Fully Scalable MPC Algorithms for Euclidean k-Center

Artur Czumaj
University of Warwick
A.Czumaj@warwick.ac.uk

Guichen Gao
Peking University
gc.gao@stu.pku.edu.cn

Mohsen Ghaffari
MIT
ghaffari@mit.edu

Shaofeng H.-C. Jiang
Peking University
shaofeng.jiang@pku.edu.cn

April 24, 2025

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.

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

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

Here, $\mathrm{dist}(x,y) := \|x-y\|_2$ for any two points $x, y \in \mathbb{R}^d$, and $\mathrm{dist}(p,C) := \min_{c\in C} \mathrm{dist}(p,c)$ 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 [EIM11, IM15, MKC+15, CPP19, BEFM21, CCM23, HZ23, AG23, BBM23, BP24, LFW+24], 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., [GSZ11, BKS17, IKL+23]).

Massively Parallel Computation (MPC) model. The MPC model, introduced in [KSV10], provides a theoretical abstraction for widely-used practical frameworks such as MapReduce [DG08], Hadoop [Whi15], Spark [ZCF+10], and Dryad [IBY+07]. 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. [BW18]) 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* [Har04, HM04] 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 $\gamma \in (0, 1)$ [BBM23], 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 sublinear 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 [EIM11, MKC+15, HZ23, AG23], 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 [BW18, BEFM21, CLN+21, CMZ22, CCM23, CGJ+24]. 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 [CCM23]: 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. [BEFM21]), 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 [BW18, CLN+21, CMZ22, EMMZ22, CGJ+24, JMNZ24, ABJ+25]. 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 $\operatorname{poly}(d)$ 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 [BEFM21, CCM23] 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 total memory $O(n \cdot \operatorname{poly} \log n \cdot (O(d\varepsilon^{-1}))^{O(d)})$.*

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 total memory $O(n \cdot \operatorname{poly}\log n \cdot (O(d\varepsilon^{-1}))^{O(d)})$.*

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., [RVW18]). 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 [BEFM21, CCM23]. 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 total memory $O(n^{1+\varepsilon} \operatorname{poly}(d \log n))$.*

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 an $O(\log^2 n)$-approximation for k-Median [CLN+21] (and in fact this ratio may be improved to $O(\log^{1.5} n)$ using the techniques from [AAH+23] 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 [AAH+23] (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

[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., [CLN+21, CGJ+24], one can assume $d = O(\log n)$ without loss of generality by a Johnson-Lindenstrauss transform.

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 [HZ23, AG23], however, 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 [BEFM21, CCM23] 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. Previously, the one-round Luby's algorithm is known to yield a $\Theta(\log n)$ bound for RS in general graphs. 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.

1.2.1 Reductions and Results for Geometric RS and MDS

To establish Theorems 1.1 to 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.

Reductions. It is known (see, e.g., [HS86], and also Fact 7.1) that any τ-IS of P for $\tau \geq 2\,\mathrm{OPT}$ must have at most k points. Therefore, an MPC algorithm that computes a $(2\,\mathrm{OPT}, \alpha)$-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 $\tau = \mathrm{OPT}$, 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 abovementioned reductions, these results readily imply Theorems 1.1 to 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 $(\varepsilon^{-1}d)^{O(d)} \cdot \mathrm{poly}(\log(n))$ local space, and $n\,\mathrm{poly}(d \log n) \cdot (\varepsilon^{-1}d)^{O(d)}$ total space. In high dimension, we obtain $(\tau, (\varepsilon^{-1} \log n / \log \log n)\tau)$-RS, using (ideal) $\mathrm{poly}(d \log n)$ local space and $n^{1+\varepsilon}\,\mathrm{poly}(d \log n)$ total space. These results are summarized in Table 1.

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 $(1+\varepsilon)\lvert \mathrm{MDS}_\tau(P)\rvert$	$(\varepsilon^{-1}d)^{O(d)} \cdot \tilde{O}(1)$	$\tilde{O}(n) \cdot (\varepsilon^{-1}d)^{O(d)}$	Lemma 5.1
$(\tau, O\left(\varepsilon^{-1}\frac{\log n}{\log\log n}\right)\tau)$-RS	$\tilde{O}(1)$	$\tilde{O}(n^{1+\varepsilon})$	Lemma 6.1

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

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 [Ona18, GU19, GGJ20, GLM+23, GP24, JKPS25], and/or are not fully scalable [CKPU23, HZ23, GP24]. 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 [SW10], 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 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

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

[4] The reader may notice the reminiscence with the widely-used notion of network decomposition in graphs [AGLP89, RG20]. 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 $\alpha = O(\log n)$.

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.

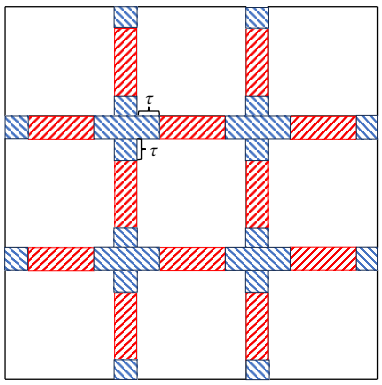

Figure 1: A space partition in 2D with $T = 3$ and $\alpha = 5$. The first group is squares with side-length $5\tau/\sqrt{2}$ (the blank space), the second is the red-shaded rectangles, and the third is the cross-like structures in blue shades.

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 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 $s \geq \Omega(\varepsilon^{-1}d)^{\Omega(d)} \text{poly}(\log n)$.

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 abovementioned 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 [Lub85], 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; see Appendix B.[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* [CJK+22] (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

[5] Similar bounds were also mentioned without proof in the literature, see e.g. [Gha22, Exercise 1.12]. We give a proof (sketch) for this tight bound for completeness.

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 $x \in P'$, $|B_{P'}(x,\tau)| \leq \Lambda = \text{poly}(\log n)$.

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, $\text{dist}(x, R) \leq T\tau$, 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 [CGJ+24], 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 $t \geq 1$, 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.

Our analysis resolves both difficulties. We show in Lemma 6.9 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 $|B_{P'}(x,\tau)| \leq \Lambda = \text{poly}(\log n)$ 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 6.12 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 $\Pr[T > t] \leq (\log\log n)^t \cdot \exp(-\Omega(t)\log(t/\log\Lambda)) \leq 1/\text{poly}(n)$.

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, in Appendix B. As mentioned, this did not seem to appear in the literature. We also provide a tight instance for this one-round Luby's algorithm, in Appendix B.2.

1.3 Related Work

The k-Center problem, as one of the fundamental clustering problems [Gon85, HS85, HS86], 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, [EIM11] 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 [MKC+15, IM15, HZ23, AG23]. There has been also some k-Center studies on doubling metrics instances (which is a generalization of $\mathbb{R}^d$) [CPP19, BBM23], 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 [BBM23].

Fully-scalable MPC algorithms have been also studied for related clustering problems of k-Median and k-Means in $\mathbb{R}^d$. [CLN+21] 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 [CLN+21] 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 [BW18, CGJ+24], 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 [CGJ+24]. In the same setting, it is also possible to achieve a $(1+\varepsilon)$-approximation for special inputs [CMZ22].

2 Preliminaries

For integer $n \geq 1$, 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 $d \geq 1$, 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 $\text{diam}(S) := \max_{x,y \in S} \text{dist}(x, y)$ be its diameter. Similarly define dist_∞ and diam_∞ as the ℓ_∞ version.

Neighborhoods. Let $B(x, r) := \{y \in \mathbb{R}^d : \text{dist}(x, y) \leq r\}$ be a ball centered at $x \in \mathbb{R}^d$ with radius $r \geq 0$. 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.,

$$N^\infty_\gamma(S) := \{x \in \mathbb{R}^d : \exists s \in S, ||x - s||_\infty \leq \gamma\},$$

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 $x \neq y \in S$, $\text{dist}(x, y) > \tau$, and we say $S \subseteq \mathbb{R}^d$ is a τ-dominating set (τ-DS) for P, if for every $x \in P$, $\text{dist}(x, S) \leq \tau$. 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 $\text{MDS}_\tau(P)$. A related well known notion is maximal independent set (MIS), where a τ-MIS for P, denoted as $\text{MIS}_\tau(P)$, is a (τ, τ)-RS for P.

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

Standard MPC primitives. In our algorithms we frequently use several basic primitives on MPC with local memory $s \geq \operatorname{poly}\log(N)$ using total memory $O(N \operatorname{poly}\log N)$ 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 $\leq \sqrt{s}$), see e.g. [Gha19]. Goodrich et al. [GSZ11] show that the task of sorting N numbers can also be performed deterministically in the above setting.

Lemma 2.1 (Packing property, cf. [Pol90, Lemma 4.1])**.** *For a point set $S \subset \mathbb{R}^d$ such that $\forall x \neq y \in S, dist(x,y) \geq \rho$, we have that $|S| \leq (\frac{3\operatorname{diam}(S)}{\rho})^d$.*

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 [CJK+22, Theorem 5.3], and the first two properties have also been established in [CJK+22]. 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 $\beta > 0$, $\ell \geq \Theta(d^{1.5}\beta)$, 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 $\operatorname{dist}(S,S') \geq \operatorname{dist}_\infty(S,S') > \beta$.*
3. *For every $0 < \tau \leq \beta$, $\left(\bigcup_{u \in f(\mathbb{R}^d)} U_\tau^\infty(f^{-1}(u))\right) \cap L(z, 2b)^d = \emptyset$,[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 $\operatorname{poly}(d)$ space to store f and to evaluate $f(x)$ for every $x \in \mathbb{R}^d$.

Proof. The proof can be found in Appendix 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 $\operatorname{diam}_\infty(S) \leq \beta$, it holds that $|f(S)| \leq d+1$.*

Geometric hash functions that has similar guarantee as in Fact 3.2 has been studied under the notion of consistent hashing [CJK+22], or sparse partitions which interpret the hashing as a space partition [JLN+05, Fil24]. 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 ([CJK+22, 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 $\operatorname{diam}(\varphi^{-1}(z)) \leq \ell$; and
- Consistency: for every $S \subset \mathbb{R}^d$ with $\operatorname{diam}(S) \leq \ell/\Gamma$, we have $|\varphi(S)| \leq \Lambda$.

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

Lemma 3.4 gives a space-efficient consistent hashing with near-optimal parameter tradeoffs. We rely on this parameter tradeoff in a preprocessing step (Algorithm 4) of our high dimension ruling set result Lemma 6.1.

Lemma 3.4 ([CJK+22, 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 in Low Dimension

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 $s \geq \Omega(\varepsilon^{-1}d)^{\Omega(d)} \cdot \mathrm{poly}(\log n)$, computes a $(\tau, (1+\varepsilon)\tau)$-RS for P in $O(\log_s n)$ rounds, using total memory $O(n\,\mathrm{poly}(d \log n)) \cdot O(\varepsilon^{-1}d)^{O(d)}$.*

We start with an efficient MPC reduction, stated in Lemma 4.3, that turns the input P to a point set that satisfies the following property.

Property 4.2. A point set P satisfies the property, if $\forall x \in \mathbb{R}^d$, $\rho \geq 1$, $B_P(x, \rho\tau) \leq (\varepsilon^{-1}\rho d)^{O(d)}$.

Lemma 4.3. *There exists an MPC algorithm that computes a point set $P' \subseteq P$ such that P' satisfies Property 4.2, and that any τ-DS for P' is a $(1 + O(\varepsilon))\tau$-DS for P, within round and space bound as in Lemma 4.1.*

Proof. To define P', let $\phi : P \to \mathcal{G}_{\varepsilon\tau/\sqrt{d}}$, such that for every $x \in P$, $\phi(x)$ maps to the nearest point in $\mathcal{G}_{\varepsilon\tau/\sqrt{d}}$ of x. For every $z \in \phi(P)$, let $\mathrm{rep}(z)$ be an arbitrary fixed point in $\phi^{-1}(z) \cap P$. Then, define $P' := \{\mathrm{rep}(z) : z \in \phi(P)\} \subseteq P$. Clearly, this whole process can be done within the claimed round and space as in Lemma 4.1.

Now fix a τ-DS S for P', and we verify that S is $(1+\varepsilon)\tau$-DS for P. Observe that for $z \in \phi(P)$, we have $\mathrm{dist}(z, \mathrm{rep}(z)) \leq \varepsilon\tau$. Then consider $x \in P$, we have

$$\begin{aligned}
\mathrm{dist}(x, S) &\leq \mathrm{dist}(x, P') + \max_{y \in P'} \mathrm{dist}(y, S) \\
&= \min_{z \in \phi(P)} \mathrm{dist}(x, \mathrm{rep}(z)) + \tau \\
&\leq \min_{z \in \phi(P)} \{\mathrm{dist}(x, z) + \mathrm{dist}(z, \mathrm{rep}(z))\} + \tau \\
&\leq (1 + 2\varepsilon)\tau.
\end{aligned}$$

Finally, to bound $|B_{P'}(x, \rho\tau)|$, observe that

$$|B_{P'}(x, \rho\tau)| = |B(x, \rho\tau) \cap \{\mathrm{rep}(z) : z \in \phi(P)\}| = |\{z \in \phi(P) : \mathrm{dist}(x, \mathrm{rep}(z)) \leq \rho\tau\}|,$$

where the second equality follows from the fact that $z \mapsto \mathrm{rep}(z)$ (for $z \in \phi(P)$) is one-to-one correspondence. Let $Q := \{z \in \phi(P) : \mathrm{dist}(x, \mathrm{rep}(z)) \leq \rho\tau\}$, and it suffices to bound $|Q|$. To this end, we wish to apply Lemma 2.1. Observe that $Q \subseteq \mathcal{G}_{\varepsilon\tau/\sqrt{d}}$. To bound the diameter of Q, consider $u \neq v \in Q \subseteq \phi(P)$, then by triangle inequality

$$\begin{aligned}
\mathrm{dist}(u, v) &\leq \mathrm{dist}(u, \mathrm{rep}(u)) + \mathrm{dist}(\mathrm{rep}(u), x) + \mathrm{dist}(x, \mathrm{rep}(v)) + \mathrm{dist}(\mathrm{rep}(v), v) \\
&\leq 2\varepsilon\tau + 2\rho\tau \\
&= O(\rho)\tau.
\end{aligned}$$

Hence, we conclude that $|B_{P'}(x, O(\tau))| = |Q| \leq (\varepsilon^{-1}\rho d)^{O(d)}$ by Lemma 2.1. This finishes the proof. □

Consider a P' as in Lemma 4.3, since a τ-MIS on P' is also a τ-DS for P', then it implies that a τ-MIS on P' is a $(\tau, (1 + O(\varepsilon))\tau)$-RS for P. Therefore, in the remainder of the proof, it suffices to design an MPC algorithm that finds a τ-MIS on a point set P with the assumption that P satisfies Property 4.2 (and we omit the notation P').

To proceed, our MIS algorithm relies on the following notion of geometric decomposition.

Definition 4.4. A collection $\mathcal{W} := \{S_1, S_2, \ldots\}$ of (disjoint) point sets of $\mathbb{R}^d$ is an α-bounded β-separated decomposition, or simply (α, β)-decomposition, if it satisfies the following.

1. For every $S \in \mathcal{W}$, $\operatorname{diam}(S) \leq \alpha$.
2. For every distinct $S_i, S_j \in \mathcal{W}$, $\operatorname{dist}(S_i, S_j) > \beta$.

In addition, we say that a set of points $X \subseteq \mathbb{R}^d$ admits an (α, β)-decomposition, if X can be *partitioned* into some (α, β)-decomposition $\mathcal{W}$.

Fact 4.5. *For threshold $\tau > 0$, let f be the hash guaranteed by Lemma 3.1 with parameter $\ell = O(d^{1.5}\tau)$. Then by the second property of Lemma 3.1, the buckets of the hash can be partitioned into $d+1$ groups $\{\mathcal{W}_i\}_{i=0}^{d}$, such that the collection of the buckets $\mathcal{W}_i$ is an $(O(d^{1.5})\tau, \tau)$-decomposition.*

Proof overview. Before we present the final algorithm, we start with a more sequential version which is listed in Algorithm 1, whose correctness is easily analyzed in Lemma 4.6. This algorithm relies on the hash guaranteed by Fact 4.5 whose buckets can be partitioned into $d + 1$ groups. It processes these $d + 1$ groups $\mathcal{W}_0, \ldots, \mathcal{W}_d$ one by one, where in a group $\mathcal{W}_i$ the MIS of all buckets $S \in \mathcal{W}_i$ are computed. Our final $O(1)$-round algorithm is a parallel implementation of it, where we utilize the locality of buckets among $\mathcal{W}_i$'s.

Algorithm 1 Basic sequential τ-MIS algorithm for the set P satisfying Property 4.2

```
let f be the hash and let W_0, ..., W_d be the groups of buckets from Lemma 3.1 and Fact 4.5
                                  ▷ so each W_i is an (O(d^1.5)τ, τ)-decomposition
for i ← 0, ..., d do
    for S ∈ W_i do
        let I^seq_{i,S} ← MIS_τ(S ∩ P \ B(I^seq_{i-1}, τ))        ▷ define I^seq_{-1} ← ∅
    end for
    let I^seq_i ← I^seq_{i-1} ∪ ⋃_{S∈W_i} I^seq_{i,S}
end for
return I^seq_d
```

Lemma 4.6. *Algorithm 1 returns a τ-MIS for P.*

Proof. The algorithm processes each $\mathcal{W}_i$ sequentially, remove the points that are within distance τ from the current independent set I^{seq}_{i-1} from $\mathcal{W}_i$, and finds MIS on the remaining points of buckets in $\mathcal{W}_i$. It is clear that the found sets I^{seq}_i for $i = 0, \ldots, d$ are all independent, and in particular the returned set I^{seq}_d is independent. Furthermore, since all $\mathcal{W}_i$'s are processed and that they form a partition of $\mathbb{R}^d$, I^{seq}_d is maximal. This finishes the proof. □

Notice that every distinct buckets $S, S' \in \mathcal{W}_i$ are τ apart. Hence, in Algorithm 1, the MIS $I^{\text{seq}}_{i,S}$ may be computed in parallel for all $S \in \mathcal{W}_i$. However, these MIS also needs to know I^{seq}_{i-1}, which depends on the buckets in $\mathcal{W}_0, \ldots, \mathcal{W}_{i-1}$, and this requires to (sequentially) examining $\mathcal{W}_0, \ldots, \mathcal{W}_{i-1}$ before computing $I^{\text{seq}}_{i,S}$.

A natural idea for avoiding this sequential computing, is to prune the buckets in sets $\mathcal{W}_0, \ldots, \mathcal{W}_{i-1}$, so that only those "relevant" to S are kept. Specifically, we are to figure out subsets $\mathcal{W}'_j \subseteq \mathcal{W}_j$ (which depends on i and S) to replace $\mathcal{W}_j$, such that it suffices to simulate the MIS on buckets in $\mathcal{W}'_j$ for computing $I^{\text{seq}}_{i,S}$. We shall see that these subsets $\mathcal{W}'_j$ are not only of small size, i.e., $O(\varepsilon^{-1}d)^{O(d)}$, but also can be easily identified since they consist of buckets that are "close enough" to S.

Making use of locality of $\mathcal{W}_i$'s. We implement this plan and derive a local version of Algorithm 1, listed in Algorithm 2. This algorithm aims to simulate the MIS, $I^{\text{seq}}_{i,S}$, for every $0 \le i \le d$ and $S \in \mathcal{W}_i$. The difference to Algorithm 1 is that it only uses the restricted $\mathcal{W}'_j = \{S' \in \mathcal{W}_j : \text{dist}(S', S) \le O(d^{2.5}) \cdot \tau\}$ for $j \le i-1$ in replacement of buckets in $\mathcal{W}_j$. Next, we show that this choice of $\mathcal{W}'_j = \{S' \in \mathcal{W}_j : \text{dist}(S', S) \le O(d^{2.5}) \cdot \tau\}$ indeed suffices for simulating Algorithm 1 locally.

Algorithm 2 Localized τ-MIS algorithm for the set P satisfying Property 4.2

let f be the hash and let $\mathcal{W}_0, \ldots, \mathcal{W}_d$ be the groups of buckets from Lemma 3.1 and Fact 4.5
▷ so each $\mathcal{W}_i$ is an $(O(d^{1.5})\tau, \tau)$-decomposition
for $i \leftarrow 0, \ldots, d$ **do**
  for $S \in \mathcal{W}_i$ **do**
    for $j \le i-1$, let $\mathcal{W}'_j \leftarrow \{S' \in \mathcal{W}_j : \text{dist}(S', S) \le O(d^{2.5}) \cdot \tau\}$ be the relevant set of buckets from $\mathcal{W}_j$
    let $I^{\text{loc}}_0 \leftarrow \emptyset$
    for $j \leftarrow 0, \ldots, i-1$ **do**
      let $I^{\text{loc}}_j \leftarrow I^{\text{loc}}_{j-1} \cup \bigcup_{S' \in \mathcal{W}'_j} \text{MIS}_\tau(S' \cap P \setminus B(I^{\text{loc}}_{j-1}, \tau))$
    end for
    let $I^{\text{loc}}_{i,S} \leftarrow \text{MIS}_\tau(S \cap P \setminus B(I^{\text{loc}}_{i-1}, \tau))$
  end for
end for
return $\bigcup_{0 \le i \le d, S \in \mathcal{W}_i} I^{\text{loc}}_{i,S}$

Lemma 4.7. *For every $0 \le i \le d$, $S \in \mathcal{W}_i$, we have $I^{\text{seq}}_{i,S} = I^{\text{loc}}_{i,S}$, where I^{seq} is as in Algorithm 1 and I^{loc} is as in Algorithm 2.*

Proof. Consider some $0 \le i \le d$ and $S \in \mathcal{W}_i$. By definition, $I^{\text{seq}}_{i,S} = \text{MIS}_\tau(S \cap P \setminus B(I^{\text{seq}}_{i-1}, \tau)) = \text{MIS}_\tau(S \cap P \setminus B(\bigcup_{j \le i-1, S' \in \mathcal{W}_j} I^{\text{seq}}_{j,S'}, \tau))$. By the first equality, we know that $I^{\text{seq}}_{i,S} \subseteq S$, and by the second equality, we can see that $I^{\text{seq}}_{i,S}$ only depends on $I^{\text{seq}}_{j,S'}$'s with $j \le i-1, S' \in \mathcal{W}_j$ such that $B(I^{\text{seq}}_{j,S'}, \tau) \cap S \ne \emptyset$, which is equivalent to $\text{dist}(I^{\text{seq}}_{j,S'}, S) \le \tau$.

Now, apply this argument again on each abovementioned $I^{\text{seq}}_{j,S'}$, it holds that such $I^{\text{seq}}_{j,S'}$ only depends on those $I^{\text{seq}}_{j',S''}$ with $j' \le j-1, S'' \in \mathcal{W}_{j'}$ such that $\text{dist}(S'', S') \le \tau$ which further implies $\text{dist}(S'', S) \le 2\tau + \text{diam}(S') \le 2\tau + O(d^{1.5})\tau$, where the last inequality holds by Fact 4.5. Apply this argument recursively, one can eventually conclude that $I^{\text{seq}}_{i,S}$ only depends on $I^{\text{seq}}_{j,S'}$'s such that $j \le i-1, S' \in \mathcal{W}_j$ and $\text{dist}(S', S) \le i \cdot \tau + (i-1)O(d^{1.5})\tau \le O(d^{2.5})\tau$, since there is at most $d+1$ groups.

Therefore, to evaluate $I^{\text{seq}}_{i,S}$, one only needs to know buckets $\mathcal{W}'_j = \{S' \in \mathcal{W}_j : \text{dist}(S', S) \leq O(d^{2.5}) \cdot \tau\}$, simulate Algorithm 1 on $\mathcal{W}'_j$ for $j \leq i-1$ to obtain some I', remove from S the points that are within I', and find $\text{MIS}_\tau(S \cap P \setminus B(I', \tau))$. Finally, to obtain $I^{\text{seq}}_{i,S} = I^{\text{loc}}_{i,S}$, it remains to ensure that the MIS on the same subset in both Algorithm 1 and Algorithm 2 are the same/consistent. Luckily, this can be trivially achieved since one can use a deterministic greedy maximal independent set algorithm for the purpose. This finishes the proof of Lemma 4.7. □

Lemma 4.8. *For each $0 \leq j \leq d-1$, $\sum_{S' \in \mathcal{W}'_j} |S' \cap P| \leq O(\varepsilon^{-1}d)^{O(d)}$ where $\mathcal{W}'_j$ is as in Algorithm 2 line 4.*

Proof. Consider some $0 \leq i \leq d$ and $S \in \mathcal{W}_i$. Fix a $j \leq i-1$. Recall that $\mathcal{W}'_j = \{S' \in \mathcal{W}_j : \text{dist}(S', S) \leq O(d^{2.5})\tau\}$ as in line 4. Since the buckets in the same $\mathcal{W}_{j'}$ for every j' are at least τ apart, we have that $\sum_{S' \in \mathcal{W}'_j} |S' \cap P| = |\bigcup_{S' \in \mathcal{W}'_j} (S' \cap P)|$. Fix any two points $x \neq y$ that come from buckets $S_1, S_2 \in \mathcal{W}'_j$ respectively. Recall that each bucket has a diameter $O(d^{1.5})\tau$. Then, by the triangle inequality and the definition of $\mathcal{W}'_j$, we have that

$$\text{dist}(x, y) \leq \text{diam}(S_1) + \text{dist}(S_1, S) + \text{diam}(S) + \text{dist}(S, S_2) + \text{diam}(S_2) \leq O(d^{2.5})\tau.$$

This means that the diameter of $\bigcup_{S' \in \mathcal{W}'_j} S'$ is at most $O(d^{2.5})\tau$. Then, by Property 4.2, we have that $|\bigcup_{S' \in \mathcal{W}'_j} (S' \cap P)| = |\bigcup_{S' \in \mathcal{W}'_j} S' \cap P| \leq O(\varepsilon^{-1}d)^{O(d)}$. This finishes the proof of Lemma 4.8. □

MPC implementation. Our final MPC algorithm is an MPC implementation of Algorithm 2. We do the two for-loops, i.e., for i and S, in parallel, and run the steps inside the for-loops, i.e., the steps to compute $I^{\text{loc}}_{i,S}$, only on a single machine. This procedure yields the same output as Algorithm 2, whose correctness has been analyzed in the above.

Space and round analysis. The local space usage is dominated by computing $I^{\text{loc}}_{i,S}$ (for some i and $S \in \mathcal{W}_i$) in a single machine, which is dominated by $O(d \cdot \sum_{S' \in \mathcal{W}'_j} |S' \cap P|)$, for $\mathcal{W}'_j$ in line 4. By Lemma 4.8 one concludes that for each $0 \leq j \leq d-1$, $\sum_{S' \in \mathcal{W}'_j} |S' \cap P| \leq O(\varepsilon^{-1}d)^{O(d)}$. This concludes the local space requirement. This also implies the total space is bounded by $O(n \,\text{poly}(d \log n)) \cdot O(\varepsilon^{-1}d)^{O(d)}$, since the computation of $\mathcal{W}'_j$ requires to copy data points. For the round complexity, since we do parallel-for loops, the round complexity is dominated by the steps inside each loop. These steps can be done in $O(\log_s n)$ rounds, since $\mathcal{W}'_j$ can be evaluated locally using the (data-oblivious) hash f, and other steps mostly require standard procedure of fetching the points from a certain bucket to a machine.

This finishes the proof of Lemma 4.1. □

5 MPC Algorithms for Approximate MDS in Low Dimension

Lemma 5.1. *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 $s \geq \Omega(\varepsilon^{-1}d)^{\Omega(d)} \,\text{poly}(\log n)$, computes a $(1+\varepsilon)\tau$-DS $S \subset \mathbb{R}^d$ for P such that $|S| \leq (1+\varepsilon)|\,\text{MDS}_\tau(P)|$ in $O(\log_s n)$ rounds, using total memory $O(n \,\text{poly}(\log n)) \cdot O(\varepsilon^{-1}d)^{O(d)}$.*

We make use of the same reduction as in Section 4, specifically Lemma 4.3, so that we can assume without loss of generality that P satisfies Property 4.2.

Proof overview. We give an outline of our MPC algorithm in Algorithm 3. The high level idea is to use the hash f as in Lemma 3.1, compute locally a MDS for each bucket, and then take the union of this as the approximate solution. This simple plan may not work well when data points are located around the "boundary" of the buckets, since the optimal solution may use only a few "boundary" points near the buckets to dominate the points inside the buckets.

To combat this issue, we plan to shift the buckets (implemented by hash f_v in line 4), and use an averaging argument to show there exists some good shift v, such that the boundary points contribute to the optimal solution very little. This averaging argument requires to use both a larger bucket diameter $\ell = O(d^{3.5}\varepsilon^{-1}\beta)$ (as opposed to the default $\ell = O(d^{1.5}\beta)$ as in Lemma 3.1), and a large enough support $\mathcal{V}^d$ of the shift. Both ℓ and $|\mathcal{V}^d|$ affect the space usage, and it is crucial to give good upper bounds for them. A key property is Fact 3.2, and this helps us to use only about $(d\varepsilon^{-1})^{O(d)}$ number of shifts.

Algorithm 3 Algorithm outline for $(1+\varepsilon)$-approximate $\mathrm{MDS}_\tau(P)$

1: let $\beta := 2\tau$, $\ell := O(d^{3.5}\varepsilon^{-1}\beta)$, $z := \ell/\sqrt{d}$, $b := d\beta + \tau$, $T := z/4b$, and $\mathcal{V} := \{0, 4b, \ldots, 4b(T-1)\}$

2: let $f : \mathbb{R}^d \to \mathbb{R}^d$ be a hash function with parameter β and ℓ as guaranteed by Lemma 3.1
3: **for** $v \in \mathcal{V}^d$ in parallel **do**
4: define $f_v : \mathbb{R}^d \to \mathbb{R}^d$ such that $f_v(x) := f(x+v)$ for $x \in \mathbb{R}^d$
5: for each bucket $u \in f_v(P)$, compute $\mathrm{MDS}_\tau(f_v^{-1}(u) \cap P)$
6: let $D_v \leftarrow \bigcup_{u \in f_v(P)} \mathrm{MDS}_\tau(f_v^{-1}(u) \cap P)$
7: **end for**
8: return $\widehat{M} := D_v$ such that $|D_v|$ is the minimum over all $v \in \mathcal{V}^d$

MPC implementation. We discuss how to implement Algorithm 3 in MPC. By Lemma 3.1, the hash in line 2 is deterministic and only takes $\mathrm{poly}(d)$ space, so it can be locally stored in every machine without communication. Next, we examine the lines in the parallel for-loop. Line 4 can be executed locally in each machine. Line 5 can be implemented by first sorting the points with respect to their hash value (i.e., bucket ID), then solving the MDS for each subset of points with the same hash value. Notice that since each bucket has diameter at most $\ell = O(d^{3.5}\varepsilon^{-1}\beta) \le O((d^{3.5}\varepsilon^{-1})\tau)$, it contains at most $O(\varepsilon^{-1}d)^{O(d)}$ points from P by Property 4.2, so all these points can fit in the same machine (due to the assumption on s). Therefore, their MDS can be solved locally without further communication. Line 6 requires removing duplicated elements in $\mathrm{MDS}_\tau(f_v^{-1}(u) \cap P)$ for $u \in f_v(P)$, which can be done using sorting. Finally, line 8 can be implemented using standard procedures, specifically counting and finding the minimum element.

Round and space complexity. By the above, the round complexity is $O(\log_s n)$ since every step can be implemented in $O(\log_s n)$ rounds and the for-loop is done in parallel. The local space requirement is dominated by line 5, which is $O(\varepsilon^{-1}d)^{O(d)} \cdot \mathrm{poly}(\log n)$. The total space is dominated by the parallel-for loop, which has $|\mathcal{V}|^d \le (d\varepsilon^{-1})^{O(d)}$ invocations, and this results in $O(n\,\mathrm{poly}(\log n)) \cdot O(\varepsilon^{-1}d)^{O(d)}$ total space.

Error analysis. Let $M^* := \mathrm{MDS}_\tau(P)$ be an optimal/minimum solution of τ-dominating set for P. Observe that the algorithm returns $\widehat{M}$, such that its size $|\widehat{M}| = \min_{v \in \mathcal{V}^d} |D_v|$. Hence, it suffices to show there exists $v^* \in \mathcal{V}^d$ such that $|D_{v^*}| \le (1+\varepsilon)|M^*|$. To do so, we start with analyzing $|D_v|$

for a generic v as follows, in order to obtain a more concrete condition of v^*.

$$\begin{aligned}
|D_v| &= \left| \bigcup_{u \in f_v(P)} \mathrm{MDS}_\tau(f_v^{-1}(u) \cap P) \right| \\
&\leq \sum_{u \in f_v(P)} |\,\mathrm{MDS}_\tau(f_v^{-1}(u) \cap P)| \\
&\leq \sum_{u \in f_v(P)} |M^* \cap N_\tau^\infty(f_v^{-1}(u))| \\
&\leq \sum_{u \in f_v(P)} |M^* \cap f_v^{-1}(u)| + |M^* \cap U_\tau^\infty(f_v^{-1}(u))| \\
&= |M^*| + \sum_{u \in f_v(P)} |M^* \cap U_\tau^\infty(f_v^{-1}(u))|. \qquad (2)
\end{aligned}$$

We next further analyze $\sum_{u \in f_v(P)} |M^* \cap U_\tau^\infty(f_v^{-1}(u))|$, and the plan is to use the properties of Lemma 3.1. However, those properties are for f instead of directly for f_v, and hence we need the following Fact 5.2 that relates f_v^{-1} and f^{-1}.

Fact 5.2. *For every $v \in \mathbb{R}^d$, the image set of f_v, i.e., $f_v(\mathbb{R}^d)$, equals that of f, and for every image $u \in f_v(\mathbb{R}^d)$, $f_v^{-1}(u) = f^{-1}(u) - v$.*

By Fact 5.2,

$$\sum_{u \in f_v(P)} |M^* \cap U_\tau^\infty(f_v^{-1}(u))| \leq \sum_{u \in f_v(\mathbb{R}^d)} |M^* \cap U_\tau^\infty(f_v^{-1}(u))| \leq \sum_{u \in f(\mathbb{R}^d)} |M^* \cap (U_\tau^\infty(f^{-1}(u)) - v)|. \quad (3)$$

We next apply Fact 3.2 to further upper bound (3), i.e.,

$$\sum_{u \in f(\mathbb{R}^d)} |M^* \cap (U_\tau^\infty(f^{-1}(u)) - v)| \leq (d+1) \cdot \left| \bigcup_{u \in f(\mathbb{R}^d)} M^* \cap (U_\tau^\infty(f^{-1}(u)) - v) \right|. \quad (4)$$

To see this, for $x \in \mathbb{R}^d$, if some u satisfies $x \in U_\tau^\infty(f^{-1}(u))$, then there exists $y \in f^{-1}(u)$ such that $\|x - y\|_\infty \leq \tau$, which is equivalent to $N_\tau^\infty(x) \cap f^{-1}(u) \neq \emptyset$. Since $\mathrm{diam}_\infty(N_\tau^\infty(x)) \leq 2\tau = \beta$, by Fact 3.2, $|\{u : N_\tau^\infty(x) \cap f^{-1}(u) \neq \emptyset\}| = |f(N_\tau^\infty(x))| \leq d+1$. Therefore, every point $x \in \mathbb{R}^d$ can only participate in at most $d+1$ terms of $\sum_{u \in f(\mathbb{R}^d)} |M^* \cap (U_\tau^\infty(f^{-1}(u)) - v)|$, and this implies (4).

Then, applying Fact 5.2 and the third property of Lemma 3.1 to the U_τ^∞'s of (4), we have

$$\bigcup_{u \in f(\mathbb{R}^d)} (U_\tau^\infty(f^{-1}(u)) - v) \subseteq \overline{L(z, 2b)^d} - v,$$

where $L(z, 2b) = \bigcup_{a \in \mathbb{Z}} [az + 2b, (a+1)z - 2b)$, and $\overline{L(z, 2b)^d}$ is the complement (with respect to $\mathbb{R}^d$) of $L(z, 2b)^d$. Therefore,

$$\left| \bigcup_{u \in f(\mathbb{R}^d)} M^* \cap (U_\tau^\infty(f^{-1}(u)) - v) \right| \leq \left| M^* \cap (\overline{L(z, 2b)^d} - v) \right|. \quad (5)$$

Hence, it suffices to show there exists $v^* \in \mathcal{V}^d$, such that $|M^* \cap (\overline{L(z, 2b)^d} - v^*)| \leq \varepsilon |M^*|$. This v^* may be viewed as a "shift" of $\overline{L(z, 2b)^d}$, and we would use an averaging argument to show the existence of it. For $j \in \mathcal{V}$, let $\hat{L}(j) := \bigcup_{a \in \mathbb{Z}} [az - 2b - j, az + 2b - j)$. Notice that $\hat{L}(j)$'s are of small length which is $4b$, and $\{\hat{L}(j) : j \in \mathcal{V}\}$ is a partition of $\mathbb{R}$. Moreover, it can be observed that $\hat{L}(0) = \overline{L(z, 2b)}$.

Picking v^*. For $i \in [d]$, define multiset $\mathrm{proj}_i(M^*) := \{x_i : x = (x_1, \ldots, x_d) \in M^*\}$ as taking the i-th coordinates of points in M^*. Now, by an averaging argument (over $j \in \mathcal{V}$), for every $i \in [d]$, there exists j_i^* such that $|\mathrm{proj}_i(M^*) \cap \hat{L}(j_i^*)| \le \frac{1}{|\mathcal{V}|} \cdot |M^*|$. We pick $v^* := (j_1^*, \ldots, j_d^*)$.

Concluding error bound. We continue to analyze (5) when plugging in $v = v^*$.

$$\begin{aligned} |M^* \cap (\overline{L(z, 2b)^d} - v^*)| &\le \sum_{i=1}^{d} |\mathrm{proj}_i(M^*) \cap (\hat{L}(0) - j_i^*)| \\ &= \sum_{i=1}^{d} |\mathrm{proj}_i(M^*) \cap \hat{L}(j_i^*)| \\ &\le \frac{d}{|\mathcal{V}|}|M^*| = \frac{d}{T}|M^*|. \end{aligned}$$

Combining this with (2), (3) and (4), we conclude that

$$|\widehat{M}| = \min_{v \in \mathcal{V}^d} |D_v| \le |D_{v^*}| \le |M^*| + \frac{d(d+1)}{T}|M^*| \le (1+\varepsilon)|M^*|.$$

This finishes the proof of Lemma 5.1. □

6 MPC Algorithms for RS in High Dimension

Lemma 6.1. *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 $s \ge \mathrm{poly}(d \log n)$, 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 total memory $O(n^{1+\varepsilon} \mathrm{poly}(d \log n))$.*

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 4, 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 [Gha22]) 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 [CGJ+24] (restated in Lemma 6.8). 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 4 and 5.

Algorithm 4 Preprocessing, with input $P \subseteq \mathbb{R}^d$ of n points, $\beta \ge 1, \tau > 0$

1: let φ be a consistent hashing with parameter $\Gamma \leftarrow O(d/\log \log n)$ $\Lambda \leftarrow \mathrm{poly}(d \log n)$ 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 $\mathrm{rep}(z)$
3: return $P' \leftarrow \mathrm{rep}(\varphi(P))$

Lemma 6.2. *R is a τ-independent set for P' (with probability 1).*

Algorithm 5 Local algorithm for RS on P' resultant from Algorithm 4, 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 $A^{\beta}_{P'}(p, \tau)$
3: return R

Proof. Fix a point $x \in R$. Suppose for the contrary that there is a point $y \neq x \in R$ satisfying $\mathrm{dist}(x, y) \leq \tau$. Then we have that $y \in B(x, \tau)$ and $x \in B(y, \tau)$. Observe that $R \subseteq P'$ which implies that $x, y \in P'$. Since for every $p \in P'$, $B_{P'}(p, \tau) \subseteq A^{\beta}_{P'}(p, \tau)$, then we have that $y \in B_{P'}(x, \tau) \subseteq A^{\beta}_{P'}(x, \tau)$ and $x \in B_{P'}(y, \tau) \subseteq A^{\beta}_{P'}(y, \tau)$. However, we have $h(x) = \min(h(A^{\beta}_{P'}(x, \tau)))$ and $h(y) = \min(h(A^{\beta}_{P'}(y, \tau)))$ by line 2 of Algorithm 5, which implies that $h(x) < h(y)$ and $h(y) < h(x)$, respectively, leading to a contradiction. Therefore, we complete the proof. □

Lemma 6.3. *For any $\alpha \geq 1$, any $(\tau, \alpha\tau)$-ruling set for P' is an $(\tau, (\alpha + \beta\Gamma)\tau)$-ruling set for P.*

Proof. Let S be a $(\tau, \alpha\tau)$-ruling set for P'. Since $S \subseteq P' \subseteq P$ and S is τ-independent by definition, it remains to verify for every $p \in P$, $\mathrm{dist}(p, S) \leq (\alpha + \ell)\tau$. Fix some $p \in P$. For a generic point set $W \subseteq \mathbb{R}^d$ and a point $x \in \mathbb{R}^d$, let $W(x)$ be the nearest point in W to x. Then

$$\mathrm{dist}(p, S) \leq \mathrm{dist}(p, P'(p)) + \mathrm{dist}(P'(p), S(p)) \leq \mathrm{dist}(p, \mathrm{rep}(\varphi(p))) + \alpha\tau \leq \ell + \alpha\tau \leq (\alpha + \beta\Gamma)\tau.$$

□

Fact 6.4. *For every $p \in P'$, $|A^{\beta}_{P'}(x, \tau)| \leq \Lambda$.*

Proof. Follows immediately from the definition of parameters Γ, ℓ, Λ and Lemma 3.4. □

Lemma 6.5. *For every $t \geq \Omega(\log^2 \Lambda)$, the set R returned by Algorithm 5 satisfies*

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

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

Proof. It suffices to show that for every point $p \in P'$, with probability $1 - \exp(-\Omega(t) \log(t/\log \Lambda))$, $\mathrm{dist}(p, R) \leq O(\beta t)\tau$, 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 6, 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 6 is defined with respect to the internal states of a specific (random) run of Algorithm 5, and it does not introduce new randomness.

Algorithm 6 Finding an assignment sequence $S = (x_0 = p, \ldots, x_T)$, for a given $p \in P'$

1: let $i \leftarrow 0$, $x_0 \leftarrow p$, $S \leftarrow (x_0)$
2: **while** Algorithm 5 does not add x_i at line 2 **do**
3: let x_{i+1} be the point from $A^{\beta}_{P'}(x_i, \tau)$ with the smallest label
4: let $S \leftarrow S \circ x_{i+1}$ and $i \leftarrow i + 1$
5: **end while**
6: return S

Observe that in Algorithm 6, 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

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

by triangle inequality. Hence, it remains to give an upper bound for T, and we show this in the following Lemma 6.6 which is the main technical lemma.

Lemma 6.6. *For $t \geq \Omega(\log^2 \Lambda)$, we have $\Pr[T \geq t] \leq \exp(-\Omega(t) \log(t/\log \Lambda))$.*

Proof. The proof is postponed in Section 6.1. □

Finally, as mentioned, applying Lemma 6.6 with a union bound finishes the proof of Lemma 6.5. □

Proof of Lemma 6.1. Our MPC algorithm for Lemma 6.1 is obtained via an MPC implementation of the offline Algorithms 4 and 5. However, before we do so, our algorithm requires to run the Johnson-Lindenstrauss (JL) transform [JL84] 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 6.7 (JL transform [JL84])**.** *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 $1 - 1/\operatorname{poly}(n)$, $\forall x, y \in P$, $\operatorname{dist}(g(x), g(y)) \in (1 \pm \varepsilon) \operatorname{dist}(x, y)$.*

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 4 and 5 in MPC. For Algorithm 4, since the hash in Lemma 3.4 that we use is data oblivious and only requires $\operatorname{poly}(d) = \operatorname{poly}(\log n)$ space, all machines can generate the same hash without communication. The other steps in Algorithm 4 may be implemented using standard MPC procedures including sorting, broad-casting and converge-casting. For Algorithm 5, the the only nontrivial step is to implement $A^{\beta}_{P'}(\cdot, \cdot)$. To this end, we make use of the following lemma from [CGJ+24]. In particular, our MPC implementation of Algorithm 5 applies Lemma 6.8 with $\beta = O(\varepsilon^{-1})$. This finishes the description of the algorithm for Lemma 6.1.

Lemma 6.8 ([CGJ+24, 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 $s \geq \operatorname{poly}(d \log n)$, 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*

$$B_P(p, \tau) \subseteq A_P(p, \tau) \subseteq B_P(p, O(\varepsilon^{-1}\tau)), \tag{6}$$

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

Now we turn to the analysis. Observe that the assumptions regarding $A_{P'}(\cdot,\cdot)$ underlying the analysis of Algorithms 4 and 5 remain valid. By Lemma 6.2, 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 $\Lambda = \mathrm{poly}(\log n)$. Therefore, applying Lemma 6.5 with $t = O(\log n/\log\log n)$, we conclude that $\forall p \in P'$, $\mathrm{dist}(p,R) \le O(\varepsilon^{-1}\log n/\log\log n)\tau$, with probability $1 - 1/\mathrm{poly}(n)$. Combining with Lemma 6.3, we conclude that R is $(\tau, O(\varepsilon^{-1}\log n/\log\log n)\tau)$-ruling set for P, with probability $1 - 1/\mathrm{poly}(n)$.

Finally, for the round complexity, local memory and total memory, these are dominated by the parallel invocations of Lemma 6.8, which are $O(\log_s n)$, $\mathrm{poly}(d\log n)$ and $O(n^{1+\varepsilon}\,\mathrm{poly}(d\log n))$, respectively. Therefore, we complete the proof of Lemma 6.1. □

6.1 Proof of Lemma 6.6: Upper Bound The Length of Auxiliary Sequence

Let S be the (random) sequence returned by Algorithm 6, and recall that we write $S = (x_0 = p, \ldots, x_T)$. For two sequences S' and S'', let $S' \circ S''$ denote their concatenation; when $S'' = (x)$ is a singleton sequence, we write $S' \circ x$, which is a shorthand for appending the point x to the sequence S'. In addition, let $S' \sqsubseteq S''$ denote that S' is a prefix of S''.

Let $\mathcal{S}$ be the set of all sequences $S' := (x'_0, x'_1, \ldots)$ such that: (1) $x'_0 = p$; (2) for every $i \ge 0$, $x_{i+1} \in A^{\beta}_{P'}(x_i, \tau)$. Consider some $S' \in \mathcal{S}$ denoted as $S' := (x'_0, x'_1, \ldots, x'_m)$ and $m \ge 1$. Define

$$\begin{aligned} A_{\mathrm{cond}}(S') &:= \bigcup_{i=0}^{m-1} A^{\beta}_{P'}(x'_i, \tau) \\ A_{\mathrm{new}}(S') &:= A^{\beta}_{P'}(x'_m, \tau) \setminus A_{\mathrm{cond}}(S'). \end{aligned} \tag{7}$$

Let $\mathcal{E}_{\mathrm{cond}}(S')$ denote the event that $S' \sqsubseteq S$, i.e., Algorithm 6 gets to line 2 with the current sequence equal to S'. Let $\mathcal{E}_{\mathrm{ext}}(S')$ denote the event that $\mathcal{E}_{\mathrm{cond}}(S')$ holds and in iteration $i = m$ of Algorithm 6, x'_m passes the test in line 2 and the point picked in line 3 is from $A^{\beta}_{P'}(x'_m, \tau)$. The next lemma establishes an upper bound for the probability of the sequence gets a new element appended, with respect to $\frac{|A_{\mathrm{new}}(S')|}{|A_{\mathrm{new}}(S') \cup A_{\mathrm{cond}}(S')|}$ which may be understood as the local geometric growth of the dataset.

Lemma 6.9. *For every $S' \in \mathcal{S}$, we have that*

$$\Pr[\mathcal{E}_{\mathrm{ext}}(S') \mid \mathcal{E}_{\mathrm{cond}}(S')] \le \frac{|A_{\mathrm{new}}(S')|}{|A_{\mathrm{new}}(S') \cup A_{\mathrm{cond}}(S')|}.$$

Proof. Let $S' := (x'_0 = p, x'_1, \ldots, x'_m) \in \mathcal{S}$ be a fixed sequence with a length of $m+1$. Observe that the event $\mathcal{E}_{\mathrm{cond}}(S')$ may depend on the random label of points in the set $A_{\mathrm{cond}}(S') = \bigcup_{i=0}^{m-1} A^{\beta}_{P'}(x'_i, \tau)$, but not on those of points outside $A_{\mathrm{cond}}(S')$. The label choices for points outside $A_{\mathrm{cond}}(S')$ and inside $A_{\mathrm{cond}}(S')$ are independent, hence by the principle of deferred decisions, conditioning on $\mathcal{E}_{\mathrm{cond}}(S')$ does not change the distribution of label choices for points in $A_{\mathrm{new}}(S') = A^{\beta}_{P'}(x'_m, \tau) \setminus A_{\mathrm{cond}}(S')$, which in turn affect the event $\mathcal{E}_{\mathrm{ext}}(S')$ (e.g., whether x'_m passes the test).

When $\mathcal{E}_{\mathrm{cond}}(S')$ occurs, each point x'_i, for $i \in [m]$, has the smallest label in $A^{\beta}_{P'}(x'_{i-1}, \tau)$, and it follows by induction that x'_m has the smallest label in $A_{\mathrm{cond}}(S')$. When $\mathcal{E}_{\mathrm{ext}}(S')$ occurs, some $x'_{m+1} \ne x'_m$ has the smallest label in $A^{\beta}_{P'}(x'_m, \tau)$, and thus $h(x'_{m+1}) < h(x'_m) = \min(h(A_{\mathrm{cond}}(S')))$, implying that $x'_{m+1} \notin A_{\mathrm{cond}}(S')$ and in fact $x'_{m+1} \in A_{\mathrm{new}}(S')$. Therefore,

$$\Pr[\mathcal{E}_{\mathrm{ext}}(S') \mid \mathcal{E}_{\mathrm{cond}}(S')] \le \sum_{q \in A_{\mathrm{new}}(S')} \Pr[h(q) = \min(h(A^{\beta}_{P'}(x'_m, \tau))) \mid \mathcal{E}_{\mathrm{cond}}(S')]. \tag{8}$$

Claim 6.10 ([CGJ+24], Claim 4.13)**.** *Fix a finite domain $\mathcal{U}$, and let $g : \mathcal{U} \to [0,1]$ be random, such that each $g(a)$ is chosen independently and uniformly from $[0,1]$. Then for every subset $X \subseteq \mathcal{U}$, element $a \in X$, and partial order $\mathcal{P}$ on $X \setminus \{a\}$, we have*

$$\Pr_g[g(a) = \min(g(X)) \mid g \propto \mathcal{P}] = 1/|X|,$$

where $g \propto \mathcal{P}$ denotes consistency in the sense that $g(x) \le g(y)$ whenever $x \prec_{\mathcal{P}} y$.

To upper bound (8), it suffices to give for every $q \in A_{\text{new}}(S')$ an upper bound for $\Pr[h(q) = \min(h(A^{\beta}_{P'}(x'_m, \tau))) \mid \mathcal{E}_{\text{cond}}(S')]$. Notice the event $\mathcal{E}_{\text{cond}}(S')$ defines a partial order via the label h, such that for $i \in [m]$, each point x'_i has the smallest label in $A^{\beta}_{P'}(x'_{i-1}, \tau)$, i.e., for each $y \neq x'_i \in A^{\beta}_{P'}(x'_{i-1}, \tau)$, $h(x'_i) < h(y)$, and this is in particular a partial order on $A_{\text{new}}(S') \cup A_{\text{cond}}(S') \setminus \{q\}$. Now, apply Claim 6.10 with $g := h$, $a := q$, $X := A_{\text{new}}(S') \cup A_{\text{cond}}(S')$ which contains a, and a partial order $\mathcal{P}$ defined from the event $\mathcal{E}_{\text{cond}}(S')$ (as mentioned above). Hence,

$$\forall q \in A_{\text{new}}(S'), \quad \Pr[h(q) = \min(h(A^{\beta}_{P'}(x'_m, \tau))) \mid \mathcal{E}_{\text{cond}}(S')] = \frac{1}{|A_{\text{new}}(S') \cup A_{\text{cond}}(S')|}.$$

Plugging this into (8), we obtain that $\Pr[\mathcal{E}_{\text{ext}}(S') \mid \mathcal{E}_{\text{cond}}(S')] \le \frac{|A_{\text{new}}(S')|}{|A_{\text{new}}(S') \cup A_{\text{cond}}(S')|}$, and this completes the proof of Lemma 6.9. □

We continue the proof of Lemma 6.6. Let $f : \mathcal{S} \to \mathbb{Z}$ be a function such that for each $S' \in \mathcal{S}$,

$$f(S') := \begin{cases} i & |A_{\text{new}}(S')| \in [2^{i-1}, 2^i) \\ -\infty & A_{\text{new}}(S') = \emptyset. \end{cases}$$

The following fact follows from Fact 6.4.

Fact 6.11. *For any $S' \in \mathcal{S}$, we have $f(S') \in \{-\infty\} \cup [\lceil \log_2 \Lambda \rceil]$.*

Intuitively, this function f takes value i when $A_{\text{new}}(S')$ is around $[2^{i-1}, 2^i)$. Now, for a given sequence S', we consider the *configuration* of S' as the f values for every prefix of S'. Namely, for every $S' \in \mathcal{S}$ denoted as $S' := (x'_0, x'_1, \ldots, x'_m)$ and $i \ge 0$, let $S'_{[0,i]}$ denote the prefix of S' ending with x'_i, and denote $\text{conf}(S') := (f(S'_{[0,1]}), \ldots, f(S'_{[0,m]}))$ as the *configuration* of S'. For $\pi \in (\{-\infty\} \cup [\lceil \log_2 \Lambda \rceil])^i$ (for some $i \ge 1$), let $\mathcal{S}_\pi := \{S' \in \mathcal{S} : \text{conf}(S') = \pi\}$.

The next lemma is a key lemma which upper bounds the probability that a sequence has length at least t, for sequences of a fixed configuration π. This lemma would be used to conclude Lemma 6.6 by a union bound over all possible configurations $\pi \in (\{-\infty\} \cup [\lceil \log_2 \Lambda \rceil])^t$.

Lemma 6.12. *For every $\pi \in (\{-\infty\} \cup [\lceil \log_2 \Lambda \rceil])^t$, $\sum_{S' \in \mathcal{S}_\pi} \Pr[S' \sqsubseteq S] \le \exp(-\Omega(t \log(t/\log \Lambda)))$.*

Proof. For every $i \in [t]$, let $\pi_{\le i}$ denote the prefix of π with a length of i, and let π_i denote the ith item of π. Observe that if $\pi_i = -\infty$ for $i < t$, then $\Pr[S' \sqsubset S] = 0$ for any $S' \in \mathcal{S}_\pi$. Hence, we only need to verify the claim for a $\pi \in (\{-\infty\} \cup [\lceil \log_2 \Lambda \rceil])^t$ such that $\pi_i \neq -\infty$ for every $i < t$.

Now, for $i \ge 2$, we start with proving

$$\sum_{S' \in \mathcal{S}_{\pi_{\le i}}} \Pr[S' \sqsubseteq S] \le \sum_{S'' \in \mathcal{S}_{\pi_{\le i-1}}} \Pr[\exists x \in P', S'' \circ x \sqsubseteq S]. \tag{9}$$

For every $i \in [t]$, let $\phi_i : \mathcal{S}_{\pi_{\le i}} \to \{(S'', x) : S'' \in \mathcal{S}_{\pi_{\le i-1}}, x \in P'\}$ be a mapping, where for $S' \in \mathcal{S}_{\pi_{\le i}}$, $\phi_i(S')$ maps to the unique pair (S'', x) satisfying $S' = S'' \circ x$ (i.e., $S'' = S'_{[0,i-1]}$). Then for every

$S' \in \mathcal{S}_{\pi_{\le i}}$, $\Pr[S' \sqsubseteq S] = \Pr[S'' \circ x \sqsubseteq S]$ where $(S'', x) = \phi_i(S')$. Moreover, ϕ_i is an injection by definition. Therefore,

$$\begin{aligned}\sum_{S' \in \mathcal{S}_{\pi_{\le i}}} \Pr[S' \sqsubseteq S] &= \sum_{\substack{(S'',x):\\ (S'',x)=\phi_i(S'), S' \in \mathcal{S}_{\pi_{\le i}}}} \Pr[S'' \circ x \sqsubseteq S] \\ &\le \sum_{S'' \in \mathcal{S}_{\pi_{\le i-1}}} \sum_{x \in P'} \Pr[S'' \circ x \sqsubseteq S] \\ &= \sum_{S'' \in \mathcal{S}_{\pi_{\le i-1}}} \Pr[\exists x \in P', S'' \circ x \sqsubseteq S],\end{aligned}$$

where the inequality uses that ϕ_i is injection. This concludes (9).

Therefore, we have that

$$\begin{aligned}\sum_{S' \in \mathcal{S}_{\pi_{\le i}}} \Pr[S' \sqsubseteq S] &\le \sum_{S'' \in \mathcal{S}_{\pi_{\le i-1}}} \Pr[\exists x \in P', S'' \circ x \sqsubseteq S] \\ &= \sum_{S'' \in \mathcal{S}_{\pi_{\le i-1}}} \Pr[\exists x \in P', S'' \circ x \sqsubseteq S \mid S'' \sqsubseteq S] \cdot \Pr[S'' \sqsubseteq S] \\ &= \sum_{S'' \in \mathcal{S}_{\pi_{\le i-1}}} \Pr[\mathcal{E}_{\text{ext}}(S'') \mid \mathcal{E}_{\text{cond}}(S'')] \cdot \Pr[S'' \sqsubseteq S] \\ &\le \sum_{S'' \in \mathcal{S}_{\pi_{\le i-1}}} \frac{|A_{\text{new}}(S'')|}{|A_{\text{new}}(S'') \cup A_{\text{cond}}(S'')|} \cdot \Pr[S'' \sqsubseteq S] \\ &\le \sum_{S'' \in \mathcal{S}_{\pi_{\le i-1}}} \frac{2^{1+\pi_{i-1}}}{\sum_{j=1}^{i-1} 2^{\pi_j}} \cdot \Pr[S'' \sqsubseteq S] \qquad (10)\end{aligned}$$

where the second inequality follows from Lemma 6.9, and the last inequality follows from (7). The assumption that $\pi_i = -\infty$ may happen only when $i = t$ ensures that the denominators $\sum_{j=1}^{i-1} 2^{\pi_j}$ cannot be zero. Applying (10) inductively,

$$\sum_{S' \in \mathcal{S}_{\pi_{\le t}}} \Pr[S' \sqsubseteq S] \le \prod_{i=1}^{t-1} \frac{2^{1+\pi_i}}{\sum_{j=1}^{i} 2^{\pi_j}}.$$

For every $i \in [\lceil \log_2 \Lambda \rceil]$, let $G_i := \{\pi_j = i : j \in [t-1]\}$ and let $w_i := |G_i|$. Then

$$\prod_{i=1}^{t-1} \frac{2^{1+\pi_i}}{\sum_{j=1}^{i} 2^{\pi_j}} \le \prod_{i \in [\lceil \log_2 \Lambda \rceil]} \frac{2^{(1+i)w_i}}{2^{iw_i} \cdot w_i!} \le \prod_{i \in [\lceil \log_2 \Lambda \rceil]} \frac{2^{w_i}}{w_i!} \le \prod_{i \in [\lceil \log_2 \Lambda \rceil]} O(\sqrt{w_i}) \cdot \frac{2^{w_i}}{(w_i/e)^{w_i}},$$

where the first inequality follows from the fact that

$$\begin{aligned}\prod_{i=1}^{t-1} \sum_{j=1}^{i} 2^{\pi_j} \ge \prod_{i \in [\lceil \log_2 \lambda \rceil]} \prod_{j \in G_i} \sum_{s=1}^{j} 2^{\pi_s} \ge \prod_{i \in [\lceil \log_2 \lambda \rceil]} \prod_{j \in G_i} \sum_{s \in [j]: s \in G_i} 2^{\pi_s} &= \prod_{i \in [\lceil \log_2 \lambda \rceil]} \prod_{j \in [w_i]} j 2^i \\ &\ge \prod_{i \in [\lceil \log_2 \lambda \rceil]} 2^{iw_i} w_i!\end{aligned}$$

Taking logarithm,

$$\begin{aligned}
\log \prod_{i=1}^{t-1} \frac{2^{1+\pi_i}}{\sum_{j=1}^{i} 2^{\pi_j}} &\le \log \prod_{i\in[\lceil \log_2 \Lambda\rceil]} O(\sqrt{w_i}) \cdot \frac{2^{w_i}}{(w_i/e)^{w_i}} \\
&= \sum_{i\in[\lceil \log_2 \Lambda\rceil]} \log\left(O(\sqrt{w_i}) \cdot \frac{2^{w_i}}{(w_i/e)^{w_i}} \right) \\
&\le \sum_{i\in[\lceil \log_2 \Lambda\rceil]} O(\log(w_i) + w_i - w_i \log(w_i/e)) \\
&\le \sum_{i\in[\lceil \log_2 \Lambda\rceil]} O(-w_i \log(w_i)) \\
&\le O(-t \log(t/\log \Lambda)),
\end{aligned}$$

where the last inequality follows from the fact that $\sum_i w_i \le t$ and Jensen's inequality. We conclude that $\log \sum_{S'\in\mathcal{S}_{\pi_{\le t}}} \Pr[S' \sqsubseteq S] \le O(-t \log(t/\log \Lambda))$, and this finishes the proof. □

Proof of Lemma 6.6. Finally, we conclude Lemma 6.6 by a union-bound argument.

$$\begin{aligned}
\Pr[T \ge t] &= \sum_{S'\in\mathcal{S}:|S'|=t+1} \Pr[S' \sqsubseteq S] \\
&= \sum_{\pi\in[\lceil \log_2 \Lambda\rceil]^t} \sum_{S'\in\mathcal{S}_\pi} \Pr[S' \sqsubseteq S] \\
&\le (\log \Lambda)^{O(t)} \cdot \exp(-\Omega(t \log(t/\log \Lambda))) \\
&\le \exp(O(t) \log\log \Lambda - \Omega(t) \log(t/\log \Lambda)) \\
&\le \exp(-\Omega(t) \log(t/\log \Lambda)),
\end{aligned}$$

where the first inequality is by applying Lemma 6.12, and the last inequality uses that $t \ge \Omega(\log^2 \Lambda)$. This completes the proof of Lemma 6.6. □

7 Proof of Theorems 1.1 to 1.3: MPC Algorithms for k-Center

We present the proofs for Theorems 1.1 to 1.3. As mentioned, we obtain these results by reducing to geometric versions of either RS or MDS. Hence, we present the reduction as a meta algorithm, as in Algorithm 7, instead of separated concrete algorithms. The concrete algorithms can be derived by plugging in suitable combination of RS or MDS results.

We note that this meta algorithm only finds a set of *centers* as approximate solution, and we discuss in Section 7.2 how to also find the approximate *assignment* from data points to the found center set. This procedure introduces a slight loss in the ratio, but it is only minor; in particular, in low dimension the ratio increases by only a $(1+\varepsilon)$ factor, and in high dimension the ratio increases by a constant factor.

Algorithm. The algorithm, listed in Algorithm 7, takes as input a dataset $P \subseteq \mathbb{R}^d$, integer $k \ge 1$ and a parameter $0 < \varepsilon < 1$. The algorithm starts with testing if $\mathrm{OPT} = 0$, which is equivalent to P has at most k distinct points. If this does not happen, the algorithm "guesses" OPT (up to $(1+\varepsilon)$ factor). This step requires to first obtain a $\mathrm{poly}(n)$-approximate value APX for OPT using Lemma 7.2 (which can be found in Section 7.1), and this is one of the randomness in the algorithm. Then we search for a suitable threshold τ and apply Lemmas 4.1, 5.1 and 6.1 to compute a center set.

Algorithm 7 MPC Algorithms for k-Center via RS and MDS with parameter $0 < \varepsilon < 1$

1: return P as the solution if P contains at most k distinct points
2: compute an $\alpha := \mathrm{poly}(n)$-approximate value APX for k-Center on P using Lemma 7.2
3: let $Z \leftarrow \{i \in \mathbb{Z} : \mathrm{APX}/\alpha \le (1+\varepsilon)^i \le \mathrm{APX}\}$ be a set of integer power of $(1+\varepsilon)$
4: **for** $\tau \in Z$ in parallel **do**
5: (RS-based approx.) find $I_\tau \leftarrow (2\tau, \gamma\tau)$-ruling set for P by Lemma 4.1 or Lemma 6.1
▷ where $\gamma = 2(1+\varepsilon)$ as in Lemma 4.1, or $\gamma = O(\varepsilon^{-1}\frac{\log n}{\log\log n})$ as in Lemma 6.1
6: (MDS-based approx.)
Let D_τ be $(1+\varepsilon)\tau$-DS with size $|D_\tau| \le (1+\varepsilon)|\mathrm{MDS}_\tau(P)|$ by Lemma 5.1
7: **end for**
8: (RS-based approx.) let $\tau^* \in Z$ be the smallest τ such that $|I_\tau| \le k$, return I_{τ^*}
9: (MDS-based approx.) let $\tau^* \in Z$ be the smallest τ such that $|D_\tau| \le (1+\varepsilon)k$, return D_{τ^*}

Round complexity analysis. Line 1 can be implemented in $O(\log_s n)$ MPC rounds via sorting and broadcasting. The number of rounds for line 2 is dominated by Lemma 7.2, which is $O(\log_s n)$ rounds. Observe that $|Z| = O(\log n)$, so there are $O(\log n)$ parallel invocations of the parallel-for. Hence, the number of rounds for the parallel-for is of the same order as in Lemma 4.1, Lemma 5.1 and Lemma 6.1, which is $O(\log_s n)$. This finishes the round analysis.

Space complexity analysis. For the space, it is dominated by that of Lemma 4.1, Lemma 5.1 and Lemma 6.1, multiplied by $|Z|$ which is the number of parallel iterations (since they need to be replicated). By Lemmas 4.1 and 5.1, we conclude that the local space requirement is $s \ge \Omega(\varepsilon^{-1}d)^{\Omega(d)}\,\mathrm{poly}(\log n)$, and the total space is $O(n\,\mathrm{poly}(\log n))\cdot O(\varepsilon^{-1}d)^{O(d)}$ for $(2+\varepsilon)$-approximation and $(1+\varepsilon, 1+\varepsilon)$-approximation, which concludes the space analysis for both Theorems 1.1 and 1.2. By Lemma 6.1, we conclude that the local space requirement is $s \ge \mathrm{poly}(d\log n)$, and the total space is $O(n^{1+\varepsilon}\,\mathrm{poly}(d\log n))$ for $O(\varepsilon^{-1}\log n/\log\log n)$-approximation, which completes the space analysis of Theorem 1.3. This concludes the space analysis.

Error analysis: the common part. As we mentioned, if the algorithm terminates at line 1, then $\mathrm{OPT} = 0$ and the entire dataset itself is the optimal solution. Now assume the dataset consists of more than k distinct points, which means $\mathrm{OPT} > 0$. As in line 2 and 3, we use Lemma 7.2 to obtain $\mathrm{poly}(n)$-approximation to OPT and it succeeds with probability at least $1 - 1/n$. Hence, with probability at least $1 - 1/n$, there exists some $\hat{\tau} \in Z$ such that $\mathrm{OPT} \le \hat{\tau} \le (1+\varepsilon)\,\mathrm{OPT}$. We condition on this high-probability event happens in the remainder of the proof.

Error analysis for RS-based approximation. We start with justifying that the algorithm is well-defined, i.e., τ^* in line 8 must exist. In particular, we claim that if $\tau \ge \mathrm{OPT}(P)$ then $|I_\tau| \le k$. This claim implies the existence of τ^* since $\tau = \hat{\tau}$ satisfies $\tau \ge \mathrm{OPT}$, which also implies

$$\tau^* \le \hat{\tau} \le (1+\varepsilon)\,\mathrm{OPT}\,. \tag{11}$$

Now, consider some $\tau \ge \mathrm{OPT}$ (noticing that $\hat{\tau}$ satisfies this). Then we make use of the following standard fact (proved for completeness in Fact 7.1, see also [HS86]), and specifically apply it with $S = P$, $\mu = \tau \ge \mathrm{OPT}$. It readily implies $|I_\tau| \le k$ since I_τ is a 2τ-independent set for P.

Fact 7.1. *Let $S \subseteq \mathbb{R}^d$. For $\mu \ge \mathrm{OPT}(S)$, any subset of S that is a 2μ-independent set for S has at most k points.*

Proof. Suppose for the contrary that there is some 2μ-independent set $I \subseteq S$ for S with $|I| \geq k+1$. Consider a clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ in an optimal solution for k-Center on S. Observe that each cluster $C_i \in \mathcal{C}$ has diameter $\operatorname{diam}(C_i) \leq 2\operatorname{OPT}(S) \leq 2\mu$. Now, consider any two distinct points $x \neq y \in I$. Since $\operatorname{dist}(x,y) > 2\mu$ by definition, they cannot belong to the same cluster. Therefore, since $|I| \geq k+1$, there must be a point in I that does not belong to any cluster of $\mathcal{C}$, leading to a contradiction (since $I \subseteq S$ and $\bigcup_{i=1}^{k} C_i = S$). This finishes the proof of Fact 7.1. □

Since $|I_{\tau^*}| \leq k$, I_{τ^*} is a feasible solution for k-Center. It remains to analyze $\operatorname{cost}(P, I_{\tau^*})$. Since I_{τ^*} is a $(2\tau^*, \gamma\tau^*)$-ruling set for P, we know that $\operatorname{cost}(P, I_{\tau^*}) \leq \gamma\tau^* \leq \gamma(1+\varepsilon)\operatorname{OPT}$, where the last inequality follows from (11). This finishes the error analysis for the $(2+\varepsilon)$-approximation and $O(\varepsilon^{-1}\log n/\log\log n)$-approximation by substituting the corresponding values of γ.

Error analysis for MDS-based approximation. Similarly, we start with showing that the algorithm is well-defined, i.e., τ^* in line 9 must exist, by showing that if $\tau \geq \operatorname{OPT}$ then $|D_\tau| \leq (1+\varepsilon)k$. This further implies $\tau^* \leq \hat{\tau} \leq (1+\varepsilon)\operatorname{OPT}$ (and hence τ^* exists).

To see this, let C^* be the optimal solution to k-Center on the input P. Then for every point $x \in P$, $\operatorname{dist}(x, C^*) \leq \operatorname{OPT} \leq \tau$. Hence, C^* is a τ-DS for P. Therefore,

$$|D_\tau| \leq (1+\varepsilon)|\operatorname{MDS}_\tau(P)| \leq (1+\varepsilon)|C^*| \leq (1+\varepsilon)k,$$

where the first inequality is by Lemma 5.1 and the second inequality follow from the optimality of MDS_τ.

Since the algorithm always returns a set of at most $(1+\varepsilon)k$ points, it remains to show $\operatorname{cost}(P, D_{\tau^*}) \leq (1+O(\varepsilon))\operatorname{OPT}$. Now, since $\tau^* \leq (1+\varepsilon)\operatorname{OPT}$, and by the fact that D_{τ^*} is a $(1+\varepsilon)\tau^*$-DS for P, we have

$$\operatorname{cost}(P, D_{\tau^*}) = \max_{x \in P} \operatorname{dist}(x, D_{\tau^*}) \leq (1+\varepsilon)\tau^* \leq (1+O(\varepsilon))\operatorname{OPT}.$$

This finishes the proof of Theorems 1.1 to 1.3. □

7.1 Coarse Approximation for Optimal Value of k-Center

Lemma 7.2. *There is a randomized MPC algorithm, that given a dataset P of n points in $\mathbb{R}^d$ distributed across MPC machines with local memory $s \geq \Omega(\operatorname{poly}(d\log n))$, computes $E \geq 0$ such that $\operatorname{OPT} \leq E \leq O(n^7)\operatorname{OPT}$ for k-Center with probability at least $1-1/n$, in $O(\log_s n)$ rounds using total memory $O(n\operatorname{poly}(d\log n))$.*

Proof. Let $v \in \mathbb{R}^d$ be an i.i.d. standard Gaussian vector, i.e., each entry of v satisfies $v_i \sim \mathcal{N}(0,1)$. For a point set $P \subset \mathbb{R}^d$, we do the inner product to every $x \in P$ with respect to v, and denote $P' := \{\langle v, x\rangle : \forall x \in P\}$ as this set. This set can be viewed as the 1D projection of P.

We claim that, the event "for every $x, y \in P$, $|\langle x, v\rangle - \langle y, v\rangle| \in [\Omega(1/n^3), O(n^3)]\operatorname{dist}(x,y)$" happens with probability $1-1/n$. To see this claim, fix some $x, y \in P$. Observe that $\frac{\langle x,v\rangle - \langle y,v\rangle}{\operatorname{dist}(x,y)} \sim \mathcal{N}(0,1)$. Plug this into the following standard fact about Gaussian, and do a union bound for all pairs x, y yields the claim.

Fact 7.3. *For some $z \geq 1$ and $u \sim \mathcal{N}(0,1)$, we have that $\Pr[|u| \leq 1/z] \leq O(1/z)$, and $\Pr[|u| \geq z] \leq O(1/z)$.*

Hence, it suffices to approximate k-Center within $O(n)$ factor on the 1D input P' (where the distance is defined as the absolute value of the difference). It would be convenient (and without loss of generality) to assume all points are distinct and there are at least $n \geq k+1$ of them.

If $k = 1$, then we simply define the maximum minus the minimum as the estimation E', and this is 2-approximation. Otherwise, for $k \geq 2$, we sort P' with respect to their value/coordinate in 1D, and we find the k-th largest gap between consecutive points, denoted as r_k, and we define $E' := n \cdot r_k$.

We claim that E' satisfies $\Omega(\mathrm{OPT}(P')) \leq E' \leq O(n)\,\mathrm{OPT}(P')$. On one hand, $\mathrm{OPT}(P')$ must be at least $r_k/2$ for otherwise it requires at least $k+1$ centers to cover the consecutive point pairs whose gap is at least r_k. This implies that $E' \leq O(n) \cdot \mathrm{OPT}(P')$.

On the other hand, we show how to construct a feasible solution whose cost is at most $O(n \cdot r_k)$, and this would imply $\mathrm{OPT}(P') \leq \mathrm{cost}(P', C_{\mathcal{P}}) \leq O(n \cdot r_k) = O(E')$. Let $\mathcal{P} := \{(p_i, q_i)\}_i$ be the consecutive point pairs (after the above sorting) such that for each pair $(p_i, q_i) \in \mathcal{P}$, $\mathrm{dist}(p_i, q_i) > r_k$, and order (p_i, q_i) by $p_i < q_i$ (in 1D coordinate). We have $|\mathcal{P}| \leq k-1$, since otherwise, there would be at least k consecutive point pairs with gaps greater than r_k, which contradicts the definition of r_k as the k-th largest gap between consecutive points. Let $i_{\max} := \mathrm{argmax}_{j:(p_j,q_j)\in\mathcal{P}}\, p_j$, and let $q_{\max} := q_{i_{\max}}$, i.e., $q_{\max}$ is the larger point in the pair whose smaller point is the largest among all consecutive pairs. Then define $C_{\mathcal{P}} := \{p_i : (p_i, q_i) \in \mathcal{P}\} \cup \{q_{\max}\}$, so $|C_{\mathcal{P}}| \leq k$.

We analyze the cost of this $C_{\mathcal{P}}$. Let x_{end} be the last point in P', and let $X := \{x \in P' : q_{\max} < x < x_{\mathrm{end}}\}$ be the set of all points in P' that lie between $q_{\max}$ and x_{end}, then $|X| \leq n$. Write $X := \{x_1, x_2, \ldots\}$ such that $x_i < x_{i+1}$ for all $i \in [|X|-1]$. Then, by triangle inequality, we have that $\mathrm{dist}(x_{\mathrm{end}}, q_{\max}) \leq \mathrm{dist}(q_{\max}, x_1) + \sum_{i=1}^{|X|-1} \mathrm{dist}(x_i, x_{i+1}) + \mathrm{dist}(x_{|X|}, x_{\mathrm{end}}) \leq O(n \cdot r_k)$, where the last inequality holds by the definition of $\mathcal{P}$. This implies that for any point in P' after $p_{\max}$, the distance to $C_{\mathcal{P}}$ is at most $O(n \cdot r_k)$. Similarly, we conclude that for any point in P' before $p_{\max}$, the distance to its nearest point in $C_{\mathcal{P}}$ is also at most $O(n \cdot r_k)$. This implies that $\mathrm{cost}(P', C_{\mathcal{P}}) \leq O(n \cdot r_k)$.

The final value we return is $E := \Theta(n^3 E')$, to make sure the returned value is at least OPT. This entire algorithm is straightforward to implement in MPC, and the complexity is dominated by sorting. □

7.2 Finding Approximate Assignments

We would assume the context in the above proof, and discuss how to find an approximate clustering for k-Center in low and high dimensions, respectively (i.e., for Theorems 1.1 to 1.3).

Lemma 7.4 (Assignment for k-Center in low dimension)**.** *There is an MPC algorithm, that given $\varepsilon \in (0,1)$, $\tau > 0$, a center set C and a dataset P of n points in $\mathbb{R}^d$ distributed across MPC machines with local memory $s \geq \Omega(\varepsilon^{-1}d)^{\Omega(d)} \mathrm{poly}(\log n)$, such that $\mathrm{cost}(P, C) = \Theta(\tau)$, computes an assignment for each point $x \in P$, mapping it to a center $c \in C$ such that $\mathrm{dist}(x, c) \leq (1 + O(\varepsilon))\,\mathrm{cost}(P, C)$, in $O(\log_s n)$ rounds using total memory $O(n\,\mathrm{poly}(d \log n)) \cdot O(\varepsilon^{-1}d)^{O(d)}$.*

Proof. Write $C := (c_1, \ldots, c_k)$. Let $\phi : P \cup C \to \mathcal{G}_{\varepsilon\tau/\sqrt{d}}$, that maps each point in P or C to its nearest neighbor in the $\varepsilon\tau/\sqrt{d}$-grid $\mathcal{G}_{\varepsilon\tau/\sqrt{d}}$. Let $P_\tau := \phi(P)$ and $C_\tau := \phi(C)$ be the sets rounded to the grid. Before defining the clustering for the original P based on C, our first step finds the *exact* nearest neighbor assignment from P_τ to C_τ. The details of this part will be discussed later. Now, assuming that the exact nearest neighbor assignment from P_τ to C_τ is obtained, we proceed to define how to assign points in P to C and prove that this assignment satisfies the approximation bound.

For every $z \in \phi(C)$, pick $\mathrm{rep}(z)$ as an arbitrary fixed point in $\phi^{-1}(z) \cap C$, i.e., a representative point in C among points whose nearest neighbor in $\mathcal{G}_{\varepsilon\tau/\sqrt{d}}$ is the same grid. Now, for a point $x \in P$, let $C_\tau(\phi(x))$ denote the nearest neighbor of $\phi(x)$ in C_τ (which is known by assumption), then we assign x to $\mathrm{rep}(C_\tau(\phi(x)))$. This step can be implemented within the claimed round and space as in Lemma 7.4.

In this assignment, by triangle inequality,

$$\begin{aligned}\operatorname{dist}(x, C) &\le \operatorname{dist}(x, \phi(x)) + \operatorname{dist}(\phi(x), C_\tau(\phi(x))) + \operatorname{dist}(C_\tau(\phi(x)), \operatorname{rep}(C_\tau(\phi(x))))\\ &\le \varepsilon\tau + \operatorname{dist}(\phi(x), C_\tau) + \varepsilon\tau\\ &\le 2\varepsilon\tau + \operatorname{dist}(x, C) + 2\varepsilon\tau\\ &\le (1 + O(\varepsilon)) \operatorname{cost}(P, C).\end{aligned}$$

This finishes the description of the assignment from P to C and the analysis of its approximation bound.

Now, we turn to define the exact nearest neighbor assignment from P_τ to C_τ. The algorithm simply computes, for every $x \in P_\tau$ the nearest center point $c \in B(x, O(\tau)) \cap C_\tau$, i.e., only searching the nearest point in an $O(\tau)$ neighborhood of x. This algorithm is well-defined, since for every $x \in P_\tau$, $\operatorname{dist}(x, C_\tau) \le \operatorname{dist}(x, C) + \varepsilon\tau \le 2\varepsilon\tau + \operatorname{cost}(P, C) \le O(\tau)$. This ensures that the found c is the exact nearest neighbor of x. Observe that for every $x \in P_\tau$, we have $|B(x, O(\tau)) \cap C_\tau| \le (\varepsilon^{-1}d)^{O(d)}$, by Lemma 2.1. Hence, this step of finding the nearest neighbors of P_τ in C_τ can be done in MPC in $O(\log_s n)$ rounds, $(\varepsilon^{-1}d)^{O(d)} \operatorname{poly}(\log n)$ local space and $(\varepsilon^{-1}d)^{O(d)} n \operatorname{poly}(\log n)$ total space. This finishes the proof. □

Lemma 7.5 (Assignment for k-Center in high dimension)**.** *There is an MPC algorithm, that given $\varepsilon \in (0, 1)$, $\tau > 0$, a center set C and a dataset P of n points in $\mathbb{R}^d$ distributed across MPC machines with local memory $s \ge \operatorname{poly}(d \log n)$, such that $\operatorname{cost}(P, C) = \Theta(\tau)$, with probability at least $1 - 1/n$, computes an assignment for each point $x \in P$, mapping it to a center $c \in C$ such that $\operatorname{dist}(x, c) \le O(\varepsilon^{-1}) \operatorname{cost}(P, C)$, in $O(\log_s n)$ rounds using total memory $O(n^{1+\varepsilon} \operatorname{poly}(d \log n))$.*

Proof. We make use of a procedure described in [CGJ+24, Section 3.1]. Specifically, in [CGJ+24, Section 3.1], for a given parameter $r > 0$, their MPC algorithm computes for each data point $x \in P$ an approximate center point $c_x \in C$ such that $\operatorname{dist}(x, c_x) \le O(\varepsilon^{-1})r$ if there exists $c \in C$ such that $\operatorname{dist}(x, c) \le r$, in $O(\log_s n)$ rounds, $\operatorname{poly}(d \log n)$ local space and $n^{1+\varepsilon} \operatorname{poly}(d \log n)$ global space.

Now, in our case, since $\operatorname{cost}(P, C) = \Theta(\tau)$, then there must exist a point $c \in C$ for each point $x \in P$ within a distance of $O(\tau)$. Hence, applying the abovementioned algorithm with $r := O(\tau)$, we can find for each $x \in P$ a center point $c_x \in C$ such that $\operatorname{dist}(x, c_x) \le O(\varepsilon^{-1}) \operatorname{cost}(P, C)$, and achieve the claimed round and space complexity as in Lemma 7.5. □

References

[AAH+23] AmirMohsen Ahanchi, Alexandr Andoni, MohammadTaghi Hajiaghayi, Marina Knittel, and Peilin Zhong. Massively parallel tree embeddings for high dimensional spaces. In *SPAA*, pages 77–88, 2023. doi:10.1145/3558481.3591096.

[ABJ+25] Amir Azarmehr, Soheil Behnezhad, Rajesh Jayaram, Jakub Lacki, Vahab Mirrokni, and Peilin Zhong. Massively parallel minimum spanning tree in general metric spaces. In *SODA*, pages 143–174, 2025. doi:10.1137/1.9781611978322.5.

[AG23] Sepideh Aghamolaei and Mohammad Ghodsi. A 2-approximation algorithm for data-distributed metric k-center. *arXiv preprint arXiv:2309.04327*, 2023. arXiv:2309.04327.

[AGLP89] Baruch Awerbuch, Andrew V Goldberg, Michael Luby, and Serge A Plotkin. Network decomposition and locality in distributed computation. In *FOCS*, pages 364–369, 1989. doi:10.1109/SFCS.1989.63504.

[BBM23] Mark de Berg, Leyla Biabani, and Morteza Monemizadeh. k-center clustering with outliers in the MPC and streaming model. In *IPDPS*, pages 853–863, 2023. arXiv:2302.12811.

[BEFM21] MohammadHossein Bateni, Hossein Esfandiari, Manuela Fischer, and Vahab S. Mirrokni. Extreme k-center clustering. In *AAAI*, pages 3941–3949, 2021. doi:10.1609/AAAI.V35I5.16513.

[BKS17] 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.

[BP24] Leyla Biabani and Ami Paz. k-center clustering in distributed models. In *SIROCCO*, pages 83–100, 2024. doi:10.1007/978-3-031-60603-8_5.

[BW18] Aditya Bhaskara and Maheshakya Wijewardena. Distributed clustering via LSH based data partitioning. In *ICML*, pages 569–578, 2018. URL: http://proceedings.mlr.press/v80/bhaskara18a.html.

[CCM23] Sam Coy, Artur Czumaj, and Gopinath Mishra. On parallel k-center clustering. In *SPAA*, pages 65–75, 2023. doi:10.1145/3558481.3591075.

[CFJ+22] 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 [CJK+22]. arXiv:2204.02095.

[CGJ+24] Artur Czumaj, Guichen Gao, Shaofeng H.-C. Jiang, Robert Krauthgamer, and Pavel Veselý. Fully-scalable MPC algorithms for clustering in high dimension. In *ICALP*, pages 50:1–50:20, 2024. doi:10.4230/LIPICS.ICALP.2024.50.

[CJK+22] Artur Czumaj, Shaofeng H.-C. Jiang, Robert Krauthgamer, Pavel Veselý, and Mingwei Yang. Streaming facility location in high dimension via geometric hashing. In *FOCS*, pages 450–461, 2022. doi:10.1109/FOCS54457.2022.00050.

[CKPU23] Mélanie Cambus, Fabian Kuhn, Shreyas Pai, and Jara Uitto. Time and space optimal massively parallel algorithm for the 2-ruling set problem. In *DISC*, pages 11:1–11:12, 2023. doi:10.4230/LIPICS.DISC.2023.11.

[CLN+21] 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.

[CMZ22] Vincent Cohen-Addad, Vahab S. Mirrokni, and Peilin Zhong. Massively parallel k-means clustering for perturbation resilient instances. In *ICML*, pages 4180–4201, 2022. URL: https://proceedings.mlr.press/v162/cohen-addad22b.html.

[CPP19] 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. URL: http://www.vldb.org/pvldb/vol12/p766-ceccarello.pdf.

[DG08] 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.

[EIM11] Alina Ene, Sungjin Im, and Benjamin Moseley. Fast clustering using MapReduce. In *KDD*, pages 681–689, 2011. doi:10.1145/2020408.2020515.

[EMMZ22] Alessandro Epasto, Mohammad Mahdian, Vahab S. Mirrokni, and Peilin Zhong. Massively parallel and dynamic algorithms for minimum size clustering. In *SODA*, pages 1613–1660. SIAM, 2022. doi:10.1137/1.9781611977073.66.

[Fil24] Arnold Filtser. Scattering and sparse partitions, and their applications. *ACM Transactions on Algorithms*, 20(4):30:1–30:42, 2024. doi:10.1145/3672562.

[GGJ20] Mohsen Ghaffari, Christoph Grunau, and Ce Jin. Improved MPC algorithms for MIS, matching, and coloring on trees and beyond. In *DISC*, pages 34:1–34:18, 2020. doi:10.4230/LIPICS.DISC.2020.34.

[Gha19] Mohsen Ghaffari. Massively parallel algorithms, 2019. Lecture Notes from ETH Zürich. URL: http://people.csail.mit.edu/ghaffari/MPA19/Notes/MPA.pdf.

[Gha22] Mohsen Ghaffari. Distributed graph algorithms, 2022. Lecture Notes from MIT. URL: https://people.csail.mit.edu/ghaffari/DA22/Notes/DGA.pdf.

[GLM+23] 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 *SPAA*, pages 443–453, 2023. doi:10.1145/3558481.3591098.

[Gon85] 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.

[GP24] Jeff Giliberti and Zahra Parsaeian. Massively parallel ruling set made deterministic. In *DISC*, pages 29:1–29:21, 2024. doi:10.4230/LIPICS.DISC.2024.29.

[GSZ11] Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the MapReduce framework. In *ISAAC*, pages 374–383, 2011. arXiv:1101.1902.

[GU19] Mohsen Ghaffari and Jara Uitto. Sparsifying distributed algorithms with ramifications in massively parallel computation and centralized local computation. In *SODA*, pages 1636–1653, 2019. doi:10.1137/1.9781611975482.99.

[Har04] Sariel Har-Peled. No, coreset, no cry. In *FSTTCS*, pages 324–335, 2004. doi:10.1007/978-3-540-30538-5_27.

[HM04] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *STOC*, pages 291–300, 2004. doi:10.1145/1007352.1007400.

[HS85] 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.

[HS86] 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.

[HZ23] Alireza Haqi and Hamid Zarrabi-Zadeh. Almost optimal massively parallel algorithms for k-center clustering and diversity maximization. In *SPAA*, pages 239–247, 2023. doi:10.1145/3558481.3591077.

[IBY+07] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: Distributed data-parallel programs from sequential building blocks. In *EuroSys*, pages 59–72, 2007. doi:10.1145/1272996.1273005.

[IKL+23] 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.

[IM15] Sungjin Im and Benjamin Moseley. Brief announcement: Fast and better distributed mapreduce algorithms for k-center clustering. In *SPAA*, pages 65–67, 2015. doi:10.1145/2755573.2755607.

[JKPS25] Hongyan Ji, Kishore Kothapalli, Sriram V. Pemmaraju, and Ajitanshu Singh. Fast deterministic massively parallel ruling sets algorithms. In *ICDCN*, pages 152–160, 2025. doi:10.1145/3700838.3700872.

[JL84] William Johnson and Joram Lindenstrauss. Extensions of Lipschitz maps into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 01 1984. doi:10.1090/conm/026/737400.

[JLN+05] Lujun Jia, Guolong Lin, Guevara Noubir, Rajmohan Rajaraman, and Ravi Sundaram. Universal approximations for TSP, Steiner tree, and set cover. In *STOC*, pages 386–395, 2005. doi:10.1145/1060590.1060649.

[JMNZ24] Rajesh Jayaram, Vahab Mirrokni, Shyam Narayanan, and Peilin Zhong. Massively parallel algorithms for high-dimensional Euclidean minimum spanning tree. In *SODA*, pages 3960–3996. SIAM, 2024. doi:10.1137/1.9781611977912.139.

[KSV10] Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for MapReduce. In *SODA*, pages 938–948, 2010. doi:10.1137/1.9781611973075.76.

[LFW+24] Ting Liang, Qilong Feng, Xiaoliang Wu, Jinhui Xu, and Jianxin Wang. Improved approximation algorithm for the distributed lower-bounded k-center problem. In *TAMC*, pages 309–319, 2024. doi:10.1007/978-981-97-2340-9_26.

[Lub85] Michael Luby. A simple parallel algorithm for the maximal independent set problem. In *STOC*, pages 1–10. ACM, 1985. doi:10.1145/22145.22146.

[MKC+15] 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. URL: https://proceedings.neurips.cc/paper/2015/hash/8fecb20817b3847419bb3de39a609afe-Abstract.html.

[Ona18] Krzysztof Onak. Round compression for parallel graph algorithms in strongly sublinear space. *arXiv preprint arXiv:1807.08745*, 2018. arXiv:1807.08745.

[Pol90] 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.

[RG20] Václav Rozhoň and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *STOC*, pages 350–363, 2020. doi:10.1145/3357713.3384298.

[RVW18] 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, November 2018. doi:10.1145/3232536.

[SW10] Johannes Schneider and Roger Wattenhofer. An optimal maximal independent set algorithm for bounded-independence graphs. *Distributed Computing*, 22(5-6):349–361, 2010. URL: https://doi.org/10.1007/s00446-010-0097-1, doi:10.1007/S00446-010-0097-1.

[Whi15] Tom White. *Hadoop: The Definitive Guide*. O'Reilly, 4th edition, 2015. URL: https://www.oreilly.com/library/view/hadoop-the-definitive/9781491901687/.

[ZCF+10] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *HotCloud*, 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 $u \in f(\mathbb{R}^d)$, $\mathrm{diam}(f^{-1}(u)) \leq \ell$.*
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 [CFJ+22, 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 abovementioned 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 $B_i := N_{\ell_i}^\infty(A_i)$

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

be ℓ_i-neighborhood of A_i (with respect to ℓ_∞ distance). Let $B_{-1} := \emptyset$ and $B_{\le i} := \bigcup_{j \le i} B_j$. The main procedure for the construction of f goes as follows.

- For every $i \in \{0, \dots, d-1\}$, for every $Q \in \mathcal{F}_i$, let $\widehat{Q} := N^\infty_{\ell_i}(Q) \setminus B_{\le i-1}$, 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 $\widehat{Q} := Q \setminus B_{\le d-1}$ 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 $0 \le i \le d-1$, define $\mathcal{W}_i := \{N^\infty_{\ell_i}(Q) \setminus B_{\le i-1} : Q \in \mathcal{F}_i\}$ and let $\mathcal{W}_d := \{Q \setminus B_{\le d-1} : Q \in \mathcal{F}_d\}$. 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 [CFJ+22, 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 ([CFJ+22, Lemma 5.5 and Lemma 5.8])**.** *For every $0 \le i \le d$ and every $S \ne S' \in \mathcal{W}_i$, $\mathrm{dist}(S, S') \ge \mathrm{dist}_\infty(S, S') > \beta$.*

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 $\bigcup_u U^\infty_\tau(f^{-1}(u))$ to a much simpler object $N^\infty_b(A_{d-1})$, which is merely the b-neighborhood of $(d-1)$-dimensional faces.

Claim A.2. $\bigcup_{u \in f(\mathbb{R}^d)} U^\infty_\tau(f^{-1}(u)) \subseteq N^\infty_b(A_{d-1})$.

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 $i \in \{0, \dots, d\}$ and for every $S \in \mathcal{W}_i$, $U^\infty_\tau(S) \subseteq N^\infty_b(A_{d-1})$. We prove this separately for the case of $i \le d-1$ and $i = d$.

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

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

$$N^\infty_\tau(S) \cap S' = \emptyset, \tag{12}$$

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 $B_{\le d-1} \cap \bigcup_{S \in \mathcal{W}_d} S = \emptyset$. By (12), $N^\infty_\tau(S)$ has no intersection with $\mathcal{W}_d$ other than on S. Hence,

$$U^\infty_\tau(S) = N^\infty_\tau(S) \setminus S \subseteq B_{\le d-1}. \tag{13}$$

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

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

where the last step follows from $A_j \subseteq A_{j+1}$ for every j. Combining this with (13), we conclude that $U^{\infty}_{\tau}(S) \subseteq N^{\infty}_{d\beta}(A_{d-1}) \subseteq N^{\infty}_{b}(A_{d-1})$. This finishes the proof of Claim A.2. □

By Claim A.2, it remains to prove $N^{\infty}_{b}(A_{d-1}) \cap L(z, 2b)^d = \emptyset$. Since $N^{\infty}_{b}(A_{d-1}) = N^{\infty}_{b}(\bigcup_{Q\in\mathcal{F}_{d-1}} Q) = \bigcup_{Q\in\mathcal{F}_{d-1}} N^{\infty}_{b}(Q) = \bigcup_{x\in Q, Q\in\mathcal{F}_{d-1}} N^{\infty}_{b}(x)$, it suffices to show $\forall x \in Q$ (for $Q \in \mathcal{F}_{d-1}$), $N^{\infty}_{b}(x) \cap L(z, 2b)^d = \emptyset$. 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 $0 \le j \le d$ and $\{a_i \in \mathbb{Z}\}_{i=0}^{d}$ such that for $0 \le i \le d$,

$$x_i \in \begin{cases} [a_i z, (a_i+1)z) & i \neq j \\ \{a_i z\} & i = j \end{cases}. \tag{14}$$

Let j be that as in (14). Then for every $y \in L(z, 2b)^d$,

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

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 $\forall x' \in N^{\infty}_{b}(x)$ and $y \in L(z, 2b)^d$, $\|x' - y\|_\infty \ge b$, by the triangle inequality of ℓ_∞. Therefore, we conclude that $N^{\infty}_{b}(x) \cap L(z, 2b)^d = \emptyset$, and this finishes the proof of Lemma 3.1. □

B Analysis of One-round Luby's Algorithm

Here we show both an upper and a matching lower bound for one-round Luby's algorithm for graphs. We give a proof sketch, in Appendix B.1, for an upper bound that the set S returned by the one-round Luby's algorithm is $O(\log n)$-ruling set with high probability. We give a tight input graph G, such that the S returned by the one-round Luby's algorithm is $\Omega(\log n)$-ruling set with high probability, in Appendix B.2.

Here, for an undirected graph $G = (V, E)$, an α-ruling set is a subset $S \subseteq V$ such that a) for any $x \neq y \in S$, $\{x, y\} \notin E$, and b) any vertex $x \in V$ has a path within α hops to S. The one-round Luby's algorithm picks a uniform random value $h(x) \in [0, 1]$ for every $x \in V$, and include x into S if and only if x has the smallest h value among x and its adjacent vertices, i.e., $\{x\} \cup \{y \in V : \{x, y\} \in E\}$. Return S as the output.

B.1 Upper Bound: A Proof Sketch

We discuss how the proof in Section 6 can be modified to prove the $O(\log n)$-ruling set bound in graphs. Even though we still use the language of Euclidean spaces in the following, the proof sketch actually does not use Euclidean property and works directly for graphs.

As we are talking about one-round Luby's algorithm, we do not need to do the preprocessing in Algorithm 4, and the in Algorithm 5 we do not need to use the approximate $A^{\beta}_{P}(\cdot, \cdot)$.

The proof of Lemma 6.2 still shows the returned point set is τ-independent. Lemma 6.3 and fact 6.4 are not needed, as we work on P directly (without preprocessing). Lemma 6.5 is

still the main lemma, and the statement can be simply changed to for $t = O(\log n)$, $\Pr[\forall p \in P, \operatorname{dist}(p, R) \le O(t)\tau] \ge 1 - 1/\operatorname{poly}(n)$. The proof plan of Lemma 6.5 still goes through, and it reduces to Lemma 6.6, where the statement is changed to for $t = O(\log n)$, $\Pr[T \ge t] \le 1/\operatorname{poly}(n)$.

The main modification is in the proof of Lemma 6.6. Lemma 6.9 is still useful even in the graph setting, and does not need to be changed. However, we use a new definition for f, and instead of mapping to $\mathbb{Z}$, we do $f : \mathcal{S} \to \{0, 1\}$. Pick some parameter $\gamma = \Theta(1)$. For $S' \in \mathcal{S}$, define

$$f(S') := \begin{cases} 1 & \frac{|A_{\text{new}}(S')|}{|A_{\text{new}}(S') \cup A_{\text{cond}}(S')|} > \gamma \\ 0 & \text{otherwise} \end{cases}$$

Intuitively, $f(S')$ is 1 if the upper bound $\frac{|A_{\text{new}}(S')|}{|A_{\text{new}}(S') \cup A_{\text{cond}}(S')|}$ of the extension probability (Lemma 6.9) is too large. Ideally, if for a sequence all extensions have f value 0, then its length is greater than t with probability $1/\operatorname{poly}(n)$. The definition of *configuration* remains the same, i.e., for $S' \in \mathcal{S}$, i.e., $\operatorname{conf}(S') := (f(S'_{[0,1]}), \ldots, f(S'_{[0,m]}))$.

Next, we need a new lemma that shows for every $\pi \in \{0, 1\}^t$ and $S' \in \mathcal{S}_\pi$, $\|\operatorname{conf}(S')\|_1 \le t/2$ (which requires the constant in the big-O of t to be picked carefully). This lemma holds because once f value is 1, the $|A_{\text{new}}|$ (which is the new part introduced by the extension) is at least constant fraction of the neighborhood of all previous elements in the sequence. Therefore, this cannot happen more than $t/2 = \Omega(\log n)$ times since otherwise $|A_{\text{new}}(S'_{[0,t/2]})| > n$.

Finally, in the modified version of Lemma 6.12, we use the abovementioned $\|\operatorname{conf}(S')\|_1 \le t/2$ bound, and conclude that there are still at least $t/2 = \Omega(\log n)$ number of extensions that happen with probability at most $\frac{|A_{\text{new}}(S')|}{|A_{\text{new}}(S') \cup A_{\text{cond}}(S')|} \le \gamma$, and hence $\sum_{S' \in \mathcal{S}_\pi} \Pr[S' \sqsubseteq S] \le \gamma^{t/2}$.

Finally, since the number of configurations is at most 2^t, taking a union bound, $2^t \cdot \gamma^{t/2} \le 1/\operatorname{poly}(n)$ by setting small enough γ.

B.2 Lower Bound

Let n be some parameter. Consider the following graph $G = (V, E)$. Let $m := \frac{\ln n}{\gamma}$ where γ is a sufficiently large constant. The vertex set V consists of m parts $V_1, \ldots, V_m$, i.e., $V := \bigcup_i V_i$, such that $|V_i| = 2^i$. We add for every $i \in [m]$, a clique in each V_i, i.e., $V_i \times V_i$, and add for every $i \in [m-1]$ a complete bipartite graph for between V_i and V_{i+1}, i.e., $V_i \times V_{i+1}$.

Now, consider the one-round Luby's algorithm on G, and we analyze the random values h. For $i \in [m-1]$, let $\mathcal{E}_i$ be the event that there exists $u \in V_{i+1}$ such that $h(u) < h(v)$ for all $v \in V_i$.

Claim. $\Pr[\bigwedge_{i \in [m-1]} \mathcal{E}_i] \ge 1/\sqrt{n}$.

Proof.

$$\begin{aligned}
\Pr[\bigwedge_{i \in [m-1]} \mathcal{E}_i] &= \prod_{i \in [m-1]} \Pr[\mathcal{E}_i \mid \bigwedge_{j \in [i-1]} \mathcal{E}_j] \\
&= \prod_{i \in [m-1]} \Pr[\mathcal{E}_i] \\
&= \prod_{i \in [m-1]} \left(1 - \left(1 - \frac{1}{2^i + 1}\right)^{2^{i+1}}\right) \\
&\ge (1 - 1/e^2)^m \\
&\ge 1/\sqrt{n}.
\end{aligned}$$

To see the second equality, consider $\Pr[\mathcal{E}_i \mid \bigwedge_{j\in[i-1]} \mathcal{E}_j]$ for some $i \in [m-1]$. Then the values h for the vertices in V_{i+1} and $\bigcup_{j\in[i]} V_j$ are independent, and that the event $\bigwedge_{j\in[i-1]} \mathcal{E}_j$ only depends on the randomness of h for the vertices in $\bigcup_{j\in[i]} V_j$. The third equality follows from the fact that the h values for vertices in V_{i+1} are independent to each other, and that for a fixed vertex $u \in V_{i+1}$ the probability that $h(u) < h(v)$ for every $v \in V_i$ is $1/(2^i + 1)$. □

Observe that the event $\bigwedge_{i\in[m-1]} \mathcal{E}_i$ implies that one-round Luby's returns $\Omega(\log n)$-ruling set.

Hence, we define a new graph G' by duplicating G for $n^{0.6}$ times (where there are no edges between the copies). This graph G' has $\mathrm{poly}(n)$ vertices. Moreover, we know that with probability at least $1 - (1 - 1/\sqrt{n})^{n^{0.6}} > 0.5$, one-round Luby's algorithm on G' returns an $\Omega(\log n)$-ruling set. This finishes the proof.