On Parallel k-Center Clustering

Sam Coy[*]
University of Warwick
Coventry, UK
S.Coy@warwick.ac.uk

Artur Czumaj[†]
University of Warwick
Coventry, UK
A.Czumaj@warwick.ac.uk

Gopinath Mishra[‡]
University of Warwick
Coventry, UK
Gopinath.Mishra@warwick.ac.uk

ABSTRACT

We consider the classic k-center problem in a parallel setting, on the low-local-space Massively Parallel Computation (MPC) model, with local space per machine of $O(n^{\delta})$, where $\delta \in (0,1)$ is an arbitrary constant. As a central clustering problem, the k-center problem has been studied extensively. Still, until very recently, all parallel MPC algorithms have been requiring $\Omega(k)$ or even $\Omega(kn^{\delta})$ local space per machine. While this setting covers the case of small values of k, for a large number of clusters these algorithms require large local memory, making them poorly scalable. The case of large k, $k \geq \Omega(n^{\delta})$, has been considered recently for the low-local-space MPC model by Bateni et al. (2021), who gave an $O(\log\log n)$-round MPC algorithm that produces $k(1+o(1))$ centers whose cost has multiplicative approximation of $O(\log\log\log n)$. In this paper we extend the algorithm of Bateni et al. and design a low-local-space MPC algorithm that in $O(\log\log n)$ rounds returns a clustering with $k(1+o(1))$ clusters that is an $O(\log^* n)$-approximation for k-center.

CCS CONCEPTS

• **Theory of computation → Massively parallel algorithms**; **Facility location and clustering**.

KEYWORDS

MPC, k-center, clustering, parallel computing, MapReduce

ACM Reference Format:
Sam Coy, Artur Czumaj, and Gopinath Mishra. 2023. On Parallel k-Center Clustering. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '23), June 17–19, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3558481.3591075

[*]Research supported in part by the Centre for Discrete Mathematics and its Applications (DIMAP), by an EPSRC studentship, and by the Simons Foundation Award No 663281 granted to the Institute of Mathematics of the Polish Academy of Sciences for the years 2021-2023.
[†]Research supported in part by the Centre for Discrete Mathematics and its Applications (DIMAP), by EPSRC award EP/V01305X/1, by a Weizmann-UK Making Connections Grant, by an IBM Award, and by the Simons Foundation Award No. 663281 granted to the Institute of Mathematics of the Polish Academy of Sciences for the years 2021–2023.
[‡]Research supported in part by the Centre for Discrete Mathematics and its Applications (DIMAP), by EPSRC award EP/V01305X/1, and by the Simons Foundation Award No 663281 granted to the Institute of Mathematics of the Polish Academy of Sciences for the years 2021-2023.

SPAA '23, June 17–19, 2023, Orlando, FL, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9545-8/23/06.
https://doi.org/10.1145/3558481.3591075

1 INTRODUCTION

Clustering large data is a fundamental primitive extensively studied because of its numerous applications in a variety of areas of computer science and data science. It is a central type of problem in modern data analysis, including the fields of data mining, pattern recognition, machine learning, networking and social networks, and bioinformatics. In a typical clustering problem, the goal is to partition the input data into subsets (called clusters) such that the points assigned to the same cluster are "similar" to one another, and data points assigned to different clusters are "dissimilar".

The most extensively studied clustering problems are k-means, k-median, k-center, various notions of hierarchical clustering, and also variants of these problems with some additional constraints (e.g., fairness or balance).

While originally the clustering problems have been studied in the context of classical sequential computation, most recently a large amount of research has been devoted to the non-sequential computational settings such as distributed and parallel computing, mainly because these are the only settings capable of performing computations in a reasonable time on large inputs, and because data is frequently collected on different sites and clustering needs to be performed in a distributed manner with low communication.

In this paper we consider one of the most fundamental clustering problems, the *k-center* problem, on the *Massively Parallel Computation (MPC)* model. MPC is a modern theoretical model of parallel computation, inspired by frameworks such as MapReduce [9], Hadoop [21], Dryad [17], and Spark [23]. Introduced just over a decade ago by Karloff et al. [18] (and later refined, e.g., in [1, 3, 8, 13]), the model has been the subject of an increasing quantity of fundamental research in recent years, becoming nowadays the standard theoretical parallel model of algorithmic study.

MPC is a parallel system with m *machines*, each with s words of *local memory*. (We also consider the *global space* g, which is the total space used across all machines, g = s · m.) Computation takes place in synchronous rounds: in each round, each machine may perform arbitrary computation on its local memory, and then exchange messages with other machines. Each message is sent to a single machine specified by the machine sending the message. Machines must send and receive at most s words each round. The messages are processed by recipients in the next round. At the end of the computation, machines collectively output the solution. The goal is to design an MPC algorithm that solves a given task in as few rounds as possible.

If the input is of size n, then one wants s to be sublinear in n (for if s $\geq n$ then a single machine can solve any problem without any communication, in a single round), and the total space across all the machines to be at least n (in order for the input to fit onto the machines) and ideally not much larger. In this paper, we focus on the *low-local-space* MPC setting, where the local space of each machine is strongly sublinear in the input size, i.e., $s = O(n^\delta)$ for some arbitrarily constant $\delta \in (0, 1)$. This low-local-space regime is especially attractive because of its scalability. At the same time, this setting is particularly challenging in that it requires extensive inter-machine communication to solve clustering problems for the input data scattered over many machines.

In recent years we have seen a number of very efficient, often constant-time, parallel clustering algorithms that have been relying on a combination of a *core-set* and a "reduce-and-merge" approach. In this setting, one gradually filters the data set by typically reducing its size on every machine to $\widetilde{O}(k)$, continuing until all the data can be stored on a single machine, at which point the problem is solved locally. Observe that this approach has an inherent bottleneck that requires that any machine must be able to store $\Omega(k)$ data points. Intuitively, this follows from the fact that if a machine sees k data points that are all very far away from each other, it needs to keep track of all k of them, for otherwise it might miss all the information about one of the clusters, which in turn could lead to a large miscalculation of the objective value. Similar arguments could be also used to argue that each machine needs to communicate $\Omega(k)$ points to the others (see [6] for a formalization of such intuition for a *worst-case partition* of points for k-median, k-means, and k-center problems, though the worst-case partition assumption means that this bound does not extend directly to MPC). Because of that, most of the earlier clustering MPC algorithms, especially those working in a constant number of rounds (see, e.g., [11, 19]), require $\Omega(k)$ or even $\Omega(k) \cdot n^{\Omega(1)}$ local space. Therefore in the setting considered in this paper, of MPC with local space per machine of $s = O(n^\delta)$, the approach described above cannot be applied when the number of clusters is large, when $k = \omega(s)$. This naturally leads to the main challenge in the design of clustering algorithms for MPC with low-local-space: *how to efficiently partition the data into k good quality clusters on an MPC with local space* $s \ll k$. We believe that this setting is quantitatively different (and more challenging) from the setting when k is smaller (or even comparable to) the amount of local space s.

In this paper, we focus on the *k-center clustering problem*, a standard, widely studied, and widely used formulation of metric clustering. The problem is, given a set of n input points, to find a subset of size k of these points (called *centers*) such that that maximum distance of a point to its nearest center is minimized. Specifically, in this work, we focus on the case where $k \gg s$ and hence, when k is quite large relative to n: one can think of these problem instances as "compressing" the input set of n points into k points. Very recently this problem has been addressed by Bateni et al. [2], who showed one can design an $O(\log\log n)$-round MPC algorithm, with local space $s = O(n^\delta)$ and global space $g = \widetilde{O}(n^{1+\delta})$,[1] that returns an $O(\log\log\log n)$-approximate solution with $k + o(k)$ centers. Our main result is an improved bound in the MPC model:

Theorem 1.1 (Main result stated informally). *In $O(\log\log n)$ rounds on an MPC, we can compute an $O(\log^* n)$ approximate solution to the k-centers problem using $k(1 + o(1))$ centers.*

The MPC has local space $s = O(n^\delta)$ and global space $g = \widetilde{O}(n^{1+\rho})$ for any constant $\rho > 0$. The n input points are in $\mathbb{R}^d$ for some constant d and we assume that $k = \Omega(\log^c n)$ for a suitable constant c. Our algorithm succeeds with high probability.

The algorithmic framework is based on a repeated application of *locally sensitive sampling*: sampling a set of "hub" points, assigning all other points to a nearby hub, and then adding new hubs to well-approximate the point set. We improve the approximation factor by a careful examination of the progress of clusters in some fixed optimal clustering over the course of the algorithm. Due to the depth of our iteration, clusters no longer satisfy certain properties with high probability, and carefully bounding the size of the clusters that fail to meet certain checks is an important challenge to overcome in our analysis. Additionally, we provide a more flexible guarantee on the global space, providing an accuracy parameter which can be set to reduce global space used at the expense of a larger approximation ratio (or vice versa).[2] This is possible because of the way we implement *locally-sensitive hashing (LSH)* in MPC. We believe our implementation of LSH in MPC could potentially see further applications, e.g., for other geometric problems.

1.1 Related work

There has been a large amount of work on various variants of the clustering problems (see, e.g., [22] for a survey of research until 2005), including some extensive study of the k-center clustering problem. The k-center problem is well known to be NP-hard and simple algorithms are known to achieve a 2-approximation [10, 12, 15]; this approximation ratio is tight unless P = NP [16].

The study of clustering in the context of parallel computing is extremely well-motivated: as the size of typical data sets continue to increase, it becomes infeasible to store input data on a single machine, let alone iterate over it many times as greedy sequential algorithms require (see, e.g., [12]). It comes therefore as no surprise that there has been a considerable amount of work on k-center clustering algorithms in MPC. In particular, several constant-round, constant-approximation algorithms in the MPC setting were given recently for general metric k-center clustering, see, e.g., [5, 11, 19]. Much of this work used *coresets* or similar techniques to approximate the structure of the underlying data, naturally implying a requirement that the local space satisfies $s = \Omega(k)$ per machine and global space is $g = \Omega(nk)$ or $\Omega(n^\epsilon k^2)$ [5, 11, 19]. Specifically, Ene et al. [11] gave a $O(1)$ round 10-approximation MPC algorithm that uses local space $s = O(k^2 n^{\Theta(1)})$, Malkomes et al. [19] obtained a 2-round 4-approximation MPC algorithm with local space $s = \Omega(\sqrt{nk})$, and Ceccarello et al. [5] obtained a 2-round $(2+\varepsilon)$-approximation MPC algorithm that uses local space $s = O_{d,\varepsilon}(\sqrt{nk})$ for the problem in metric spaces with doubling dimension d. As mentioned earlier, these algorithms are not scalable if k is large relative to n (for example, when $k = n^{1/3}$), making the case of large k challenging. Furthermore, as argued by Bateni et al. [2], the case of large k appears naturally in some applications of k-center clustering, including label propagation used in semi-supervised learning, or same-meaning query clustering for online advertisement or document search [20]. Unfortunately, we do not know of any

[1] $\widetilde{O}(f)$ hides a polynomial factor in $\log f$.

[2] In particular, the constant in $O(\log^* n)$ of Theorem 1.1 depends on ρ.

$O(1)$-round, $O(1)$-approximation MPC algorithm that would use local space $s = o(k)$.

In order to address the case of large k, Bateni et al. [2] considered a relaxed version of k-center clustering for low dimensional Euclidean spaces with constant dimension. The goal of that work was to design a scalable MPC algorithm for the k-center clustering problem with a *sublogarithmic number of rounds* of computation, *sublinear space per machine*, and small global space. Bateni et al. [2] showed that in $O(\log\log n)$ rounds on an MPC with $s = O(n^{\delta})$, one can compute an $O(\log\log\log n)$-approximate solution to constant-dimension Euclidean k-center with $k(1+o(1))$ centers. Their algorithm uses $\widetilde{O}(n^{1+\delta} \cdot \log \Delta)$ global space. Bateni et al. [2] complemented their analysis by some empirical study to demonstrate that the designed algorithm performs well in practice.

Finally, in the related PRAM model of parallel computation Blelloch and Tangwonsan gave a 2-approximation algorithm for k-center [4]. However, their algorithm requires $\Omega(n^2)$ processors and it is therefore difficult to translate the approach to our setting.

1.2 Technical contributions

Our main result in Theorem 1.1 is an extension of the approach developed in Bateni et al. [2] that significantly improves the quality of the approximation guarantee. To present these two results in the right context, we will briefly describe the main differences between these two works at a high level.

The approach of Bateni et al. [2] starts with the entire point set P as a set of potential centers (solution), and refines it to $P = P_0 \supseteq \cdots \supseteq P_\tau$, until $|P_\tau| = k + o(k)$. The final set P_τ is reported as the output. It is not difficult to see that if we take an optimal clustering C^* for P (i.e., C^* is the optimal solution to the k-center problem for P), then the number of potential centers in any cluster $C \in C^*$ reduces over rounds (that is, $|P_{i+1} \cap C| \le |P_i \cap C|$). Let us define a cluster $C \in C^*$ to be *irreducible from round i*, if i is the minimum index such that $|C \cap P_i| \le 1$. Two central properties of the cluster refinement due to Bateni et al. [2] are that after $O(\log\log n)$ rounds the size of each cluster in C^* reduces to $\widetilde{O}(\log n)$, and that after that, the total number of the points in the reducible clusters in C^* reduces after each round by a constant factor, implying that another $O(\log\log n)$ rounds suffice to ensure the desired number of centers (at most k due to the irreducible clusters and $o(k)$ due to the reducible clusters) and hence $\tau = O(\log\log n)$. This is then complemented by the analysis of the quality of the refinements which guarantees that each new refinement adds an additive term of $O(\text{OPT})$ to the cost of the solution, giving in total a double logarithmic approximation ratio. They further gave a sketch of the analysis to get an approximation ratio of $O(\log\log\log n)$.

In our paper we substantially improve the approximation factor to $O(\log^* n)$ by extending the framework in the following sense. We show that, after $O(\log\log n)$ rounds, the size of each cluster in C^* reduces to $\widetilde{O}(\log n)$ such that the refinement in each round adds an additive error of $O(\frac{\text{OPT}}{\log\log n})$ to the cost of the solution. Then, we show that after additional $O(\log\log\log n)$ rounds, the sizes of *almost all* (but not all) clusters in C^* reduce to a $\widetilde{O}(\log\log n)$ such that the refinement in each round adds an additive error of $O(\frac{\text{OPT}}{\log\log\log n})$ to the cost of the solution. Next, we show that after another $O(\log\log\log\log n)$ rounds, the sizes of *almost all* clusters in C^* reduce to a $\widetilde{O}(\log\log\log n)$ such that the refinement in each round adds $O(\frac{\text{OPT}}{\log\log\log\log n})$ to the cost of the solution, and so on. We continue this until the sizes of almost all clusters in C^* reduce to $O(\log^* n)$. Observe that the total number of rounds taken so far is bounded by $O(\log\log n)$, and we can argue that the current solution has an approximation ratio of $O(\log^* n)$. An important challenge in analyzing this approach is that *not all clusters satisfy these size guarantees with high probability.* Indeed, we cannot obtain a high probability guarantee by cluster refinement relying on random sampling of the already small clusters; we can ensure only that *most of the clusters are getting small.* Let $C^{**} \subseteq C^*$ be the clusters that satisfy the reduction property as discussed above, that is, such that the number of points in each cluster of C^{**} is bounded by $O(\log^* n)$ currently. We argue that the total number of points in the reducible clusters in C^{**} reduces by a constant factor after each successive round, adding an additive error of $O(\text{OPT})$ each time. This implies that another $O(\log(\log^* n))$ rounds are good enough to ensure that we have the desired number of centers at the end. To bound the total number of centers, we also need to show that the number of centers in clusters in $C^* \setminus C^{**}$ (that is, the set of clusters which fail to adhere to a size guarantee at some point during the algorithm) is bounded. Note that we cannot *track* which clusters succeed or fail (doing so would require us to know an optimal clustering), and so we use C^* and C^{**} only for the analysis. In summary, the approach sketched above will reduce the number of clusters to $k(1+o(1))$, and will ensure that the total number of rounds spent by our algorithm is $O(\log\log n)$ and the approximation ratio of our solution is $O(\log^* n)$. A more detailed overview is in Section 2.

Our approach relies heavily on the use of LSH (locality sensitive hashing), and we provide a flexible implementation of LSH in MPC which one can configure with an appropriate parameter ρ. Reducing the value of ρ decreases the amount of global space used by the algorithm (global space used is $\widetilde{O}(n^{1+\rho})$) while increasing the approximation ratio.

1.3 Notation and preliminaries

We now introduce the notation used through the paper.

First, we present the setting of the parameters of our MPC. The k-center algorithm in this paper works for any local space $s = O(n^{\delta})$ for a constant $0 < \delta < 1$: the setting of δ has only a constant factor impact on the running time. Similarly, the MPC can have any global space $g = \widetilde{O}(n^{1+\rho})$ for some constant $\rho > 0$: ρ can be made arbitrarily small, and its setting has a constant factor impact on the approximation ratio. We sometimes refer to MPC with these choices of s and g simply as "MPC" in the rest of this paper.

Let us recall that certain operations, particularly sorting and prefix sum of n elements, and broadcasting a value of size $<$ s, can be computed deterministically in $O(1)$ rounds (see [13]).

The input to our problem is a set P of n points in $\mathbb{R}^d$, where d is a constant, and an integer parameter $k < n$. We define $d(p,q)$ as the Euclidean distance between points p and q in $\mathbb{R}^d$. We generalize this notation to the distance between a point and a set: $d(p,S) := \min_{q \in S} d(p,q)$ is the minimum distance from p to a point in S. We define $\textsc{Cost}(P,S) := \max_{p \in P} d(p,S)$ as the distance of the point in

P which is "furthest away" from any point in S. Without loss of generality, we assume that the input set is re-scaled so that the minimum distance between any two points in P is 1; then we let Δ to be the maximum distance between any two points in P.

We denote the set $\{1, \ldots, t\}$ by $[t]$ and $\log^{(i)} n := \underbrace{\log \ldots \log}_{i} n$ the iterated logarithm of n. By convention $\log^{(0)} n := n$. The notations $\widetilde{O}(f)$ and $\widetilde{\Theta}(f)$ hide polynomial factors in $\log f$.

We now define formally the k-center clustering problem.

Definition 1.2. Let P be a set of points in $\mathbb{R}^d$. A *clustering* C of P is a partition of P into nonempty clusters $C_1, \ldots, C_t$. The *radius* of cluster C_i is $\min_{x \in C_i} \max_{y \in C_i} d(x, y)$, and the *cost* of the clustering C is the maximum of the radii of the clusters $C_1, \ldots, C_t$.

Definition 1.3 (k-center clustering problem). Let $k, n, d \in \mathbb{N}$ with $k \leq n$, and P be a set of n points in $\mathbb{R}^d$. The k-center problem for P is to find a set $S^* \subseteq P$ such that

$$S^* = \operatorname*{arg\,min}_{S \subseteq P : |S| = k} \textsc{Cost}(P, S).$$

Moreover, $\textsc{Cost}(P, S^*)$ is defined as the (optimal) cost of the k-center problem for P.

We assume throughout the paper that $k > \mathrm{s}$. However, our algorithms work as described provided that $k = \Omega((\log n)^c)$ for a suitable constant c (which is also the main focus of the work).

1.4 Our results — detailed bounds

We now present in details the main result of this paper:

Theorem 1.4 (Main result). *Let P be any set of n points in $\mathbb{R}^d$ and let* $\textit{opt}$ *denote the optimal cost of the k-center clustering problem for P. There exists an MPC algorithm that in $O(\log \log n)$ rounds determines with high probability a set $T \subseteq P$ of $k + o(k)$ centers, such that $\textsc{Cost}(P, T) = O(\log^* n) \cdot \textit{opt}$. The MPC uses local space $\mathrm{s} = O(n^\delta)$ and global space $\mathrm{g} = \widetilde{O}(n^{1+\rho} \cdot \log^2 \Delta)$.*

Theorem 1.4 follows directly from a more general theorem.

Theorem 1.5 (Generalization of Theorem 1.4). *Let α be an arbitrary integer, $1 \leq \alpha \leq \log^* n - c_0$ for some suitable constant c_0. Let P be any set of n points in $\mathbb{R}^d$ and let* $\textit{opt}$ *denote the optimal cost of the k-center clustering problem for P. There exists an MPC algorithm that in $O(\log \log n)$ rounds determines with high probability a set $T \subseteq P$ of centers, such that $\textsc{Cost}(P, T) = O((\alpha + \log^{(\alpha+1)} n)) \cdot \textit{opt}$ and $|T| \leq k \cdot \left(1 + \frac{1}{\widetilde{\Theta}(\log^{(\alpha)} n)}\right) + \widetilde{\Theta}((\log^{(\alpha)} n)^3)$. The MPC uses local space $\mathrm{s} = O(n^\delta)$ and global space $\mathrm{g} = \widetilde{O}(n^{1+\rho} \cdot \log^2 \Delta)$.*

Observe that in Theorem 1.5 we have $|T| = k + o(k)$, since we are assuming $k = \Omega((\log n)^c)$ for some suitable constant c.

Let α_0 be the solution to the equation $\alpha = \log^{(\alpha+1)} n$; observe that $\alpha_0 = \Theta(\log^* n)$. Then Theorem 1.4 is a corollary of Theorem 1.5 when we choose $\alpha = \alpha_0$.

Theorem 1.5 can be seen as a fine-grained version of Theorem 1.4: as α increases the cost of the solution decreases and number of center increases (with the number of rounds always being $O(\log \log n)$). Therefore Theorem 1.5 is more amiable in practical scenarios in the following sense: α in Theorem 1.5 can be set to trade off between the quality of the solution and the number of centers in the solution. We would also like to highlight that the result of Bateni et al. [2] is a special case of Theorem 1.5 when $\alpha = 1$ and $\alpha = 2$ to obtain $O(\log \log n)$ and $O(\log \log \log n)$ approximation, respectively.

1.5 Organization of the paper

In Section 2 we give a proof of our main result predicated on the correctness of our main algorithm, and then give an overview of the subroutines which our main algorithm contains. In Section 3 we explain how LSH (locality-sensitive hashing) on MPC can be implemented to assign each point $p \in P$ to a hub in $H \subseteq P$ which is within a constant factor of the closest hub to p. In Sections 4 and 5 we prove critical properties of subroutines used in our main algorithm, and then in Section 6 we prove the correctness of our main algorithm. Finally, Section 7 contains some conclusions. Because of space constraints, missing proofs are deferred to the full version [7].

2 TECHNICAL OVERVIEW

Recall that Theorem 1.4 is our main result and Theorem 1.5 is its parameterized generalization. Our proof of Theorem 1.5 (and hence of Theorem 1.4) relies on the following main technical theorem.

Theorem 2.1 (Main technical theorem proved in this paper). *Let α be an arbitrary integer, $1 \leq \alpha \leq \log^* n - c_0$ for some suitable constant c_0. Let r be an arbitrary positive real. Let P be any set of n points in $\mathbb{R}^d$ and let C_r be a clustering of P that has the minimum number of centers among all clusterings of P with cost at most r and $|C_r| = \Omega((\log n)^c)$ for a suitable constant c. There exists an MPC algorithm* Ext-k-Center *(Algorithm 5) that with probability at least $1 - \frac{1}{(\log^{(\alpha-1)} n)^{\Omega(1)}}$, in $O(\log \log n)$ rounds determines a set $T \subseteq P$ of centers, such that $\textsc{Cost}(P, T) = O(r \cdot (\alpha + \log^{(\alpha+1)} n))$ and $|T| \leq |C_r| \cdot \left(1 + \frac{1}{\widetilde{\Theta}(\log^{(\alpha)} n)}\right) + \widetilde{\Theta}((\log^{(\alpha)} n)^3)$. The MPC uses local space $\mathrm{s} = O(n^\delta)$ and global space $\mathrm{g} = \widetilde{O}(n^{1+\rho} \cdot \log \Delta)$.*

Algorithm Ext-k-Center used in Theorem 2.1 takes two parameters: an accuracy parameter α and a cost parameter r, and produces the output in a form similar to that required in Theorem 1.5, except that the number of clusters is equal to the number of centers in an optimal clustering of P with cost at most r. This is in contrast with a standard clustering setting where the number of clusters is given as input, with no relationship to the cost of the solution. Therefore, if we knew a constant factor approximation to the optimal cost to the k-center problem, then setting it to be r in Theorem 2.1, we would get a desired solution as required in Theorem 1.5. This naturally suggests to run Ext-k-Center multiple times in parallel in order to obtain Theorem 1.5. Note that the success probability of Theorem 2.1 is not high. Hence we first run Ext-k-Center a suitable number of times in parallel to get an algorithm Ext-k-Center$'$ whose output and space requirements are same as that of Ext-k-Center, but the success probability is high. Then we run Ext-k-Center$'$ for $O(\log \Delta)$ choices of r (starting with $r = \Delta$ and decreasing a constant factor each time) in parallel to get algorithm Ext-k-Center$''$ (the algorithm of Theorem 1.5). Moreover, Ext-k-Center$''$ reports the output of Ext-k-Center$'$

for the minimum r for which we get the number of centers equals to $k + o(k)$. The details are in the full version [7].

2.1 Overview of the proof of Theorem 2.1

The idea to prove Theorem 2.1 is based on the framework which we call *locally sensitive sampling*. We generate a set $H \subseteq P$ of points (called *hubs*) by sampling each point in P independently with a *suitable* probability, and assign all other points to one of the hubs based on its *locality*. Let B_h be the *bag of the hub h*—the set of points associated to a hub $h \in H$. We run a variation of a well known greedy algorithm [12] (for k-center in the sequential setting) for each bag in parallel to find a set of intermediate centers C_h for hub h such that $\text{Cost}(B_h, C_h) = O(r)$. We again repeat the procedure by setting $\bigcup_{h \in H} C_h$ as the point set. We continue this process a particular number of times with a particular choice of probability and radius parameters, and report the centers, at that point of time, as the final solution.

This framework was recently used by Bateni et al. [2] to give an $O(\log \log n)$-round MPC algorithm with local space $\mathrm{s} = O(n^{\delta})$ and global space $\mathrm{g} = \widetilde{O}(n^{1+\delta})$, which computes an $O(\log \log \log n)$-approximate solution to k-center with $k(1 + o(1))$ centers, with high probability. We extend their framework and generalize the analysis to give an $O(\log^* n)$ approximate solution as stated in Theorem 1.4. Note that Theorem 2.1 takes care of Theorem 1.4 via Theorem 1.5.

The algorithm corresponding to Theorem 2.1 is Ext-k-Center (Algorithm 5 in Section 6). Before describing Ext-k-Center, we describe and contextualize the three subroutines which it uses (Nearest-Hub-Search, Sample-And-Solve and Uniform-Center). The main algorithm of Bateni et al. [2] uses subroutine Sample-And-Solve and Uniform-k-center. We use analogous subroutines Sample-And-Solve and Uniform-Center in our algorithm corresponding to Sample-And-Solve and Uniform-k-center in Bateni et al. [2], respectively, to achieve the desired result. But there are some differences which we will discuss when we describe Sample-And-Solve and Uniform-Center. Due to our implementation of Nearest-Hub-Search, we are able to give a more flexible bound on global space. We can improve the approximation ratio mainly due to generalizing their Uniform-k-Center to Uniform-Center in our case and using sophisticated analysis in our main algorithm that calls Uniform-Center.

Let us discuss first at a high level what these subroutines achieve in the context of the framework of locally sensitive sampling (discussed at the beginning of this section). Intuitively, the purpose of Sample-And-Solve is to sparsify dense regions of points: it samples nodes with a given probability and iteratively adds centers in order to ensure that the cost of the centers remains low. Uniform-Center repeatedly uses Sample-And-Solve: its main purpose is to guarantee that the number of centers in each cluster of some fixed optimal clustering decreases in a certain way over time.

Nearest-Hub-Search *(Q, H).* Takes as input a set Q of at most n points and a set of hubs $H \subseteq Q$. For all points $q \in Q \setminus H$, it finds a point $\text{close}(q) \in H$ such that $d(q, \text{close}(q)) = O(d(q, H))$, with probability at least $1 - \frac{1}{n^{\Omega(1)}}$. Nearest-Hub-Search can be implemented in MPC with local space $\mathrm{s} = O(n^{\delta})$ and global space $\mathrm{g} = \widetilde{O}(n^{1+\rho} \cdot \log \Delta)$ in $O(1)$ rounds. Nearest-Hub-Search uses *locally sensitive hashing* [14] and its implementation in MPC. For details on Nearest-Hub-Search, see Section 3.

Sample-And-Solve *(Q, p, r).* Takes a set of Q of at most n points, a sampling parameter p, and a radius parameter r. It produces some set of centers $S \subseteq Q$ such that $\text{Cost}(Q, S) = O(r)$.[3] Importantly, this can be implemented in an MPC with local space $\mathrm{s} = O(n^{\delta})$ and global space $\widetilde{O}(n^{1+\rho} \cdot \log \Delta)$ in $O(1)$ rounds (Lemma 4.1) as, aside from using Nearest-Hub-Search to assign points to hubs, the computation is all done locally. Sample-And-Solve first samples each point in Q (independently) with probability p: let $H \subseteq Q$ be the set of sampled points called hubs. Then Sample-And-Solve calls Nearest-Hub-Search with input point set Q and hub set H. After getting $\text{close}(q)$ for each $q \in Q \setminus H$, Sample-And-Solve collects all points $B_h \subseteq Q$ assigned to a hub $h \in H$ (including hub h) and selects a set of centers C_h from B_h greedily using a variation of the sequential algorithm of [12], such that $\text{Cost}(B_h, C_h) = O(r)$. Finally, the algorithm outputs $S = \bigcup_{h \in H} C_h$. However, there is a difficulty to overcome: note that $|B_h|$ may be $\omega(n^{\delta})$. So B_h may not fit into the local memory of a machine. We show that this can be handled by distributing the points in B_h into multiple machines, duplicating h across all such machines. See Section 4 for more details about Sample-And-Solve.

Algorithm Sample-And-Solve in our paper serves essentially the same purpose as the corresponding algorithm due to Bateni et al. [2]. The approximation guarantee and number of rounds performed are the same in both cases. However, the global space used by our algorithm Sample-And-Solve is more flexible in the following sense: reducing the value of ρ decreases the amount of global space used by the algorithm (global space used is $\widetilde{O}(n^{1+\rho}) \cdot \log \Delta$) while increasing the approximation ratio.

Uniform-Center (V_t, r, t). Takes a set V_t of at most n points, a radius parameter r, and an additional parameter $t \leq n$. It produces a set S of centers, by calling Sample-And-Solve $\tau = \Theta(\log \log t)$ times. S_{i-1} is the input to the i-th call and S_i is the output of the i-th call: overall we have $S_0 = V_t$ and $S_\tau = S$ (the output of Uniform-Center). The probability and radius parameters to the calls to Sample-And-Solve are set *suitably*. From the guarantees we have from Sample-And-Solve, we have the following guarantee for Uniform-Center: (i) it can be implemented in an MPC with local space $\mathrm{s} = O(n^{\delta})$ and global space $\mathrm{g} = \widetilde{O}(n^{1+\rho} \cdot \log \Delta)$ in $O(\log \log t)$ rounds (Lemma 5.1), and (ii) $\text{Cost}(V_t, S) = O(r \cdot \tau) = O(r \log \log t)$ (Lemma 5.2). Uniform-Center guarantees a reduction in cluster sizes in an optimal clustering in the following sense. Consider a fixed clustering $\mathcal{C}_r^t$ of V_t that has cost at most r. For $C \in \mathcal{C}_r^t$: if $|C \cap V_t| \leq t$, then $|C \cap S| = O(\log t \cdot (\log \log t)^2)$, with probability at least $1 - \frac{1}{t^{\Omega(1)}}$. This is formally stated in Lemma 5.3: note that this ceases to be high probability when $t \in o(n)$. This guarantee on the size reduction plays a crucial role when proving the number of centers reported by Ext-k-Center in Section 6. For more details on Uniform-Center, see Section 5.

Our Uniform-Center is a full generalization of the analagous Uniform-k-center in Bateni et al. [2]. In particular, Uniform-k-Center is a special case of our Uniform-Center when $t = n$.

[3]The constant inside $O(\cdot)$ depends on ρ.

This generalization plays a crucial role in the correctness of Ext-k-Center when we call Uniform-Center multiple times. Uniform-k-Center is not robust enough to be called from Ext-k-Center multiple times to achieve the desired result.

Ext-k-Center. The algorithm consists of two phases, where **Phase 1** consists of α subphases and **Phase 2** consists of $\beta = \Theta(\log^{(\alpha+1)} n)$ subphases. In the j-th subphase of **Phase 1**, that is, in **Phase 1.j**, Ext-k-Center calls Uniform-Center$(T_{j-1}, r_{j-1}, t_{j-1})$, where $T_0 = P$, $r_0 = \frac{r}{\log\log n}$, $t_0 = n$, $t_j = \widetilde{\Theta}(\log^{(j)} n)$, and $r_j = \frac{r}{\log\log t_j}$. Observe that the guarantees of Uniform-Center ensure the following:

(i) **Phase 1** can be implemented in an MPC with local space s $= O(n^{\delta})$ and global space g $= \widetilde{O}(n^{1+\rho} \cdot \log \Delta)$ in $\sum_{j=1}^{\alpha} \log\log t_{j-1} = O(\log\log n)$ rounds;

(ii) $\text{Cost}(T_{j-1}, T_j) = O(r_{j-1} \log\log t_{j-1}) = O(r)$ for each $j \in [\alpha]$. Hence, $\text{Cost}(P, T_\alpha) = O(r\alpha)$.

Now consider **Phase 2** of Ext-k-Center.

In the i-th subphase of **Phase 2**, that is **Phase 2.i**, Ext-k-Center calls Sample-And-Solve$(T_{\alpha+i-1}, \frac{1}{2}, r)$, where $T = T_{\alpha+\beta}$ is the final output of Ext-k-Center. From the guarantee of Sample-And-Solve, we have

(i) **Phase 2** can be implemented in an MPC with local space s $= O(n^{\delta})$ and global space g $= \widetilde{O}(n^{1+\delta} \cdot \log \Delta)$ in $O(\beta) = O(\log^{(\alpha+1)} n)$ rounds;

(ii) $\text{Cost}(T_{\alpha+i-1}, T_{\alpha+1}) = O(r)$ for each $i \in [\beta]$. Hence,

$$\begin{aligned}\text{Cost}(P, T) = \text{Cost}(P, T_{\alpha+\beta}) &= \text{Cost}(P, T_\alpha) + O(\beta r)\\ &= O(r \cdot (\alpha + \log^{(\alpha+1)} n)) \ .\end{aligned}$$

Combining the guarantees concerning the round complexity, global space and approximation guarantee of **Phase 1** and **Phase 2**, we get the claimed guarantees on round complexity, global space and approximation guarantees in Theorem 2.1 (see Lemma 6.1 for round and global space guarantee and Lemma 6.2 for the guarantee on approximation factor).

Now, we discuss how we bound the number of centers that Ext-k-Center outputs, that is, $|T|$. Consider an optimal clustering $\mathcal{C}_r$ of P with cost at most r. A cluster $C \in \mathcal{C}_r$ is said to be *active* (after **Phase 1**) if $|C \cap T_j| \le t_j$ for each j with $1 \le j \le \alpha$. We say C is *inactive*, otherwise. Using the guarantee given by Uniform-Center concerning the reduction in cluster sizes, we can show that the total number of centers in T_α, that are in inactive clusters, is $O\left(\frac{|\mathcal{C}_r|}{(\log^{(\alpha)} n)^{\Omega(1)}}\right)$, with probability at least $1 - \sum_{i=1}^{\alpha} \frac{1}{t_{i-1}^{\Omega(1)}}$ (see Lemma 6.6). Note that T_α denotes the set of intermediate centers we have after **Phase 1**. So, for any cluster $C \in \mathcal{C}_r$ that is active after **Phase 1**, it satisfies $|C \cap T_\alpha| \le t_\alpha = \widetilde{\Theta}(\log^{(\alpha)} n)$. That is, with probability at least $1 - \sum_{i=1}^{\alpha} \frac{1}{t_{i-1}^{\Omega(1)}}$, we have the following:

$$|T_\alpha| \le |\mathcal{C}_r| \cdot t_\alpha + O\left(\frac{|\mathcal{C}_r|}{(\log^{(\alpha)} n)^{\Omega(1)}}\right) \ .$$

We define an active cluster $C \in \mathcal{C}_r$ is i-large if $|C \cap T_{\alpha+i-1}| \ge 2$. We show that the total number of intermediate centers in any large clusters reduces by a constant factor in **Phase 2.i**, with probability at least $1 - \frac{1}{t_{\alpha-1}^{\Omega(1)}}$. Note that the total number of intermediate centers in all active large clusters, just before **Phase 2**, is at most $|\mathcal{C}_r| \cdot t_\alpha = |\mathcal{C}_r| \cdot \widetilde{\Theta}(\log^{(\alpha)} n)$, and we are executing $\beta = \Theta(\log^{(\alpha+1)} n)$ many sub-phases in **Phase 2**. We can show that the total number of centers in the active large clusters, after **Phase 2**, is at most $\frac{|\mathcal{C}_r|}{\widetilde{\Theta}(\log^{(\alpha)} n)} + \widetilde{\Theta}((\log^{(\alpha)} n)^3)$, with probability at least $1 - \frac{1}{t_{\alpha-1}^{\Omega(1)}}$ (Lemma 6.7). Combined with the fact the number of active small clusters can be at most $|\mathcal{C}_r|$ with the bound on number of inactive clusters in **Phase 2**, we have the desired bound on $|T|$. Full details of Ext-k-Center and its analysis are presented in Section 6.

3 NEAREST HUB SEARCH

Recall that our Nearest-Hub-Search algorithm takes a set Q of points and a set $H \subseteq Q$ of hubs. For each $q \in (Q \setminus H)$, we want to find a hub $h \in H$ such that the distance between q and h is only a constant-factor more than the distance between q and the closest hub to q in H: informally, h is "almost" the closest hub to q in H.

In this section, we use *locally sensitive hashing* (LSH) [14] to implement algorithm Nearest-Hub-Search (Q, H) on MPC. Our implementation of locally sensitive hashing is parameterizable: by setting the parameter ρ appropriately, one can reduce the global space while increase the approximation ratio, or vice versa.

First, we begin by recalling the definition of locally sensitive hashing, introduced in [14]:

Definition 3.1 (Locally sensitive hashing [14]**).** Let $r \in \mathbb{R}^+$, $c > 1$ and $p_1, p_2 \in (0,1)$ be such that $p_1 > p_2$. A hash family $\mathcal{H} = \{h : \mathbb{R}^d \to U\}$ is said to be a (r, cr, p_1, p_2)-LSH family if for all $x, y \in \mathbb{R}^d$ the following hold:

- If $d(x,y) \le r$, then $\Pr_{h \in \mathcal{H}}(h(x) = h(y)) \ge p_1$;
- If $d(x,y) \ge cr$, then $\Pr_{h \in \mathcal{H}}(h(x) = h(y)) \le p_2$.

Consider the following proposition that talks about the existence of a particular hash family, which will be useful to describe and analyse Nearest-Hub-Search (Q, H) in Algorithm 1.

Proposition 3.2 ([14]). *Let $r, n \in \mathbb{N}$ and $\rho \in (0,1)$. There exists a $(r, c_\rho r, (1/n)^\rho, 1/n)$-LSH family, where c_ρ is a constant depending only on ρ.*

In Nearest-Hub-Search (Q, H), Q is a set of at most n points and $H \subseteq Q$ is the set of hubs. Our objective is to find a hub for each point which is at most some constant factor further away than the nearest hub, rather than finding the hub which is the closest. We do this by making $\log \Delta$ guesses about the distance to the nearest hub, and for each guess trying to find a hub within that distance.

For our $\log \Delta$ guesses for r (the distance to the closest hub), we take (independently and uniformly at random) $L = \Theta(n^\rho)$ many hash functions from a $(r, c_\rho r, (1/n)^\rho, 1/n)$-LSH family and use them to hash all the points, including the hubs.[4] Then we gather all points with the same hash value on consecutive machines. We then need to find, for each point, a hub that is close to it. This is difficult if the number of hubs mapped to a given hash value is large: if h

[4]The constant ρ is the same constant as in the exponent of n in the global space bound. Recall our tradeoff: choosing a smaller ρ decreases the amount of global space needed, but increases the approximation ratio.

hubs and m points are mapped to the same hash value, then we have to perform $h \cdot m$ distance checks, which is potentially prohibitive if $h \cdot m > \mathrm{s}$. To overcome this we show that, if many hubs are mapped to the same hash-value, we are able to discard all but a constant number of them, and retain for each point a hub that is within a constant factor of the distance of the closest hub. This works because of the choice of our hash function and by the definition of LSH. The full algorithm Nearest-Hub-Search (Q, H) is described in Algorithm 1, and its correctness is proved in Lemma 3.3.

Lemma 3.3 (Nearest hub search). *Let Q be a set of at most n points in $\mathbb{R}^d$, $H \subseteq Q$ denote the set of hubs, and c_ρ be a suitable constant depending on ρ. There exists an MPC algorithm* Nearest-Hub-Search *(Q, H) (as described in Algorithm 1) that with high probability, in $O(1)$ rounds, for all $q \in Q \setminus H$, finds a hub close$(q) \in H$ such that $d(q, close(q)) < 2c_\rho \cdot d(q, H)$. The MPC uses local space $\mathrm{s} = O(n^\delta)$ and global space $\mathrm{g} = O(n^{1+\rho} \cdot \log^2 n \cdot \log \Delta)$.*

Algorithm 1: Nearest-Hub-Search (Q, H)

```
Input: A set $Q$ of at most $n$ points and a set of hubs $H \subseteq Q$.
Output: For each point in $Q$, report close$(p) \in H$ such that
        $d(p, \text{close}(p)) \le 2c_\rho \cdot d(p, H)$, where $c_\rho$ is a
        suitable constant depending only on $\rho$.
1  begin
2      for ($i = 1$ to $I = \Theta(\log n)$) do
3          for ($j$=0 to $\log \Delta$) do
4              Set $r = 2^j$
5              Take $L = \Theta(n^\rho)$ many hash function $f_1, \dots, f_L$
                (independently and uniformly at random)
                from a $(r, c_\rho r, (1/n)^\rho, 1/n)$-LSH family.
6              for ($\ell = 1$ to $L$) do
7                  Determine $f_\ell(q)$ for each $q \in Q$.
8                  Find the distance of each $q \in Q$ with at most
                    a constant (say 10) number of hubs $h \in H$
                    such that $f_l(q) = f_l(h)$. If we get such a
                    $h \in H$ such that $d(q, h) \le c_\rho \cdot r$, then we
                    set close$_{ij\ell}(q) = h$ and NULL, otherwise.
9              end
10             Set close$_{ij}(q)$ = NULL if close$_{ij\ell}(q)$ = NULL for
                all $\ell \in [L]$. Otherwise, set
                close$_{ij}(q)$ = close$_{ij\ell}(q)$ for some $\ell \in L$.
11         end
12         Set close$_i(q)$ = NULL if close$_{ij}(q)$ = NULL for all
            $j \in [\log \Delta]$. Otherwise, close$_i(q)$ = close$_{ij^*}(q)$
            such that $j^*$ is minimum among all $j$ for which
            close$_{ij}(q)$ is not NULL.
13     end
14     If there exists a $q \in Q$ such that close$_i(q)$ is NULL for all
        $i \in [\log n]$, then report FAIL.
15     Otherwise, set close$(q)$ = close$_i(q)$ for some $i \in I$.
16 end
```

From Algorithm 1, note that we repeat a procedure (lines 3–12 that find an almost closest hub with probability $2/3$) $I = \Theta(\log n)$ times, and report the output we get from any of the instances. Consider Lemma 3.4, that says that, in Nearest-Hub-Search each point $q \in Q$ finds close$(q) \in H$ satisfying the required property with high probability. This will immediately imply the correctness of Lemma 3.3. We discuss the MPC implementation of Nearest-Hub-Search in the full version [7].

Note that close$_i(q)$ (which is either NULL or a point in H such that $d(q, \text{close}_i(q)) = O(d(q, H))$) denotes the output of Nearest-Hub-Search for point $q \in Q \setminus H$ and the instance $i \in I$.

Lemma 3.4. *For a particular $q \in Q \setminus H$ and $i \in I$, close$_i(q) \in H$ is not* NULL *and $d(q, close_i(q)) \le 2c_\rho \cdot d(q, H)$, with probability at least $2/3$.*

Now, consider the way Nearest-Hub-Search sets the value of close(q) in lines 14–15 from close$_i(q)$'s. By Lemma 3.4, we have close(q) such that it is not NULL and $d(q, \text{close}(q)) \le 2c_\rho \cdot d(q, H)$ with probability at least $1 - \frac{1}{n^{\Omega(1)}}$. This is because $I = \Theta(\log n)$. Applying the union bound over all points in $Q \setminus H$, we see that Lemma 3.3 is implied by Lemma 3.4, except the details of MPC implementation (which is discussed in the full version [7]).

4 SAMPLE AND SOLVE

In this section, we describe Sample-And-Solve (Q, p, r), which is a subroutine in Uniform-Center and Ext-k-Center in Sections 5 and 6, respectively. Sample-And-Solve (Q, p, r) takes a set Q of at most n points, a sampling parameter p, and a radius parameter r, it relies on Nearest-Hub-Search discussed in Section 3, and produces a set of centers $S \subseteq Q$ such that Cost$(Q, S) = O(r)$.

Algorithm 2: Greedy (R, h, r)

```
Input: Set $R$ of at most $n$ points; radius parameter $r \in \mathbb{R}^+$.
Output: A set $G \subseteq R$ of centers.
1  begin
2      Set $G \leftarrow \{h\}$.
3      while ($\exists\, x \in R$ with $d(x, G) = \min_{y \in R} d(x, y) > 4c_\rho r$)
        do
4          //            Here $c_\rho$ is the constant as in Lemma 3.3.
5          Let $x \in R$ be the point furthest from $G$; add $x$ to $G$.
6      end
7      Report the set $G$ of centers.
8  end
```

Sample-And-Solve (Q, p, r) calls algorithm Greedy (R, h, r) as a subroutine, which produces a set of centers $G \subseteq R$ such that Cost$(R, G) = O(r)$. Greedy (R, h, r) is a variation of a classic 2-approximation algorithm for k-center in the sequential setting [12]. In Sample-And-Solve (Q, p, r), the idea is to sample each point in Q (independently) with probability p to form a set of hubs H. Then each point $q \in Q$ will be assigned to some hub $h \in H$ by using Nearest-Hub-Search (as described in Algorithm 1). For $h \in H$, let B_h be the set of points assigned to h (including h itself). We run Greedy for the points in B_h, to produce a set of centers S_h. Finally, $\bigcup_{h \in H} S_h$ is the output reported by Sample-And-Solve. There are other technicalities – $|B_h|$ may be much larger than s. In that case, we distribute the points in $B_h \setminus \{h\}$ across a number of machines,

Algorithm 3: Sample-And-Solve (Q, p, r)

```
Input: Set $Q$ of at most $n$ points; probability parameter
       $p \in (0,1)$; radius parameter $r \in \mathbb{R}^+$.
Output: A set $S \subseteq Q$ of centers.
1  begin
2      if $\left(|Q| \leq \mathrm{s} = O(n^\delta)\right)$ then
3          Call Greedy$(Q, q, r)$ for some arbitrary $q \in Q$, and
           report the set of centers output by it as $S$.
4      end
5      Sample each point in $Q$ independently with probability
       $p$. Points which are sampled form the set of hubs $H$.
6      If $H = \emptyset$, report Fail.
7      For each point $q$ in $Q$, assign it to the closest hub in $H$
       by calling Nearest-Hub-Search$(Q, H, \rho)$. We call the
       set of points assigned to a hub $h \in H$ the bag
       corresponding to $h$, and denote it as $B_h$. Note that $B_h$
       includes $h$.
8      for (each $h \in H$) do
9          if $(|B_h| \leq \mathrm{s})$ then
10             Collect $B_h$ on a single machine.
11             $S_h \leftarrow$ Greedy$(B_h, h, r)$.
12         end
13         else
14             Form bags $B_{h_1}, \ldots, B_{h_w}$, keeping $h$ in every $B_{h_i}$
               ($i \in [w]$) and putting other point in $B_h \setminus \{h\}$
               into exactly one of the $B_{h_i}$'s, such that
               $|B_{h_i}| \leq \mathrm{s} = O(n^\delta)$ for each $i \in [w]$.
15             $S_{h_i} \leftarrow$ Greedy$(B_{h_i}, h, r)$, where $i \in [w]$.
16             $S_h \leftarrow \bigcup_{i=1}^{w} S_{h_i}$.
17         end
18     end
19     Report set of centers $S = \bigcup_{h \in H} S_h$.
20 end
```

but we send h to each machine, ensuring that the total number of points assigned to a machine (including h) is less than s—and then we apply Greedy to the points on each of these machines.

The formal algorithm for Sample-And-Solve is presented in Algorithm 3. The approximation guarantee, round complexity and space complexity of Sample-And-Solve are stated in Lemma 4.1. An additional property of Sample-And-Solve is stated in Lemma 4.3 which will be useful in both Section 5 and Section 6.

Lemma 4.1 (Approximation guarantee, round complexity and space complexity of Sample-And-Solve**).** *Consider* Sample-And-Solve *(Q, p, r), as described in Algorithm 3. With probability at least* $1 - \min\left\{e^{-\Omega(p \cdot n^\delta)}, \frac{1}{n^{\Omega(1)}}\right\}$*, it does not report* Fail*, and moreover:*

(i) It produces a set of centers $S \subseteq Q$ such that Cost$(Q, S) \leq 4c_\rho r = O(r)$*, where c_ρ is the constant as in Lemma 3.3;*

(ii) It takes $O(1)$ MPC rounds with local space $\mathrm{s} = O\left(n^\delta\right)$ and global space $\mathrm{g} = \widetilde{O}(n^{1+\rho} \cdot \log \Delta)$.

Remark 4.2. We call Sample-And-Solve from Uniform-Center (Algorithm 4) with probability parameter $p = \Omega\left(\frac{\log n}{n^\delta}\right)$. Therefore, the success probability of Sample-And-Solve in our case is always at least $1 - \frac{1}{n^{\Omega(1)}}$.

Lemma 4.3 (An additional guarantee of Sample-And-Solve**).** *Let C_r be a clustering of Q having cost at most r. Then, with high probability, the following holds for any $C \in C_r$: if at least one hub is selected from C, then no further point in $C \setminus H$ is selected as a center, that is, $|S \cap C| = |H \cap C|$.*

5 UNIFORM CENTER ALGORITHM

In this section, we describe Uniform-Center (V_t, r, t), which iteratively refines a set of centers to a smaller set of centers, by calling Sample-And-Solve on a quadratically-increasing probability schedule. It calls Sample-And-Solve $\Theta(\log \log t)$ times. The i-th call to Sample-And-Solve is Sample-And-Solve (S_{i-1}, p_{i-1}, r) (in particular): it produces a set $S_i \subseteq S_{i-1}$ of centers as the output, where $S_0 = V_t$ and the probability parameters are set suitably. The algorithm is given in Algorithm 4; the round complexity, space complexity and approximation guarantee are given in Lemmas 5.1 and 5.2 — they follow from the guarantees we have for Sample-And-Solve in Lemma 4.1 and the fact that Uniform-Center (V_t, r, t) calls Sample-And-Solve $O(\log \log t)$ times. Uniform-Center has an additional guarantee (see Lemma 5.3) relating to the reduction of cluster sizes. This plays a crucial role in proving the correctness of Ext-k-Center in Section 6. In particular it helps to bound the number of centers output by Ext-k-Center.

Algorithm 4: Uniform-Center (V_t, r, t)

```
Input: A set of points $V_t$ of at most $n$ points, a radius
       parameter $r \in \mathbb{R}^+$, and an additional parameter
       $t \leq n$.
Output: A set $S \subseteq V_t$ of centers.
1  begin
2      $p_0 = \Theta\left(\frac{\log n}{n^\delta}\right)$, $s_0 = t$, and $S_0 \leftarrow V_t$.
3      for $i = 1$ to $\tau = \Theta(\log \log t)$ do
4          $S_i \leftarrow$ Sample-And-Solve $(S_{i-1}, p_{i-1}, r)$ .
5          $s_i \leftarrow \sqrt{s_{i-1}}$ and $p_i = \frac{1}{s_i}$.
6      end
7      Report $S = S_\tau$.
8  end
```

Lemma 5.1 (Round complexity and global space of Uniform-Center**).** *Consider* Uniform-Center *(V_t, r, t), as described in Algorithm 4. The number of rounds taken by the algorithm is $O(\log \log t)$ and the global space used by the algorithm is* $\mathrm{g} = \widetilde{O}\left(n^{1+\rho} \cdot \log \Delta\right)$.

Lemma 5.2 (Approximation guarantee of Uniform-Center**).** *Consider* Uniform-Center *(V_t, r, t) as described in Algorithm 4. It produces output S such that* Cost$(V_t, S) = O(r \cdot \log \log t)$.

Lemma 5.3 (Reduction in cluster sizes). *Consider* Uniform-Center *(V_t, r, t) as described in Algorithm 4, and a fixed clustering*

C_r^t of V_t that has cost r. It produces output $S \subseteq V_t$ such that the following holds for any $C \in C_r^t$: if $|C \cap V_t| \leq t$, then with probability at least $1 - \frac{1}{t^{\Omega(1)}}$, we have $|C \cap S| = O\left(\log t \cdot (\log\log t)^2\right)$.

Proof. Let $b_i = (1+\eta)^i s_i \log t \cdot (\log\log t)^2$, where i is an non-negative integer and $\eta = \Theta\left(\frac{1}{\log\log t}\right)$. Observe that

$$b_{\tau-1} = (1+\eta)^{\tau-1} s_{\tau-1} \log t (\log\log t)^2 = O\left(\log t \cdot (\log\log t)^2\right).$$

Using induction on i ($i \in \mathbb{N}$), we will show that $|C \cap S_i| \leq b_{i-1}$ for each i with $1 \leq i \leq \tau$, with probability at least $1 - \frac{1}{t^{\Omega(1)}}$. This will imply the desired result as S_i is the output after the i-th iteration, and $S = S_\tau$. Hence, $|C \cap S| \leq b_{\tau-1} = O\left(\log t \cdot (\log\log t)^2\right)$.

For $i = 1$,

$$|C \cap S_1| \leq |C \cap S_0| = |C \cap V_t| \leq t \leq b_0.$$

The first inequality follows as $S_1 \subseteq S_0$; the second equality follows as $S_0 = V_t$; the third inequality follows from the given condition that $|C \cap V_t| \leq t$; and the fourth one holds by the definition of b_0.

Suppose the statement holds for each i with $1 \leq i \leq \ell - 1$, that is, $|C \cap S_i| \leq b_{i-1}$ for each i with $1 \leq i \leq \ell - 1$. Now we argue for $i = \ell$. If $|C \cap S_{\ell-1}| \leq b_{\ell-1}$, then $|C \cap S_\ell| \leq b_{\ell-1}$, as $S_\ell \subseteq S_{\ell-1}$. So, let us assume that $|C \cap S_{\ell-1}| > b_{\ell-1}$.

Consider the ℓ-th iteration of Uniform-Center: it calls algorithm Sample-And-Solve$(S_{\ell-1}, p_{\ell-1}, r)$, and produces S_ℓ as the set of intermediate centers. Let $H_\ell \subseteq S_{\ell-1}$ be the set of hubs sampled in the call of Sample-And-Solve$(S_{\ell-1}, p_{\ell-1}, r)$, where each point in $S_{\ell-1}$ (independently) included in H_ℓ with probability $p_{\ell-1}$.

Before proceeding, we make two observations:

Observation 5.4. The probability, that at least one point from $C \cap S_{\ell-1}$ is in H_ℓ, is at least $1 - \frac{1}{t^{\Omega(1)}}$.

Proof. The probability, that no point in $C \cap S_{\ell-1}$ ($|C \cap S_{\ell-1}| > b_{\ell-1}$) is included in H_ℓ, is at most

$$(1 - p_{\ell-1})^{b_{\ell-1}} = \left(1 - \frac{1}{s_{\ell-1}}\right)^{b_{\ell-1}} \leq \frac{1}{t^{\Omega(1)}}.$$

Here, we have used that $b_{\ell-1} = (1+\eta)^{\ell-1} s_{\ell-1} \log t \cdot (\log\log t)^2$. □

Observation 5.5. With probability at least $1 - \frac{1}{t^{\Omega(1)}}$, the number of points in $C \cap S_{\ell-1}$ that are in H_ℓ is at most $b_{\ell-1}$.

Proof. By induction hypothesis, $|C \cap S_{\ell-1}| \leq b_{\ell-2}$. The expected number of points of $C \cap S_{\ell-1}$ that are in H_ℓ is

$$\begin{aligned} |C \cap S_{\ell-1}| \cdot \frac{1}{s_{\ell-1}} &\leq \frac{b_{\ell-2}}{s_{\ell-1}} \\ &= \frac{(1+\eta)^{\ell-2} s_{\ell-2} \log t \cdot (\log\log t)^2}{s_{\ell-1}} \\ &= (1+\eta)^{\ell-2} s_{\ell-1} \log t \cdot (\log\log t)^2 = \mu. \end{aligned}$$

By using Chernoff bound, the probability, that the number of points of $C \cap S_{\ell-1}$ that are in H_ℓ is more than $(1+\eta)\mu = b_{\ell-1}$, is at most $e^{-\frac{\eta^2 \mu}{3}} \leq \frac{1}{t^{\Omega(1)}}$. □

Observations 5.4 and 5.5, along with Lemma 4.3, give us that $|C \cap S_\ell| = |H_\ell| \leq b_{\ell-1}$ with probability at least $1 - \frac{1}{t^{\Omega(1)}}$. □

6 THE MAIN ALGORITHM

In this section, we present our main algorithm Ext-k-Center. Recall the overall description of Ext-k-Center in Section 2. Ext-k-Center has two phases. In **Phase 1**, it calls Uniform-Center α times, and in **Phase 2**, it calls Sample-And-Solve β times, where α is the input precision parameter and $\beta = \Theta(\log^{(\alpha+1)} n)$. The formal algorithm is described in Algorithm 5. We prove the round complexity and space complexity of Ext-k-Center in Lemma 6.1, the approximation guarantee in Lemma 6.2 and the bound on the number of centers in Lemma 6.3.

Algorithm 5: Ext-k-Center (P, r)

```
Input: Set $P$ of $n$ points; tradeoff parameter $\alpha$; radius
       parameter $r \in \mathbb{R}^+$.
Output: A set $T \subseteq P$ of centers.
1  begin
2      Phase 1:
3      $T_0 \leftarrow P$, $t_0 = n$, and $r_0 = \log\log n$.
4      for ($j = 1$ to $\alpha$) do
5          Phase 1.j:
6          $T_j \leftarrow$ Uniform-Center$(T_{j-1}, r_{j-1}, t_{j-1})$.
7          $t_j = \Theta(\log t_{j-1} \cdot (\log\log t_{j-1})^{d+2})$.
8                  // Note that $t_j = \widetilde{\Theta}(\log t_{j-1}) = \widetilde{\Theta}(\log^{(j)} n)$.
9          $r_j = \frac{r}{\log\log t_j}$.
10     end
11     Phase 2:
12     for ($i = 1$ to $\beta = \Theta(\log^{(\alpha+1)} n)$) do
13         Phase 2.i:
14         $T_{\alpha+i} \leftarrow$ Sample-And-Solve$(T_{\alpha+i-1}, \frac{1}{2}, r)$.
15     end
16     Report $T = T_{\alpha+\beta}$.
17 end
```

Lemma 6.1 (Round complexity and global space of Ext-k-Center**).** *Consider Ext-k-Center (P, t), as described in Algorithm 5. The number of rounds taken by the algorithm is $O(\log\log n)$ and the global space used by the algorithm is $\mathrm{g} = \widetilde{O}\left(n^{1+\rho} \cdot \log\Delta\right)$.*

Lemma 6.2 (Approximation guarantee of Ext-k-Center**).** *Let us consider Ext-k-Center (P, r) as described in Algorithm 5. It produces output $T \subseteq P$ such that $\mathrm{Cost}(P, T) = O(r \cdot (\alpha + \log^{(\alpha+1)} n))$.*

Proof. Observe that

$$\mathrm{Cost}(P, T) = \mathrm{Cost}(T_0, T_{\alpha+\beta}) \leq \mathrm{Cost}(T_0, T_\alpha) + \mathrm{Cost}(T_\alpha, T_{\alpha+\beta}) .$$

It therefore suffices to show that $\mathrm{Cost}(T_0, T_\alpha)$ and $\mathrm{Cost}(T_\alpha, T_{\alpha+\beta})$ are bounded by $O(r\alpha)$ and $O(r \cdot \log^{(\alpha+1)} n)$, respectively.

For any j with $1 \leq j \leq \alpha$, note that Ext-k-Center (P, r) calls Uniform-Center$(T_{j-1}, r_{j-1}, t_{j-1})$ in **Phase 1.j** and produces T_j as the output. So, by Lemma 5.2, $\mathrm{Cost}(T_{j-1}, T_j) = O(r_{j-1} \cdot \log\log t_{j-1})$, which is $O(r)$. Hence,

$$\mathrm{Cost}(T_0, T_\alpha) \leq \sum_{j=1}^{\alpha} \mathrm{Cost}\left(T_{j-1}, T_j\right) = \alpha \cdot O(r) = O(r \cdot \alpha) .$$

For any i with $1 \le i \le \beta$, note that Ext-k-Center (P, r) calls Sample-And-Solve$(T_{\alpha+i-1}, 1/2, r)$ in **Phase 2.i** and produces $T_{\alpha+i}$ as the output. So, by Lemma 4.1 (i), $\textsc{Cost}(T_{\alpha+i-1}, T_{\alpha+i}) = O(r)$. Hence, as $\beta = \Theta(\log^{(\alpha+1)} n)$,

$$\textsc{Cost}\left(T_\alpha, T_{\alpha+\beta}\right) \le \sum_{i=1}^{\beta} \textsc{Cost}\left(T_{\alpha+i-1}, T_{\alpha+i}\right) = O(r \cdot \log^{(\alpha+1)} n) \ . \ \square$$

Lemma 6.3 (Number of centers reported by Ext-k-Center). *Consider* Ext-k-Center (P, r) *as described in Algorithm 5. It produces output T such that, with probability at least* $1 - \frac{1}{(\log^{(\alpha-1)} n)^{\Omega(1)}}$,

$$|T| \le |C_r| \left(1 + \frac{1}{\widetilde{\Theta}(\log^{(\alpha)} n)}\right) + \widetilde{\Theta}((\log^{(\alpha)} n)^3) \ .$$

Here, C_r is a clustering of P that has the minimum number of centers among all possible clustering of P with cost at most r such that $|C_r| = \Omega((\log n)^c)$, where c is a suitable constant.

Now, we introduce the notion of *active* and *inactive* clusters in the following definition, which is useful in proving Lemma 6.3. Inactive clusters are clusters which, at some point during **Phase 1**, fail to reduce in size sufficiently. After the sub-phase during which they fail to reduce in size sufficiently, we assume that they never reduce in size again (since this is the worst case). We are then able to bound the total number of centers in inactive clusters (Lemma 6.6). Active clusters, by contrast, always reduce in size as we expect: the number of centers in active clusters is therefore easy to bound.

Definition 6.4. Let C_r be an optimal clustering with cost at most r. For each $C \in C_r$ and j with $1 \le j \le \alpha$, we say C is *inactive* in **Phase 1.j** if $|C \cap T_i| > t_i$ for some i with $1 \le i < j$. Otherwise, if $|C \cap T_i| \le t_i$ for every i with $1 \le i < j$, C is called *active* in **Phase 1.j**.

Let $C'_r \subseteq C_r$ be the set of clusters that are active after **Phase 1**, that is, **Phase 1.α**. By the definition of active clusters, for each $C \in C'_r$, $|C \cap T_\alpha| \le t_\alpha$. Note that Ext-$k$-Center goes over β sub-phases in **Phase 2**. After **Phase 1** and before the start of **Phase 2**, it has T_α as the set of intermediate centers. For $1 \le i \le \beta$, in **Phase 2.i**, we call Sample-And-Solve $\left(T_{\alpha+i-1}, \frac{1}{2}, r\right)$, and get $T_{\alpha+i}$ as the intermediate centers. For $0 \le i \le \beta$; a cluster $C \in C'_r$ is said to be *i-large* if $|C \cap T_{\alpha+i}| \ge 2$. Let $\Gamma_i \subseteq C'_r$ denote the set of *i-large* clusters, and let Y_i denote the total number of points that are in i-large clusters, that is, $Y_i = \sum_{C \in \Gamma_i} |C \cap T_{\alpha+i}|$.

Note that, in Lemma 6.3, we want to bound the number of centers in $T = T_{\alpha+\beta}$. We first observe that $|T|$ can be expressed as the sum of three quantities:

Observation 6.5. $|T| = \left|T_{\alpha+\beta}\right| = |C_r| + Y_\beta + \sum_{C \in C_r \setminus C'_r} |C \cap T_\alpha|$.

Proof. Observe that since $T_{\alpha+\beta} \subseteq T_\alpha$, we obtain,

$$\begin{aligned} T_{\alpha+\beta} &= \sum_{C \in C_r \setminus C'_r} \left|C \cap T_{\alpha+\beta}\right| + \sum_{C \in C'_r} \left|C \cap T_{\alpha+\beta}\right| \\ &\le \sum_{C \in C_r \setminus C'_r} |C \cap T_\alpha| + \sum_{C \in C'_r} \left|C \cap T_{\alpha+\beta}\right| \ . \end{aligned}$$

To bound the second inequality by $|C_r| + Y_\beta$, observe that

$$\begin{aligned} \sum_{C \in C'_r} \left|C \cap T_{\alpha+\beta}\right| &= \sum_{C \in C'_r : |C \cap T_{\alpha+\beta}|=1} \left|C \cap T_{\alpha+\beta}\right| + \sum_{C \in C'_r : |C \cap T_{\alpha+\beta}| \ge 2} \left|C \cap T_{\alpha+\beta}\right| \\ &\le \left|C'_r\right| + Y_\beta \le |C_r| + Y_\beta \ , \end{aligned}$$

which used that $C'_r \subseteq C_r$. This yields Observation 6.5. $\square$

In the following lemmas, we bound $\sum_{C \in C_r \setminus C'_r} |C \cap T_\alpha|$ and Y_β, and (with Observation 6.5) the result of Lemma 6.3 immediately follows from these bounds. Lemmas 6.6 and 6.7 are technical whose proofs are in the full version [7] due to paucity of space.

Lemma 6.6. *With probability at least* $1 - \sum_{i=1}^{\alpha} \frac{1}{t_{i-1}^{\Omega(1)}}$, $\sum_{C \in C_r \setminus C'_r} |C \cap T_\alpha|$ *is* $O\left(\frac{|C_r|}{(\log^{(\alpha)} n)^{\Omega(1)}}\right)$, *that is, the number of points in T_α that are present in clusters that are inactive after* **Phase 1** *is* $O\left(\frac{|C_r|}{(\log^{(\alpha)} n)^{\Omega(1)}}\right)$.

Lemma 6.7. *With probability at least* $1 - \frac{1}{t_{\alpha-1}^{\Omega(1)}}$, *we have* $Y_\beta = O\left(\frac{|C_r|}{t_\alpha} + t_\alpha^3 \cdot \log t_\alpha\right)$.

Proof of Lemma 6.3 using Lemma 6.6 and Lemma 6.7. From the above two lemmas along with Observation 6.5 and the fact $t_j = \widetilde{\Theta}(\log^{(j)} n)$, we have the following bound on $|T|$ with probability at least $1 - \sum_{i=1}^{\alpha} \frac{1}{t_{i-1}^{\Omega(1)}} \ge 1 - \frac{1}{(\log^{(\alpha-1)} n)^{\Omega(1)}}$:

$$|T| \le |C_r| \left(1 + \frac{1}{\widetilde{\Theta}(\log^{(\alpha)} n)}\right) + \widetilde{\Theta}\left((\log^{(\alpha)} n)^3\right) \ . \ \square$$

7 CONCLUSIONS

In this paper we show that even for large values of k, the classic k-center clustering problem in low-dimensional Euclidean space can be efficiently and very well approximated in the parallel setting of low-local-space MPC. While some earlier works (see, e.g., [5, 11, 19]) were able to obtain constant-round MPC algorithms, they were relying on a large local space $s \gg k$ allowing to successfully apply the core-set approach, which permits only limited communication. On the other hand, the low-local-space setting considered in this paper seems to require extensive communication between the machines to achieve any reasonable approximation guarantees. Therefore we believe (without any evidence) that the number of rounds of order $O(\log \log n)$ may be almost as good as it gets. Also, we concede that our algorithm does not achieve a constant approximation guarantee, but we feel the approximation bound of $O(\log^* n)$ is almost as good. Finally, our algorithm does not resolve the perfect setting of the k-center clustering in that it allows in the solution slightly more centers, $k + o(k)$ centers. The improvement of these three parameters is interesting open work.

We believe that solely using the technique in this paper, improving the approximation factor and/or number of rounds may not be possible (a detailed explanation is in the full version [7]), but the approach may be useful for related problems in MPC or other models. An interesting open problem is the extension of our work to high-dimensional Euclidean space (or a general metric space). We are not aware of any *efficient* LSH implementations for high dimensional space, and this appears to be the main challenge.

REFERENCES

[1] Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. 2014. Parallel Algorithms for Geometric Graph Problems. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*. 574–583. https://doi.org/10.1145/2591796.2591805

[2] MohammadHossein Bateni, Hossein Esfandiari, Manuela Fischer, and Vahab Mirrokni. 2021. Extreme k-Center Clustering. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence(AAAI)*. 3941–3949. https://ojs.aaai.org/index.php/AAAI/article/view/16513

[3] Paul Beame, Paraschos Koutris, and Dan Suciu. 2017. Communication Steps for Parallel Query Processing. *J. ACM* 64, 6 (2017), 40:1–40:58. https://doi.org/10.1145/3125644

[4] Guy E Blelloch and Kanat Tangwongsan. 2010. Parallel approximation algorithms for facility-location problems. In *Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures*. 315–324.

[5] Matteo Ceccarello, Andrea Pietracaprina, and Geppino Pucci. 2019. Solving k-center Clustering (with Outliers) in MapReduce and Streaming, almost as Accurately as Sequentially. *Proceedings of the VLDB Endowment* 12, 7 (2019), 766–778. https://doi.org/10.14778/3317315.3317319

[6] Jiecao Chen, He Sun, David P. Woodruff, and Qin Zhang. 2016. Communication-Optimal Distributed Clustering. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS)*. 3720–3728. https://proceedings.neurips.cc/paper/2016/hash/7503cfacd12053d309b6bed5c89de212-Abstract.html

[7] Sam Coy, Artur Czumaj, and Gopinath Mishra. 2023. On Parallel k-Center Clustering. *Preprint arXiv:2304.05883* (2023).

[8] Artur Czumaj, Jakub Łącki, Aleksander Mądry, Slobodan Mitrović, Krzysztof Onak, and Piotr Sankowski. 2018. Round Compression for Parallel Matching Algorithms. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*. 471–484.

[9] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM* 51, 1 (January 2008), 107–113. https://doi.org/10.1145/1327452.1327492

[10] Martin E. Dyer and Alan M. Frieze. 1985. A Simple Heuristic for the p-Centre Problem. *Operations Research Letters* 3, 6 (1985), 285–288.

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

[12] Teofilo F. Gonzalez. 1985. Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science* 38 (1985), 293–306. Issue 2-3. https://doi.org/10.1016/0304-3975(85)90224-5

[13] Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. 2011. Sorting, Searching, and Simulation in the MapReduce Framework. In *Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC)*. 374–383.

[14] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. 2012. Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality. *Theoretical Computer Science* 8, 1 (2012), 321–350. https://doi.org/10.4086/toc.2012.v008a014

[15] Dorit S. Hochbaum and David B. Shmoys. 1985. A Best Possible Heuristic for the k-Center Problem. *Mathematics of Operations Research* 10, 2 (1985), 180–184. https://doi.org/10.1287/moor.10.2.180

[16] Wen-Lian Hsu and George L. Nemhauser. 1979. Easy and Hard Bottleneck Location Problems. *Discrete Applied Mathematics* 1, 3 (1979), 209–215. https://doi.org/10.1016/0166-218X(79)90044-1

[17] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. 2007. Dryad: Distributed Data-parallel Programs from Sequential Building Blocks. *SIGOPS Operating Systems Review* 41, 3 (March 2007), 59–72.

[18] Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. 2010. A Model of Computation for MapReduce. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 938–948. https://doi.org/10.1137/1.9781611973075.76

[19] Gustavo Malkomes, Matt J. Kusner, Wenlin Chen, Kilian Q. Weinberger, and Benjamin Moseley. 2015. Fast Distributed k-Center Clustering with Outliers on Massive Data. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS)*. 1063–1071. https://proceedings.neurips.cc/paper/2015/hash/8fecb20817b3847419bb3de39a609afe-Abstract.html

[20] Haofen Wang, Yan Liang, Linyun Fu, Gui-Rong Xue, and Yong Yu. 2009. Efficient Query Expansion for Advertisement Search. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 51–58. https://doi.org/10.1145/1571941.1571953

[21] Tom White. 2015. *Hadoop: The Definitive Guide: Storage and Analysis at Internet Scale* (4th ed.). O'Reilly Media, Sebastopol, CA.

[22] Rui Xu and Donald Wunsch. 2005. Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks* 16, 3 (2005), 645–678. https://doi.org/10.1109/TNN.2005.845141

[23] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: Cluster Computing with Working Sets. In *Proceedings of the 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*.