Almost Optimal Massively Parallel Algorithms for k-Center Clustering and Diversity Maximization

Alireza Haqi
Sharif University of Technology
Tehran, Iran
alireza.haqi@sharif.edu

Hamid Zarrabi-Zadeh
Sharif University of Technology
Tehran, Iran
zarrabi@sharif.edu

ABSTRACT

Clustering and diversification are two central problems with various applications in machine learning, data mining, and information retrieval. The k-center clustering and k-diversity maximization are two of the most well-studied and widely-used problems in this area. Both problems admit sequential algorithms with optimal approximation factors of 2 in any metric space. However, finding distributed algorithms matching the same optimal approximation ratios has been open for more than a decade, with the best current algorithms having factors at least twice the optimal. In this paper, we settle this open problem by presenting constant-round distributed algorithms for k-center clustering and k-diversity maximization in the massively parallel computation (MPC) model, achieving an approximation factor of $2 + \varepsilon$ in any metric space for any constant $\varepsilon > 0$, which is essentially the best possible considering the lower bound of 2 on the approximability of both these problems. Our algorithms are based on a novel technique for approximating vertex degrees and finding a so-called k-bounded maximal independent set in threshold graphs, using only a constant number of MPC rounds. Other applications of our general technique is also implied, including an almost optimal $(3 + \varepsilon)$-approximation algorithm for the k-supplier problem in any metric space in the MPC model.

CCS CONCEPTS

• **Theory of computation** → **Massively parallel algorithms**; *Facility location and clustering*; • **Information systems** → *Information retrieval diversity*.

KEYWORDS

k-center clustering, diversity maximization, maximal independent set, massively parallel algorithms.

ACM Reference Format:
Alireza Haqi and Hamid Zarrabi-Zadeh. 2023. Almost Optimal Massively Parallel Algorithms for k-Center Clustering and Diversity Maximization . In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '23), June 17–19, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3558481.3591077

SPAA '23, June 17–19, 2023, Orlando, FL, USA

ACM ISBN 978-1-4503-9545-8/23/06…$15.00
https://doi.org/10.1145/3558481.3591077

1 INTRODUCTION

Given a set P of n points in a metric space, the objective in *k-center clustering* is to find a k-subset of P, called centers, so as the maximum distance of any point in P to its closest center is minimized. The objective in *k-diversity maximization* (also known as *remote-edge* diversity maximization) is to find a k-subset of P such that the minimum pairwise distance in the subset is maximized.

The k-center clustering and k-diversity maximization problems are known to be NP-hard [16, 24]. A simple greedy algorithm, called GMM, yields a 2-approximate solution to the k-center problem [16]. The algorithm repeatedly picks as center a point furthest away from the centers already chosen. Interestingly, the output of GMM provides a 2-approximate solution to k-diversity maximization as well [24]. No polynomial-time algorithm with an approximation factor better than 2 is possible for any of these two problems, unless P = NP [17, 24].

Motivated by applications in massive data processing, we consider distributed versions of k-center clustering and k-diversity maximization. In particular, we focus on the massively parallel computation (MPC) model, which is a standard abstraction of modern parallel frameworks, such as MapReduce, Hadoop, and Spark [20]. In this model, the input data is initially partitioned among a set of machines, each having a local memory. At each round, machines can perform polynomial computations on their local memory, and send messages to other machines, which are delivered at the beginning of the next round. The total size of messages sent and received by each machine at each round must not exceed the size of its local memory. It is generally desired to keep the number of rounds as small as possible, preferably a constant.

1.1 Our Contribution

In this paper, we study the k-center clustering and k-diversity maximization problems in the MPC model, and provide several new results as described below.

- We provide a massively parallel algorithm for the k-center problem in any metric space, with an almost optimal approximation factor of $2 + \varepsilon$. This greatly improves the best previously-known approximation factor of 4 available for the problem [22]. Our algorithm runs in a constant number of rounds, and requires $\tilde{O}(mk)$ communication per machine, where m is the number of machines.
- We also present a massively parallel algorithm for the k-diversity maximization problem in any metric space, achieving an almost optimal approximation factor of $2 + \varepsilon$ in constant MPC rounds. This significantly improves the best current approximation factor of 6 available for the problem [19]. Our algorithm uses $\tilde{O}(mk)$ communication per machine.

- For the k-supplier problem, which is a generalization of k-center, we present a massively parallel algorithm achieving an approximation factor of $3 + \varepsilon$, which is essentially the best possible, considering the lower bound of 3 proved on the approximability of this problem [18].

Our algorithms are based on a novel technique for computing a so-called k-bounded maximal independent set (MIS). Intuitively, a k-bounded MIS is a maximal independent set whose size is bounded from above by a parameter k. This concept is crucial for designing memory/communication efficient MPC algorithms for a variety of problems. As a byproduct of our main results, we also provide the following general tools:

- We present an algorithm for computing a k-bounded MIS in a threshold graph in a constant number of MPC rounds. Our algorithm contains several novel and non-trivial ingredients, including vertex degree approximation, vertex pruning, and round compression.
- We also present a novel degree-approximation algorithm that approximates the degrees of each vertex in a threshold graph to within a factor of $1 \pm \varepsilon$. To achieve this, we partition vertices of the graph into heavy and light subsets, based on a random sample taken from the vertices. Afterward, we estimate the degree of the heavy vertices within a $1 \pm \varepsilon$ factor w.h.p., while calculating the exact degrees for light vertices.

All our algorithms have constant rounds and use $\tilde{O}(mk)$ communication per machine. Considering the initial space needed for storing the input n points, the memory required by each machine is $\tilde{O}(n/m + mk)$. This memory requirement matches the best current coreset-based algorithms for k-center clustering [22] and k-diversity maximization [19], within a multiplicative factor of $\log(n)$. Note that each machine has a sublinear space in our model, otherwise, a single machine could see the whole points and run a local algorithm. Therefore, we assume that the number of machines is n^γ, for some $\gamma > 0$, to accommodate the whole input data.

1.2 Related Work

In 1985, Gonzales [16] presented his elegant algorithm for the k-center problem with an approximation factor of 2. Ravi et al. [24] proved that the same algorithm, which they call GMM, provides an approximation factor of 2 for the k-diversity maximization problem as well. Using a different parameter pruning approach, Hochbaum and Shmoys [18] presented another 2-approximation algorithm for the k-center problem that relies on finding maximal independent sets in pruned squared graphs. To handle datasets with noisy points, Charikar et al. [8] presented a 3-approximation algorithm for the k-center problem, when at most z input points can be ignored.

The MPC model has received considerable attention over the past decade, since its introduction by Karloff et al. [20]. In particular, efficient algorithms are provided in this model for several important graph problems, including graph connectivity, matching, coloring, independent sets, and clustering (see, e.g., [2, 5, 7, 12, 14, 15, 25]).

For the k-center clustering problem in the MPC model, Ene et al. [11] were the first to present a constant-round algorithm achieving an approximation factor of 10 using $O(kn^\varepsilon)$ memory per machine, for any $\varepsilon > 0$. Subsequently, Malkomes et al. [22] presented a two-round MPC algorithm for k-center that achieves an improved approximation factor of 4, using $O(\sqrt{nk})$ memory per machine. Their algorithm was based on running GMM on the union of a set of composable-coresets obtained again using GMM in each machine. To deal with noisy data, they also presented a 13-approximation MPC algorithm for k-center with outliers. Several other variants of k-center clustering are studied in the MPC model (see e.g., [3, 4, 6, 9]).

There are several algorithms for diversity maximization in the MPC model as well. In particular, the composable-coreset framework was first introduced by Indyk et al. [19] in the context of diversity maximization. They presented a two-round MPC algorithm for computing a 3-composable coreset for k-diversity maximization, yielding a 6-approximation algorithm for the problem in the MPC model, using $O(\sqrt{nk})$ memory per machine. They also considered other measures for diversity maximization, including remote-clique, that aims for maximizing the sum of pairwise distances in the k-subset. Improved MPC algorithms are provided for remote-clique diversity maximization using the notion of randomized composable coresets [1, 13, 23].

2 PRELIMINARIES

Let (U, d) be a metric space, and $V \subseteq U$ be an input point set of size n. We assume that the distance between any two points in the space can be obtained in $O(1)$ time. Let m be the number of machines. In our algorithms, we assume that the input set V is initially partitioned into m subsets $V_1, \ldots, V_m$, each stored in one of the machines. For a real value $\tau > 0$, we denote by G_τ a *threshold graph* on the vertex set V such that two vertices $u, v \in V$ are adjacent in G_τ if and only if $d(u, v) \leq \tau$. Note that the adjacency of two vertices in a threshold graph can be determined in $O(1)$ time by our distance oracle.

For a vertex v in a graph G, we denote by $N_G(v)$ the set of vertices in G adjacent to v. Whenever the graph G is clear from the context, we simply write $N(v)$ instead of $N_G(v)$. We define our new notion of k-bounded MIS as follows.

Definition 1. *Given a graph $G = (V, E)$, a vertex set $S \subseteq V$ is called a k-bounded MIS if either S is a maximal independent set of size at most k or S is an independent set of size exactly k.*

2.1 Diversity

Given a point set S in a metric space, the *diversity* of S, denoted by $\mathrm{div}(S)$, is the minimum pairwise distance in S, i.e., $\mathrm{div}(S) = \min_{p,q \in S} d(p, q)$. The *$k$-diversity* of S, denoted by $\mathrm{div}_k(S)$, is the maximum diversity over all k-subsets of S, i.e.,

$$\mathrm{div}_k(S) = \max_{Q \subseteq S, |Q| = k} \mathrm{div}(Q).$$

The k-diversity function is *monotone*, i.e., if $Q \subseteq S$, then $\mathrm{div}_k(Q) \leq \mathrm{div}_k(S)$, for any $1 \leq k \leq |Q|$.

2.2 GMM Algorithm

The following simple greedy algorithm, called GMM, computes a 2-approximation to both k-center and k-diversity problems in any metric space [16, 24]. The algorithm repeatedly picks a point furthest away from the set of points already chosen.

Algorithm 1 GMM(S)

1: Let $T = \{$an arbitrary point in $S\}$.
2: **while** $|T| < k$ **do**
3: Let p be a point in S maximizing $d(p, T)$.
4: $T = T \cup \{p\}$.
5: **end while**
6: Return T.

If r denotes the minimum pairwise distance in the set $T = \text{GMM}(S)$, then the following two properties, known as *anti-cover properties*, hold:

- $\forall p \in T : d(p, T \setminus \{p\}) \geq r$
- $\forall p \in S : d(p, T) \leq r$

2.3 Chernoff Bounds

We use the following statement of Chernoff bound in this paper.

Theorem 2 ([10]). *Let $X_1, \ldots, X_n$ be independent random variables taking values in $[0, 1]$. Define $X = \sum_{i=1}^{n} X_i$ and let $\mu = E[X]$. Then, for any $\varepsilon \in [0, 1]$, we have*

$$\Pr[X \leq (1 - \varepsilon)\mu] \leq \exp\left(\frac{-\varepsilon^2 \mu}{2}\right), \tag{1}$$

and

$$\Pr[X \geq (1 + \varepsilon)\mu] \leq \exp\left(\frac{-\varepsilon^2 \mu}{3}\right). \tag{2}$$

3 $(2 + \varepsilon)$-APPROXIMATION MPC ALGORITHM FOR k-DIVERSITY

In this section, we present a $(2+\varepsilon)$-approximation algorithm for the k-diversity problem in the MPC model, assuming that a constant-round MPC algorithm is available for the k-bounded MIS. The pseudo-code of our algorithm is presented in Algorithm 2. The algorithm starts by finding a constant-factor approximation to k-diversity on the input set, and then finds the largest threshold for which a k-bounded MIS of size k exists in the corresponding threshold graph. We show in Theorem 3 that the returned k-bounded MIS provides a $(2 + \varepsilon)$-approximation to k-diversity.

Algorithm 2 $(2 + \varepsilon)$-Approximation MPC Diversity Maximization

1: Let $T_i = \text{GMM}(V_i)$ and $r_i = \text{div}(T_i)$, for $1 \leq i \leq m$.
2: Compute $S = \text{GMM}(T = \bigcup_{i=1}^{m} T_i)$, and let $r_0 = \text{div}(S)$.
3: Let $r = \max_{0 \leq i \leq m} \{r_i\}$, and let Q be the k points realizing r.
4: Define $\tau_i = r \cdot (1 + \varepsilon)^i$, for $i = 0, 1, \ldots, t = \log_{1+\varepsilon}(4) + 1$.
5: Let $M_0 = Q$, and M_i be a k-bounded MIS in G_{τ_i}, for $0 < i \leq t$.
6: Find an index j such that $|M_j| = k$ and $|M_{j+1}| < k$.
7: Return M_j.

Theorem 3. *Algorithm 2 computes a $(2 + \varepsilon)$-approximation to the k-diversity maximization problem in any metric space in $O(\log \frac{1}{\varepsilon})$ MPC rounds, for any constant $\varepsilon > 0$.*

Proof. Let $O = \{o_1, \ldots, o_k\}$ be an optimal solution to k-diversity and let $r^* = \text{div}(O)$. We first show that the value r computed in line 3 of the algorithm is a 4-approximation to r^*. Let C_i be a ball of radius $r^*/4$ centered at o_i, for $1 \leq i \leq k$. If each C_i contains a point $t_i \in T$, then the set $\{t_1, \ldots, t_k\}$ has diversity at least $r^*/2$, and hence, running GMM on T yields a 4-approximation. Otherwise, one of the balls, say C_j, contains no point from T. Suppose that o_j is contained in machine i. Then, by anti-cover property, $r_i \geq d(o_j, T_i) \geq r^*/4$. Therefore, $\max_i \{r_i\}$ is a 4-approximation in this case. In any case, r is a 4-approximation to r^*, which means $r \leq r^* \leq 4r$.

Now, let j be an index such that $|M_j| = k$ and $|M_{j+1}| \leq k - 1$. Note that such an index exists, because $|M_0| = k$ and $|M_t| < k$ for $t = \log_{1+\varepsilon}(4) + 1$, as $\tau_t > 4r \geq r^*$. Clearly, M_j is a feasible solution with diversity τ_j, because any two points in M_j have distance at least τ_j. Moreover, M_{j+1} is a maximal independent set of size less than k. Consider the balls of radius τ_{j+1} around the points in M_{j+1}. Any point in V is contained in at least one of the balls, otherwise, M_{j+1} could be extended by a point outside the balls, contradicting its maximality. Therefore, by the pigeonhole principle, there are at least two points $o_1, o_2 \in O$ contained in the same ball of radius τ_{j+1}. Thus, by triangle inequality, $d(o_1, o_2) \leq 2\tau_{j+1}$, which implies $\text{div}(O) \leq 2\tau_{j+1} = 2(1 + \varepsilon)\tau_j$. As such, $r^* \leq 2(1 + \varepsilon)\tau_j$, which yields the approximation factor of the algorithm. Note that an index j with $|M_j| = k$ and $|M_{j+1}| < k$ can be found via a binary search in $O(\log \frac{1}{\varepsilon})$ rounds. □

As a side product, lines 1–3 of Algorithm 2 provide a simple two-round 4-approximation algorithm for the k-diversity problem, improving upon the current two-round 6-approximation MPC algorithm available via composable coresets [19].

4 APPROXIMATING VERTEX DEGREES

Our massively parallel algorithm for computing k-bounded MIS relies on knowing the (approximate) vertex degrees in a threshold graph. In this section we present an algorithm that approximates the degree of each vertex to within a factor of $1 \pm \varepsilon$. The main idea behind our algorithm is to sample vertices in each machine with probability $p = 1/m$, and then send sampled vertices of each machine to all other machines. After the sampling phase, the algorithm takes the number of vertices in the sample adjacent to each vertex as an estimate of its actual degree, when multiplied by p. This works for vertices of high degree. However, this estimate might not be accurate enough for low-degree (light) vertices. As such, the algorithm switches to computing the degree of light vertices exactly. The extra cost for computing the exact degree of light vertices can be only afforded if the number of light vertices is not too high. Fortunately, we can show that if there are too many such light vertices, then an independent set of size k can be extracted directly from the light vertices.

The pseudo-code of our algorithm is provided in Algorithm 3. In this algorithm, δ is a constant which will be fixed later in our analysis. For any vertex $v \in V$, we denote by $d(v)$ the degree of v in the input graph, and by $d_i(v)$ the number of vertices in V_i adjacent to v, where V_i is the set of vertices stored in machine i. As such, $d(v) = \sum_{i=1}^{m} d_i(v)$. Note that in threshold graphs, the adjacency of any two vertices can be determined by the distance oracle, if the value of the threshold is known. We partition graph vertices into light and heavy as follows:

Algorithm 3 MPC Degree Approximation

1: Each machine i takes a sample S_i by picking any vertex of V_i with probability $p = 1/m$.
2: Each machine sends its sampled set to all other machines.
3: Let $S = \bigcup_{i=1}^{m} S_i$. ($S$ is available to all machines.)
4: Let L be the set of all light vertices in all machines.
5: **if** $|L| > 2\delta\, mk \ln(n)$ **then**
6: Compute an independent set of size k in L and terminate.
7: **else**
8: Each machine sends its light vertices to all other machines.
9: Each machine i sends $d_i(v)$ for each light vertex v to others.
10: **for** each vertex $v \in V_i$ in machine i **do**
11: Set $p_v = \sum_{i=1}^{m} d_i(v)$, if v is light.
12: Set $p_v = \frac{1}{p} \cdot |N(v) \cap S|$, if v is heavy.
13: **end for**
14: **end if**

Definition 4. *Given a sample $S \subseteq V$, we call a vertex $v \in V$ heavy w.r.t. S, if $|N(v) \cap S| \geq \delta \ln(n)$, and light otherwise.*

The following two lemmas show that if the number of light vertices is too large, then an independent set of size k can be extracted directly from the light vertices.

Lemma 5. *For all light vertices $v \in V$, $d(v) < 2\delta\, m \ln(n)$ w.h.p.*

Proof. Fix a vertex v. Let $X = |N(v) \cap S|$. Note that X is the sum of $d(v)$ independent Bernoulli random variables, with $E[X] = p \cdot d(v)$. By Chernoff bound (Theorem 2) we have

$$\Pr[X \leq (1-\gamma)E[X]] \leq e^{\frac{-\gamma^2 E[X]}{2}}.$$

Suppose by contrary that $d(v) \geq 2\delta\, m \ln(n)$. Then, $E[X] \geq p \cdot (2\delta\, m \ln n) = 2\delta \ln n$. Now, using Chernoff bound with $\gamma = \frac{1}{2}$ we have

$$\Pr[X \leq \delta \ln n] \leq n^{\frac{-\delta}{4}} \leq \frac{1}{n^3},$$

for all $\delta \geq 12$. Therefore, with probability at least $1 - 1/n^3$, if v is a light vertex, then $d(v) < 2\delta\, m \ln(n)$. Since there are at most n light vertices, the inequality $d(v) < 2\delta\, m \ln(n)$ holds for all light vertices with probability at least $1 - 1/n^2$. □

Lemma 6. *If the number of light vertices exceeds $2\delta\, m \ln(n)$, then we can find an independent set of size k w.h.p. in $O(1)$ rounds using $\tilde{O}(mk)$ total communication.*

Proof. We do as follows. Each machine first sends the number of its light vertices as a single integer number to the central machine. The central machine computes the value $\rho = (2\delta\, m \ln(n))/|L|$, where $|L|$ is the total number of light vertices in all machines, and sends ρ to each machine. Each machine then sends ρ fraction of its light vertices to the central machine. Let P be the set of light vertices received by the central machine. We can now run the greedy algorithm locally on P to find a maximal independent set. By Lemma 5, $d(v) < 2\delta\, m \ln(n)$ for all vertices in P w.h.p. Therefore, at each iteration of the greedy algorithm, at most $2\delta\, m \ln(n)$ vertices of P are removed. Thus, the greedy algorithm has at least

$$\frac{|P|}{2\delta m \ln n} = k$$

iterations w.h.p. The lemma statement follows. □

The light vertices are thus handled properly. In the following, we provide $(1 \pm \varepsilon)$ approximation for the heavy vertices.

Lemma 7. *For all heavy vertices $v \in V$, $d(v) > \frac{\delta}{2} m \ln(n)$ w.h.p.*

Proof. Fix a vertex v. Let $X = |N(v) \cap S|$. We know that $E[X] = p \cdot d(v) = \frac{d(v)}{m}$. By Chernoff bound (Theorem 2) we have

$$\Pr[X \geq (1+\gamma)E[X]] \leq e^{\frac{-\gamma^2 E[X]}{3}}.$$

Suppose by contrary that $d(v) \leq \frac{\delta}{2} m \ln n$. Therefore, by setting $\gamma = \frac{\delta m \ln n}{d(v)} - 1$, we have

$$\Pr[X \geq (1+\gamma)E[X]] = \Pr[X \geq \delta \ln n] \leq e^{\frac{-\gamma^2 d(v)}{3m}}.$$

Since $\frac{\delta m \ln n}{d(v)} \geq 2$, we have $\gamma = \frac{\delta m \ln n}{d(v)} - 1 \geq \frac{\delta m \ln n}{2d(v)}$. Therefore,

$$\gamma^2 \frac{d(v)}{m} \geq \frac{\delta^2 m \ln^2 n}{4d(v)} \geq \frac{\delta \ln n}{2},$$

where the last inequality holds since $d(v) \leq \frac{\delta}{2} m \ln n$. Plugging into the Chernoff bound, we get

$$\Pr[X \geq \delta \ln n] \leq e^{\frac{-\delta \ln n}{6}} \leq n^{\frac{-\delta}{6}} \leq 1/n^3,$$

for all $\delta \geq 18$. Since there are at most n heavy vertices, the inequality $d(v) \geq \frac{\delta}{2} m \ln n$ holds for all heavy vertices with probability at least $1 - 1/n^2$. □

Lemma 8. *For every heavy vertex v, $\frac{1}{p}|N(v) \cap S|$ approximates $d(v)$ within a factor of $1 \pm \varepsilon$ w.h.p.*

Proof. Fix a heavy vertex v, and let $X = |N(v) \cap S|$. We have $E[X] = p \cdot d(v) \geq \frac{\delta}{2} \ln(n)$ by Lemma 7 w.h.p. Now, Chernoff bound implies

$$\begin{aligned}\Pr[X \leq (1-\varepsilon)E[X]] + \Pr[X \geq (1+\varepsilon)E[X]] &\leq e^{\frac{-\varepsilon^2 E[X]}{2}} + e^{\frac{-\varepsilon^2 E[X]}{3}} \\ &\leq 2n^{-\varepsilon^2 \delta/6},\end{aligned}$$

which is at most $2/n^2$ for all $\delta \geq 12/\varepsilon^2$. □

Theorem 9. *There is a constant-round MPC algorithm that approximates the degree of each vertex in a threshold graph to within a factor of $1 \pm \varepsilon$, or returns an independent set of size k, w.h.p., using $\tilde{O}(mk)$ communication per machine.*

Proof. Consider Algorithm 3. If there are too many light vertices in the graph, then by Lemma 6, the algorithm returns an independent set of size k, w.h.p. Otherwise, by Lemma 8, the algorithm correctly approximates the degree of each vertex to within a factor of $1 \pm \varepsilon$, w.h.p. If the first case happens, we can keep the communication within $\tilde{O}(mk)$ as follows: each machine sends the number of its light vertices to the central machine, and the central machine computes the value $\rho = 2\delta\, mk \ln(n)/|L|$, where $|L|$ is the total number of light vertices. Each machine then sends ρ fraction of its light vertices to the central machine, making a set of $2\delta\, mk \ln(n)$ light vertices, from which the central machine extracts an independent set of size k. This guarantees that the total number of vertices sent to the central machine is $\tilde{O}(mk)$ in this case. In the second case,

Algorithm 4 Massively Parallel k-Bounded MIS

1: MIS $\leftarrow \varnothing$
2: **while** $G \neq \varnothing$ and $|\text{MIS}| < k$ **do**
3: Compute a $(1 \pm \varepsilon)$-approx. p_v of $d(v)$ for each vertex v.
4: If a k-bounded MIS is found in line 3, return it and terminate.
5: Each machine i takes m samples $S_i^1, \ldots, S_i^m$, where each sample contains any vertex $v \in V_i$ with probability $\frac{1}{2p_v}$.
6: **if** $\sum_{v \in V} \frac{1}{2p_v} > 10\,k \ln n$ **then** ▷ pruning step
7: Let $T_j = \text{trim}(\bigcup_{i=1}^m \text{trim}(S_i^j))$, for $1 \le j \le m$.
8: Return a k-subset of largest T_j and terminate.
9: **end if**
10: Send all samples S_i^j to the central machine.
11: **for** $j = 1, \ldots, m$ **do** (in central machine)
12: Let $S_j = (\bigcup_{i=1}^m S_i^j) \cap V(G)$.
13: Compute $M_j = \text{trim}(S_j)$.
14: Add M_j to MIS.
15: Remove $M_j \cup N(M_j)$ from G.
16: **end for**
17: Central machine sends MIS to other machines.
18: Each machine removes MIS $\cup\, N(\text{MIS})$ from its vertices.
19: **end while**
20: Return a k-subset of MIS.

there are $\tilde{O}(mk)$ light vertices in total. Therefore, the total number of vertices sent/received by each machine is $\tilde{O}(mk)$. □

5 FINDING k-BOUNDED MIS

In this section, we provide our massively parallel algorithm for finding a k-bounded MIS in a threshold graph. The pseudo-code of our algorithm is presented in Algorithm 4. At each round of the algorithm, each machine i computes a $(1 \pm \varepsilon)$-approximation p_v of $d(v)$ for each vertex $v \in V_i$. It then takes m samples $S_i^1, \ldots, S_i^m$ independently at random, where each vertex $v \in V_i$ appears in any of the samples with probability $1/2p_v$. Each machine then sends all its sampled sets (after a size limit check) to the central machine. The central machine takes independent sets from union of sampled sets in order, and updates G respectively. The algorithm continues until either G becomes empty, or an independent set of size k is found.

Given a set S of vertices in G, we find an independent set in S using the following function, which is a local variant of the Luby's algorithm [21]:

$$\text{trim}(S) = \{v \in S \,:\, p_v > p_u \text{ for all } u \in N(v) \cap S\}.$$

In the following, we analyze the correctness and the round complexity of our algorithm. To simplify the analysis, we henceforth fix the precision of the degree approximation algorithm to $\varepsilon = 1/6$.

Lemma 10. *Let S be a set containing any vertex v of a graph G with probability $\frac{1}{2p_v}$, where $p_v \ge (1-\varepsilon)d_G(v)$. Then $\text{trim}(S)$ contains any vertex v of G with probability at least $\frac{1}{5p_v}$.*

Proof. Let $M = \text{trim}(S)$, and let v be an arbitrary vertex in G. The probability that v is contained in M can be written conditionally as follows:

$$\Pr[v \in M] = \Pr[v \in S] \cdot \Pr[v \in M \,:\, v \in S].$$

To obtain a lower bound on $\Pr[v \in M \,:\, v \in S]$, we upper bound $\Pr[v \notin M \,:\, v \in S]$. The event that a vertex $v \in S$ is not in M can only occur if it adjacent to a vertex u with $p_u \ge p_v$. Therefore,

$$\Pr[v \notin M \,:\, v \in S] \;\le \sum_{\substack{u \in N(v),\\ p_u \ge p_v}} \frac{1}{2p_u} \;\le\; \frac{d_G(v)}{2p_v} \;\le\; \frac{1}{2(1-\varepsilon)},$$

which is at most $\frac{3}{5}$ by setting $\varepsilon = \frac{1}{6}$. Therefore, $\Pr[v \in M] \ge \frac{1}{2p_v} \cdot \left(1 - \frac{3}{5}\right) = \frac{1}{5p_v}$. □

To express how much a vertex v tends to be removed from G during the algorithm, we define a function λ which intuitively indicates the probability that a vertex is removed by its neighbors.

Definition 11. *We define a function $\lambda : V \mapsto \mathbb{R}$ as follows:*

$$\lambda(v) = \sum_{u \in N(v)} \frac{1}{2p_u}.$$

Moreover, for a subset $Z \subseteq N(v)$, we define

$$\lambda_Z(v) = \sum_{u \in Z} \frac{1}{2p_u}.$$

The following lemma provides a lower bound on the probability of removing a vertex v depending on the value of its function $\lambda(v)$.

Lemma 12. *After each iteration of the for loop in Algorithm 4, a vertex v is removed with probability at least $\min(\frac{\lambda(v)}{10}, \frac{1}{25})$.*

Proof. Let G_0 be the graph G at the beginning of the current round, and let G_j be the graph G at the beginning of iteration j of the for loop. By line 3 of the algorithm, we have $p_v \ge (1-\varepsilon)d_{G_0}(v)$ for all v. Note that the p_v values do not change during the whole iterations of a round. However, since neighbors of a vertex can be only removed during iterations, we have $d_{G_j}(v) \le d_{G_0}(v)$, and the inequity $p_v \ge (1-\varepsilon)d_{G_j}(v)$ still holds for all v and for all j. Moreover, the set S_j at the beginning of each iteration j can be seen as a sample taken directly from vertices of G_j, where each vertex v of G_j is contained in S_j with probability $\frac{1}{2p_v}$.

Now, fix an iteration j of the for loop, and consider a vertex v in G_j. Let E be the event that v is removed from the graph in line 15 of the algorithm. We give a lower bound on the probability of E based on the value of $\lambda(v)$. We consider the following two cases.

Case 1: $\lambda(v) \le \frac{1}{10}$. The probability of removing v is at least as large as the probability that a neighbor of v joins M_j. Hence,

$$\Pr[E] \;\ge \sum_{w \in N(v)} \Pr[w \in M_j] \;-\; \sum_{\substack{u, w \in N(v)\\ u \ne w}} \Pr[w \in M_j \text{ and } u \in M_j].$$

But, the event of joining a point to M_j is contained in the event of joining the point to S_j. Thus the second term of RHS is upper bounded by the probability that both points are contained in S_j. Therefore,

$$\Pr[E] \;\ge \sum_{w \in N(v)} \Pr[w \in M_j] \;-\; \sum_{\substack{u, w \in N(v)\\ u \ne w}} \Pr[w \in S_j \text{ and } u \in S_j].$$

By Lemma 10, we have

$$\sum_{w\in N(v)} \Pr[w \in M_j] \;\geq\; \sum_{w\in N(v)} \frac{1}{5p_w}.$$

Moreover, since sampling points in S_j are independent form each other, we have $\Pr[w \in S_j \text{ and } u \in S_j] = \frac{1}{2p_w} \cdot \frac{1}{2p_u}$. Therefore,

$$\begin{aligned}
\Pr[E] \;&\geq\; \sum_{w\in N(v)} \frac{1}{5p_w} \;-\; \sum_{\substack{u,w\in N(v)\\ u\neq w}} \left(\frac{1}{2p_w}\cdot\frac{1}{2p_u}\right)\\
&\geq\; \sum_{w\in N(v)} \frac{1}{p_w}\left(\frac{1}{5} - \sum_{u\in N(v)} \frac{1}{4p_u}\right)\\
&\geq\; \sum_{w\in N(v)} \frac{1}{p_w}\left(\frac{1}{5} - \frac{\lambda(v)}{2}\right)\\
&=\; 2\lambda(v)\left(\frac{1}{5} - \frac{\lambda(v)}{2}\right) \;\geq\; \frac{3\lambda(v)}{10},
\end{aligned}$$

where the last inequality is implied by plugging $\lambda(v) \leq \frac{1}{10}$ into the last parenthesis.

Case 2: $\lambda(v) > \frac{1}{10}$. In this case, we show that a subset $Z \subseteq N(v)$ can be found such that $\frac{1}{10} \leq \lambda_Z(v) \leq \frac{1}{5}$, or we have a single vertex $w \in N(v)$ satisfying $p_w \leq 5$. If there is a vertex $w \in N(v)$ such that $p_w \leq 5$, then $\Pr[E] \geq \Pr[w \in M_j] \geq \frac{1}{5p_w} \geq \frac{1}{25}$, and we are done. Otherwise, we can find a subset $Z \subseteq N(v)$ such that $\frac{1}{10} \leq \lambda_Z(v) \leq \frac{1}{5}$, by greedily removing a vertex with highest weight from $N(v)$ as long as $\lambda_Z(v) > \frac{1}{5}$.

Using the same analysis as in the previous case, but just using $\lambda_Z(v)$ instead of $\lambda(v)$, we conclude that $\Pr[E] \geq 2\lambda_Z(v)(\frac{1}{5} - \frac{1}{2}\lambda_Z(v))$. Since the function $x(1 - \frac{5x}{2})$ is concave, its minimum in any interval is attained at one of the endpoints. In particular, the minimum of the function over the interval $[\frac{1}{10}, \frac{1}{5}]$ is attained at $x = \frac{1}{10}$, giving a value of $\frac{3}{40}$ which is less than $\frac{1}{10}$. It yields $\Pr[E] \geq \frac{1}{25}$ in this case. □

Note that in line15 of the algorithm, we only need to remove $M_j \cup N(M_j)$ from a local copy of G that only consists of all sample points received from the machines in the current round, whose total size is $\tilde{O}(mk)$ by our pruning step. Removing the points globally from G is actually performed in lines 17-18 of the algorithm.

We can now use Lemma 12 to prove the round complexity of our algorithm in the following theorem.

Theorem 13. *Algorithm 4 terminates in $O(\frac{1}{\gamma})$ rounds w.h.p., when $m = n^\gamma$.*

Proof. We first show that after each round of Algorithm 4, the number of edges of the graph decreases by a factor of $\frac{1}{5}m^{\frac{1}{2}}$ w.h.p. Fix a round of the algorithm, and let $\lambda_j(v)$ denote the value of $\lambda(v)$ in the jth iteration of the for loop in that round. Note that for any vertex v, $\lambda_j(v)$ is decreasing in j, as the neighbors of v can be only removed during iterations. If $\lambda_m(v) \geq m^{\frac{-1}{2}}$, then the probability that v is not removed after m iterations is at most

$$\begin{aligned}
\prod_{1\leq j\leq m}\left(1 - \min\left(\frac{3\lambda_j(v)}{10}, \frac{1}{25}\right)\right) &\leq \prod_{1\leq j\leq m} e^{-\min\left(\frac{3\lambda_j(v)}{10}, \frac{1}{25}\right)}\\
&\leq e^{\frac{-3T\cdot m^{\frac{-1}{2}}}{10}} = e^{\frac{-3T^{\frac{1}{2}}}{10}} \leq e^{-2\ln(n)} = \frac{1}{n^2},
\end{aligned}$$

where the last inequality holds for all $m \geq 45\ln^2 n$. Therefore, for each remaining vertex v, we have $\lambda_m(v) \leq m^{\frac{-1}{2}}$ w.h.p.

Now, consider the graph $G = (V, E)$ after m iterations. Denote by $N^+(v)$ the set of neighbors u of v such that $p_u \leq p_v$. Note that $|E| \leq \sum_{v\in V} |N^+(v)|$, as each edge is counted at least once in the right-hand side. Moreover, we have $\lambda_m(v) \geq \frac{|N^+(v)|}{2p_v}$. If $G_0 = (V_0, E_0)$ denote the initial graph just before the for loop starts, we have

$$\begin{aligned}
|E| \;\leq\; \sum_{v\in V} |N^+(v)| \;&\leq\; 2\sum_{v\in V} p_v\lambda_m(v)\\
&\leq\; 2(1+\varepsilon)\sum_{v\in V_0} d(v)m^{-\frac{1}{2}}\\
&= 2\left(1+\frac{1}{6}\right)(2\cdot|E_0|)m^{-\frac{1}{2}} \;\leq\; 5m^{-\frac{1}{2}}|E_0|.
\end{aligned}$$

Since the number of edges decreases by a factor of $\frac{1}{5}m^{\frac{1}{2}} = \frac{1}{5}n^{\frac{\gamma}{2}}$ w.h.p. in each round, the algorithm terminates in $O(\frac{1}{\gamma})$ rounds with high probability. □

The total size of samples taken by the machines may exceed $\tilde{O}(mk)$. As such, the algorithm performs a check before sending the samples to the central machine and runs a pruning step if required. The following theorem describes how this pruning step keeps the total sample size within a desired range.

Theorem 14. *If the total number of sampled points exceeds $\tilde{O}(mk)$, then the pruning step returns an independent set of size k w.h.p.*

Proof. Let $S_j = \bigcup_{i=1}^m S_i^j$ and $X = |S_j|$. Since each vertex v is added to S_j with probability $\frac{1}{2p_v}$, we have $E[X] = \sum_{v\in V} \frac{1}{2p_v}$. Note that $E[X]$ is the same for all samples S_j, $1 \leq j \leq m$. We consider the following two cases.

Case 1. $E[X] \leq 10\,k\ln(n)$: By Chernoff bound, we have

$$\Pr[X \geq (1+\gamma)E[X]] \leq e^{\frac{-\gamma^2 E[X]}{3}}$$

Setting $\gamma = \frac{20k\ln(n)}{E[X]} - 1$ yields

$$\begin{aligned}
\Pr[X \geq 20\,k\ln(n)] \leq e^{\frac{-(20k\ln(n)-E[X])^2}{3E[X]}} &\leq e^{\frac{-(10k\ln(n))^2}{3E[X]}}\\
&\leq e^{-\frac{10}{3}k\ln(n)} \leq \frac{1}{n^2}.
\end{aligned}$$

Therefore, in this case, the size of S_j is at most $20\,k\ln(n)$ w.h.p., and hence, all samples together have size $20\,mk\ln(n) = \tilde{O}(mk)$. Hence, no pruning is done by the algorithm in this case.

Case 2. $E[X] \geq 10\,k\ln(n)$: Let $M_j = \text{trim}(S_j)$ and $Y = |M_j|$. We will show that $Y \geq k$ w.h.p. in this case, meaning that M_j contains an independent set of size k. Note that M_j is a subset of $T_j = \text{trim}(\bigcup_{i=1}^m \text{trim}(S_i^j))$, defined in the algorithm. This is because any vertex contained is M_j is necessarily contained in T_j, since a vertex is removed by the trim function only if it is connected

to a vertex of larger weight. Therefore, proving $Y \geq k$ implies that $|T_j| \geq k$ as well. Hence, we focus on proving the probability of $Y \geq k$. Recall that for each vertex $v \in V$, $\Pr[v \in S_j] = \frac{1}{2p_v}$ and $\Pr[v \in M_j] \geq \frac{1}{5p_v}$ by Lemma 10. Therefore, $E[Y] \geq \frac{2}{5}E[X]$. Moreover, since $X \geq Y$, we have $\Pr[Y \geq 5E[Y]] \leq \Pr[X \geq 2E[X]]$. By applying Chernoff bound on X we get:

$$\Pr[X \geq 2E[X]] \leq e^{\frac{-E[X]}{3}} \leq e^{-k\ln(n)} \leq \frac{1}{n}.$$

Now, we rewrite the expected value of Y as follows:

$$\begin{aligned} E[Y] \leq \Pr[Y < \frac{1}{4}E[Y]] \cdot \frac{1}{4}E[Y] \\ + \Pr[\frac{1}{4}E[Y] \leq Y < 5E[Y]] \cdot 5E[Y] \\ + \Pr[Y \geq 5E[Y]] \cdot n, \end{aligned} \tag{3}$$

where for each range, the probability of Y being in the range is multiplied by the largest possible value of Y in that range. If $\Pr[\frac{1}{4}E[Y] \leq Y < 5E[Y]] < \frac{1}{10}$, then by inequality (3), we have:

$$E[Y] < 1 \cdot \frac{1}{4}E[Y] + \frac{1}{10} \cdot 5E[Y] + \frac{1}{n} \cdot n = \frac{3}{4}E[Y] + 1 < E[Y],$$

which is a contradiction. Therefore, $\Pr[\frac{1}{4}E[Y] \leq Y < 5E[Y]] \geq \frac{1}{10}$, which implies $\Pr[Y < \frac{1}{4}E[Y]] \leq \frac{9}{10}$, and hence $\Pr[Y < k] \leq \frac{9}{10}$, as $E[Y] \geq \frac{2}{5}E[X] \geq 4k$. By taking $10\ln(n)$ independent sample S_j, as in the pruning step of the algorithm, the probability $\Pr[Y < k]$ is reduced to $1/n$. Therefore, $Y \geq k$ w.h.p. in this case, and we are done. Note that in line 7 of the algorithm, whenever $\text{trim}(S_i^j)$ has size more than k, we can return a k-subset of it directly and terminate. This ensures that each of the sets $\bigcup_{i=1}^{m} T_j$ has size $O(mk)$. Therefore, the total communication in the pruning step is $\tilde{O}(mk)$. □

We summarize the main result of this section in the following theorem.

Theorem 15. *There is a constant-round MPC algorithm that computes a k-bounded MIS with high probability using $\tilde{O}(mk)$ communication and $\tilde{O}(n/m + mk)$ memory per machine.*

Proof. There are three possibilities for Algorithm 4 to terminate. The first two cases guarantee an independent set of size k (lines 4 and 8 of the algorithm), and the third case guarantees the output of the algorithm to be a maximal independent set. Therefore, the algorithm returns a k-bonded MIS in any case.

At each round, the communication used by each machine in the degree approximation and the pruning step is $\tilde{O}(mk)$ by Theorems 9 and 14. Moreover, the total size of samples sent to the central machine is $\tilde{O}(mk)$ by our pruning step (Theorem 14). Considering the initial space needed for storing the input n points, the memory required by each machine is $\tilde{O}(n/m + mk)$. The round complexity of the algorithm is implied by Theorem 13. □

6 OTHER APPLICATIONS OF k-BOUNDED MIS

In this section, we show how our MPC algorithm for the k-bounded MIS can be used to obtain improved algorithms for other problems in the MPC model. In particular, we provide a $(2+\varepsilon)$-approximation algorithm for the k-center problem and an almost optimal $(3+\varepsilon)$-approximation algorithm for the k-supplier problem in any metric space in the MPC model.

6.1 A $(2+\varepsilon)$-Approximation MPC Algorithm for k-Center

Given two point sets X and Y in a metric space, we define

$$r(X, Y) = \max_{x \in X} d(x, Y).$$

The objective in the k-center problem is to find a subset $C \subseteq V$ of size k minimizing $r(V, C)$. We first prove the following lemma regarding the GMM algorithm.

Lemma 16. *If $T = \text{GMM}(S)$, then $r(S, T) \leq \text{div}_{k+1}(S)$.*

Proof. Let p be a point in S maximizing $d(p, T)$, and let $r = d(p, T) = r(S, T)$. Note that $T \cup \{p\}$ is a set of $k+1$ points with diversity r. Moreover, $\text{div}_{k+1}(T \cup \{p\}) \leq \text{div}_{k+1}(S)$ by monotonicity of the diversity function. Hence, $r \leq \text{div}_{k+1}(S)$. □

Now, we provide a $(2+\varepsilon)$-approximation MPC algorithm for the k-center problem. The pseudo-code of our algorithm in presented in Algorithm 5. Recall that G_τ denotes a graph on the input set V such that two vertices $u, v \in V$ are adjacent if $d(u, v) \leq \tau$.

Algorithm 5 $(2+\varepsilon)$-Approximation MPC k-Center

1: Let $T_i = \text{GMM}(V_i)$ and $T = \bigcup_{i=1}^{m} T_i$.
2: Compute $Q = \text{GMM}(T)$.
3: Let $r = r(V, Q)$.
4: Define $\tau_i = r/(1+\varepsilon)^i$, for $i = 0, \ldots, t = \log_{1+\varepsilon}(4) + 1$.
5: Let $M_0 = Q$, and let M_i be a $(k+1)$-bounded MIS in graph G_{τ_i}, for $0 < i \leq t$.
6: Find an index j such that $|M_j| \leq k$ and $|M_{j+1}| = k+1$.
7: Return M_j.

Theorem 17. *Algorithm 5 computes a $(2+\varepsilon)$-approximation to the k-center problem in any metric space in $O(\log \frac{1}{\varepsilon})$ MPC rounds, for any constant $\varepsilon > 0$. using $\tilde{O}(mk)$ communication and $\tilde{O}(n/m + mk)$ memory per machine, where m is the number of machines.*

Proof. Let $C^* = \{c_1, \ldots, c_k\}$ be an optimal k-center solution with radius r^*. We first show that the value r computed in line 3 of the algorithm is a 4-approximation to r^*. Note that $\text{div}_{k+1}(V) \leq 2r^*$. This is because in any $(k+1)$-subset of V, at least two points are covered by the same center in C^*, and hence, their distance is at most $2r^*$. Now, fix a point v in machine i. By Lemma 16, $r(V_i, T_i) \leq \text{div}_{k+1}(V_i)$. Therefore, there is a point $t \in T_i$ such that $d(v, t) \leq \text{div}_{k+1}(V_i) \leq \text{div}_{k+1}(V) \leq 2r^*$. Moreover, by Lemma 16, $r(T, Q) \leq \text{div}_{k+1}(T)$. Therefore, for any point $t \in T$, there is a point $q \in Q$ such that $d(t, q) \leq \text{div}_{k+1}(T) \leq \text{div}_{k+1}(V) \leq 2r^*$. Hence,

$$d(v, q) \leq d(v, t) + d(t, q) \leq 2r^* + 2r^* = 4r^*.$$

Thus, r is a 4-approximation to r^*, i.e., $r/4 \leq r^* \leq r$.

Now, let j be an index such that $|M_j| \leq k$ and $|M_{j+1}| = k+1$. Note that such an index exists, because $|M_0| = k$ and $|M_t| = k+1$. If $|M_t| \leq k$, then M_t is a maximal independent set in G_{τ_t}, and hence, it is a k-center solution with radius τ_t, which is a contradiction,

because $\tau_t < r/4 \leq r^*$. Clearly, M_j is a k-center solution with radius τ_j, since any point in $V \setminus M_j$ has distance at most τ_j from M_j. Moreover, M_{j+1} contains $k+1$ points of pairwise distance more than τ_{j+1}. By the pigeonhole principle, two points of M_{j+1} are covered by the same center in C^*. Therefore, $r^* \geq \frac{1}{2}\tau_{j+1} = \frac{1}{2}\tau_j/(1+\varepsilon)$. Thus, $\tau_j \leq 2(1+\varepsilon)r^*$, which yields the approximation factor of the algorithm. Note that an index j with $|M_j| \leq k$ and $|M_{j+1}| = k+1$ can be found via a binary search in $O(\log \frac{1}{\varepsilon})$ rounds. □

6.2 An MPC Algorithm for k-Supplier

In the k-supplier problem, we are given a set S of suppliers (facilities) and a set C of customers, and the objective is to select k suppliers such that the maximum distance between customers and suppliers is minimized. In other words, the objective is to find a subset $Q \subseteq S$ of size k minimizing $r(C, Q)$.

In the following, we provide a constant-round MPC algorithm for the k-supplier problem, achieving an approximation factor $3+\varepsilon$, which is essentially the best possible, considering the lower bound of 3 proved on the approximability of this problem [18]. Hereafter, we assume that each machine i stores a subset $C_i \subseteq C$ of customers, and a subset $S_i \subseteq S$ of suppliers. The pseudo-code of our algorithm in presented in Algorithm 6. In this algorithm, the graph G_τ denotes a graph on the set of customers, C, such that two vertices in the graph are adjacent if their distance is at most τ.

Algorithm 6 $(3+\varepsilon)$-Approximation MPC k-Supplier

1: Let $T_i = \text{GMM}(C_i)$ and $T = \bigcup_{i=1}^{m} T_i$.
2: Compute $Q = \text{GMM}(T)$.
3: Let $r = r(C, Q) + r(Q, S)$.
4: Define $\tau_i = (r/9) \cdot (1+\varepsilon)^i$, for $i = 0, 1, \ldots, t = \log_{1+\varepsilon}(9)$.
5: Let $M_t = Q$, and let M_i be a $(k+1)$-bounded MIS in $G_{2\tau_i}$, for $0 \leq i < t$.
6: Find the smallest index j such that $|M_j| \leq k$ and $r(M_j, S) \leq \tau_j$.
7: If no such j exists, then set $j = 0$.
8: Return a subset of S realizing $r(M_j, S) \leq \tau_j$.

Theorem 18. *Algorithm 6 computes a $(3+\varepsilon)$-approximation to the k-supplier problem in any metric space in $O(\log \frac{1}{\varepsilon})$ MPC rounds using $\tilde{O}(mk)$ communication and $\tilde{O}(n/m + mk)$ memory per machine, where m is the number of machines.*

Proof. Let $S^* = \{s_1, \ldots, s_k\}$ be an optimal solution to the k-supplier problem with radius $r^* = r(C, S^*)$. We first show that the value r computed in line 3 of the algorithm is a 9-approximation to r^*. Let P_i be the set of customers closest to s_i, for $1 \leq i \leq k$, and let p_i be an arbitrary point in P_i. Then the set $\{p_1, \ldots, p_k\}$ provides a valid instance to the k-center of C with radius at most $2r^*$, since each p_i covers P_i with radius at most $2r^*$. Therefore, if $\hat{r}$ denotes the radius of an optimal k-center solution on the set C, we have $\hat{r} \leq 2r^*$. On the other hand, as we proved in Theorem 17, Q is a 4 approximation to the k-center of C, i.e., $r(C, Q) \leq 4\hat{r}$. Therefore,

$$r(C, Q) + r(Q, S) \leq 4\hat{r} + r^* \leq 8r^* + r^* = 9r^*.$$

Note that $r(C, Q)$ and $r(Q, S)$ can be computed in two rounds as follows: The central machine sends Q to all machines. Each machine i then computes $r(C_i, Q)$ and $r(Q, S_i)$ locally, and send the two values to the central machine. The central machine can then find $r(C, Q) = \max_i r(C_i, Q)$ and $r(Q, S) = \max_i r(Q, S_i)$. Hence. we can obtain in constant round a value r such that $r/9 \leq r^* \leq r$.

Now, let j be an index such that two conditions $|M_j| \leq k$ and $r(M_j, S) \leq \tau_j$ are satisfied, but either $|M_{j-1}| \leq k$ or $r(M_{j-1}, S) \leq \tau_j$ is violated. Clearly, M_j is a solution to k-center with radius $2\tau_j$, since any point in $V \setminus M_j$ has distance at most τ_j from M_j. Moreover, every point in M_j can be covered by a point from S which is at most τ_j apart. Therefore, there is a subset of suppliers of size at most k covering customers with radius at most $2\tau_j + \tau_j = 3\tau_j$. Note that M_t satisfies both conditions $|M_t| \leq k$ and $r(M_t, S) \leq r$, because M_t covers C with radius at most r, and $r(M_t, C) + r(M_t, S) = r$ by algorithm.

We consider the following two cases. The first case is when the inequality $M_{j-1} \leq k$ is violated. Here, there is an independent set of size at least $k+1$ in $C_{2\tau_{j-1}}$. Thus, $r^* \geq \tau_{j-1}$, since no supplier can cover two elements of such independent set. Based on the two satisfied inequalities for index j, we can find k suppliers covering customers with radius at most $3\tau_j \leq 3r^*(1+\varepsilon)$.

In the second case, when $r(M_{j-1}, S) > \tau_{j-1}$, we necessarily have $r^* > \tau_{j-1}$, since there is a customer which cannot be covered by any supplier with radius less than τ_{j-1}. With a similar argument, we have a solution covering customers with radius $3 \cdot \tau_j \leq 3r^* \cdot (1+\varepsilon)$.

Note that if such an index j exists, we can find it via a binary search in $O(\log(\frac{1}{\varepsilon})$ rounds. Otherwise, M_0 covers all customers with radius $3r/9$. Since r is a 9 approximation to r^*, $r/3$ is a 3-approximation to r^*, which completes the proof. □

7 CONCLUSION

In this paper, we presented almost optimal MPC algorithms for two classic problems of k-center clustering and k-diversity maximization. Our algorithms are based on a novel technique for computing a so-called k-bounded maximal independent set in a threshold graph, using only a constant number of MPC rounds. Our algorithm for the k-bounded MIS is interesting on its own, as it can be used for designing MPC algorithms for several other problems. For instance, we have been able to use the k-bounded MIS successfully to obtain an almost optimal approximation algorithm for the k-supplier problem, and also, a constant-factor approximation to the minimum dominating set in graphs with bounded neighborhood independence, both in constant number of MPC rounds. Our massively parallel algorithm for approximating vertex degrees in a threshold graph seems interesting as well, as a primitive for designing algorithm for other graph-related problems in the MPC model.

REFERENCES

[1] S. Abbasi Zadeh, M. Ghadiri, V. Mirrokni, and M. Zadimoghaddam. Scalable feature selection via distributed diversity maximization. In *Proc. 32nd AAAI Conference on Artificial Intelligence*, 2017.
[2] S. Assadi, X. Sun, and O. Weinstein. Massively parallel algorithms for finding well-connected components in sparse graphs. In *Proc. 38th ACM Sympos. Principles of Distributed Computing*, pages 461–470, 2019.
[3] M. Bateni, A. Bhaskara, S. Lattanzi, and V. Mirrokni. Distributed balanced clustering via mapping coresets. In *Proc. 27th Annu. Conf. Neural Info. Proc. Systems*, pages 2591–2599, 2014.
[4] M. Bateni, H. Esfandiari, M. Fischer, and V. Mirrokni. Extreme k-center clustering. In *Proc. 35th AAAI Conference on Artificial Intelligence*, pages 3941–3949, 2021.
[5] S. Behnezhad, S. Brandt, M. Derakhshan, M. Fischer, M. Hajiaghayi, R. M. Karp, and J. Uitto. Massively parallel computation of matching and mis in sparse graphs.

In *Proc. 38th ACM Sympos. Principles of Distributed Computing*, pages 481–490, 2019.

[6] M. Ceccarello, A. Pietracaprina, and G. Pucci. Solving k-center clustering (with outliers) in MapReduce and streaming, almost as accurately as sequentially. *Proc. VLDB Endow.*, 12(7):766âĂŞ778, 2019.

[7] Y.-J. Chang, M. Fischer, M. Ghaffari, J. Uitto, and Y. Zheng. The complexity of $(\Delta + 1)$ coloring in congested clique, massively parallel computation, and centralized local computation. In *Proc. 38th ACM Sympos. Principles of Distributed Computing*, pages 471–480, 2019.

[8] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, pages 642–651, 2001.

[9] V. Cohen-Addad, F. Mallmann-Trenn, and D. Saulpic. A massively parallel modularity-maximizing algorithm with provable guarantees. In *Proc. 41st ACM Sympos. Principles of Distributed Computing*. 2022.

[10] B. Doerr. Probabilistic tools for the analysis of randomized optimization heuristics. In *Theory of evolutionary computation*, pages 1–87. Springer, 2020.

[11] A. Ene, S. Im, and B. Moseley. Fast clustering using MapReduce. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 681–689, 2011.

[12] A. Epasto, M. Mahdian, V. Mirrokni, and P. Zhong. Massively parallel and dynamic algorithms for minimum size clustering. In *Proc. 2022 ACM-SIAM Sympos. Discrete Algorithms*, pages 1613–1660, 2022.

[13] A. Epasto, V. Mirrokni, and M. Zadimoghaddam. Scalable diversity maximization via small-size composable core-sets (brief announcement). In *Proc. 31st ACM Sympos. Parallel Algorithms Architect.*, pages 41–42, 2019.

[14] M. Ghaffari, C. Grunau, and C. Jin. Improved mpc algorithms for mis, matching, and coloring on trees and beyond. *arXiv preprint arXiv:2002.09610*, 2020.

[15] M. Ghaffari, C. Jin, and D. Nilis. A massively parallel algorithm for minimum weight vertex cover. In *Proc. 32nd ACM Sympos. Parallel Algorithms Architect.*, pages 259–268, 2020.

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

[17] D. S. Hochbaum and D. B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of operations research*, 10(2):180–184, 1985.

[18] D. S. Hochbaum and D. B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. 33(3):533–550, 1986.

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

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

[21] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986.

[22] G. Malkomes, M. J. Kusner, W. Chen, K. Q. Weinberger, and B. Moseley. Fast distributed k-center clustering with outliers on massive data. *Advances in Neural Information Processing Systems*, 28, 2015.

[23] V. Mirrokni and M. Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *Proc. 47th Annu. ACM Sympos. Theory Comput.*, pages 153–162, 2015.

[24] S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42(2):299–310, 1994.

[25] G. Yaroslavtsev and A. Vadapalli. Massively parallel algorithms and hardness for single-linkage clustering under l_p distances. In *International Conference on Machine Learning*, pages 5600–5609. PMLR, 2018.