B_{\leq d-1} = \bigcup_{j=0}^{d-1} B_j = \bigcup_{j=0}^{d-1} N_{\ell_j}^\infty(A_j) \subseteq \bigcup_{j=0}^{d-1} N_{d\beta}^\infty(A_j) \subseteq N_{d\beta}^\infty(A_{d-1}),$$

where the last step follows from $A_j \subseteq A_{j+1}$ for every j. Combining this with (4), we conclude that $U_\tau^\infty(S) \subseteq N_{d\beta}^\infty(A_{d-1}) \subseteq N_b^\infty(A_{d-1})$. This finishes the proof of Claim A.2. ◁

By Claim A.2, it remains to prove $N_b^\infty(A_{d-1}) \cap L(z, 2b)^d = \emptyset$. Since $N_b^\infty(A_{d-1}) = N_b^\infty(\bigcup_{Q \in \mathcal{F}_{d-1}} Q) = \bigcup_{Q \in \mathcal{F}_{d-1}} N_b^\infty(Q) = \bigcup_{x \in Q, Q \in \mathcal{F}_{d-1}} N_b^\infty(x)$, it suffices to show $\forall x \in Q$ (for $Q \in \mathcal{F}_{d-1}$), $N_b^\infty(x) \cap L(z, 2b)^d = \emptyset$. Now fix some $Q \in \mathcal{F}_{d-1}$ and $x \in Q$. Since Q is a $(d-1)$-dimensional face, it has $d-1$ dimensions spanning an entire interval of length z, and has exactly one dimension taking a value of the form az ($a \in \mathbb{Z}$). Formally, $x \in Q$ if and only if there exists $0 \leq j \leq d$ and $\{a_i \in \mathbb{Z}\}_{i=0}^d$ such that for $0 \leq i \leq d$,

$$x_i \in \begin{cases} [a_i z, (a_i+1)z) & i \neq j \\ \{a_i z\} & i = j \end{cases}. \tag{5}$$

Let j be that as in (5). Then for every $y \in L(z, 2b)^d$,

$$\|x - y\|_\infty \geq |x_j - y_j| \geq 2b,$$

where the last inequality follows from the definition of $L(z, 2b)$ and recall that $L(z, 2b) = \bigcup_{a \in \mathbb{Z}} [az + 2b, (a+1)z - 2b)$. This further implies $\forall x' \in N_b^\infty(x)$ and $y \in L(z, 2b)^d$, $\|x' - y\|_\infty \geq b$, by the triangle inequality of ℓ_∞. Therefore, we conclude that $N_b^\infty(x) \cap L(z, 2b)^d = \emptyset$, and this finishes the proof of Lemma 3.1. ◀