A MapReduce Algorithm for Metric Anonymity Problems

Sepideh Aghamolaei* Mohammad Ghodsi† Hamid Miri

Abstract

We focus on two metric clusterings namely r-gather and (r, ϵ)-gather. The objective of r-gather is to minimize the radius of clustering, such that each cluster has at least r points. (r, ϵ)-gather is a version of r-gather clustering with the extra condition that at most $n\epsilon$ points can be left unclustered (outliers).

MapReduce is a model used for processing big data. In each round, it distributes data to multiple servers, then simultaneously processes each server's data.

We prove a lower bound 2 on the approximation factor of metric r-gather in MapReduce model, even if an optimal algorithm for r-gather exists. Then, we give a $(3+\epsilon)$-approximation algorithm for r-gather in MapReduce which runs in $O(\frac{1}{\epsilon})$ rounds. Also, for (r, ϵ)-gather, we give a $(7 + \epsilon)$-approximation algorithm which runs in $O(\frac{1}{\epsilon})$ MapReduce rounds.

1 Introduction

Privacy is a fundamental concern in publishing data [7] or providing input to untrusted programs [19]. One of the privacy-preserving methods is **k-anonymity** [20, 9], where given a set of records in a table, the goal is to change some of the attributes from some of the records such that for any record there exist at least $k-1$ other equal records and the maximum distance between the modified and the original records is minimized. Constant factor approximation algorithms for $k = 2, 3$ [1] and $O(\log k)$-approximation exists for general k [18].

When records are points in a metric space, the problem is called **r-gather** [2]. Aggarwal et al [2] introduced a version of k-anonymity where records are points in a metric space. They proved metric r-gather has no 2-approximation algorithm assuming $P \neq NP$, and they gave a 2-approximation algorithm for this problem. Ene et al [12] proved the lower bound 1.8 for Euclidean r-gather.

If we allow $n\epsilon$ points to be left unclustered (outliers), the problem is called **(r, ϵ)-gather** and has a 4-approximation algorithm [2].

r-gather is formally defined in Definition 1.1. In the rest of the paper, we denote $\{1, 2, \cdots, n\}$ with $[n]$.

*Department of Computer Engineering, Sharif University of Technology, aghamolaei@ce.sharif.edu

†Department of Computer Engineering, Sharif University of Technology, ghodsi@sharif.edu

Definition 1.1 *r-Gather problem clusters n points $p_1, \cdots, p_n$ in a metric space into a set of clusters centered at $C \subset \{p_1, \cdots, p_n\}$, such that each cluster has at least r points. The objective is to minimize the maximum radius among the clusters (R).*

- $\forall i \in [n], \exists c \in C, d(p_i, c) \leq R$,
- $\forall c \in C, |\{p_i : i \in [n], d(p_i, c) \leq R\}| \geq r$.

Some closely related problems are k-center with lower bound (Euclidean [12]) or both lower and upper bounds [11, 4] on the number of points in each cluster. In these problems, the number of clusters (k) and the minimum number of points in each cluster (r) are given and the goal is to minimize the radius (R). General frameworks for using sequential algorithms in MapReduce via coresets [5, 14] do not directly work for r-gather, since the points of a cluster might be distributed among many servers. Aghamolaei and Ghodsi [3] proved that for k-center the black box algorithm of these algorithms can be replaced by a covering algorithm (coreset). Running our algorithms to compute the coreset, and then applying of these general frameworks with a sequential algorithm at the last step extends our results to $\theta(\frac{n}{k})$-balanced k-center and k-center with lower bound on the number of points in each cluster.

In this paper we give constant factor approximation algorithms for r-gather and (r, ϵ)-gather in MapReduce model. We summarize the results in Table 1.

Table 1: Summary of results. '+' denotes the combination of algorithms. We assume $k \leq n/p$, for p servers.

Conditions	**Rounds**	**App.**	**Refs**
r-gather:			
lower bound	$\geq k$	2	Thm. 1
-	2	16	Alg. 5
-	$O(\frac{1}{\epsilon})$	$3+\epsilon$	Alg. 4
(r, ϵ)-gather	$O(\frac{1}{\epsilon})$	$7+\epsilon$	Alg. 7
k-center:			
balanced	$O(1)$	160	[11]+[5]
$\theta(\frac{n}{k})$-balanced	$O(1)$	$96+\epsilon$	Alg. 4 + [5]
lower-bounded	$O(1)$	$6+\epsilon$	Alg. 4+ [14]
lower bound	$\geq k$	2	Thm. 1
with outliers:			
-	$O(\frac{1}{\epsilon})$	$7+\epsilon$	Alg. 7 ($r = 1$)
-	2	13	[16]
doubling dim.	2	$2+\epsilon$	[8]

2 Preliminaries

2.1 MapReduce

In the MapReduce model [10], data is distributed among a set of independent servers. A MapReduce algorithm runs in several rounds. In each round, servers process their data locally and simultaneously. At the end of each round, they communicate with each other by sending a subset of their data to other servers. Two main theoretical models for MapReduce are MapReduce Class (MRC) [15] and Massively Parallel Communication (MPC) [6].

In the MPC model, there are p servers, each with memory $m = O(\frac{N}{p^{1-\epsilon}})$ bits of data, where N is the input size and $\epsilon \in [0, 1]$ is a parameter of the model. The complexity of MPC algorithms is measured in the number of rounds and the communications between servers.

2.2 k-Center

Given a set of points, k-Center chooses k points as centers minimizing the maximum distance from each point to its nearest center. We use Greedy Min-Max (GMM) [13] for computing a 2-approximation of metric k-center (algorithm 1), and a modification of it that adds centers until all points are covered using radius R (algorithm 2).

Algorithm 1 GMM-k

```
Input: a set of points $S$, the number of clusters $k$
Output: a set of centers $T$, the cluster sizes $\{n_c\}_{c \in T}$
1: $T$ = an arbitrary point $p \in S$
2: for $i = 2, \cdots, k$ do
3:     find a point $p \in S \setminus T$ maximizing $\min_{t \in T} d(p,t)$
4:     $T \leftarrow T \cup \{p\}$
5: end for
6: assign each point to its nearest center from set $T$
7: $n_c$ = the number of points assigned to $c$
8: return $T, \{n_c\}_{c \in T}$
```

Algorithm 2 GMM-R

```
Input: a set of points $S$, the radius of clustering $R$
Output: a set of centers $T$, the cluster sizes $\{n_c\}_{c \in T}$
1: $dist = \infty, T$ = an arbitrary point $p \in S$
2: while $dist \geq R$ do
3:     find a point $p \in S \setminus T$ maximizing $\min_{t \in T} d(p,t)$
4:     $T \leftarrow T \cup \{p\}$
5:     $dist \leftarrow \min_{t \in T} d(p,t)$
6: end while
7: assign each point to its nearest center from set $T$
8: $n_c$ = the number of points assigned to $c$
9: return $T, \{n_c\}_{c \in T}$
```

A 4-approximation k-center algorithm in MapReduce [17] is shown in algorithm 3. We slightly modify the algorithm to keep the order of centers as they are found.

Algorithm 3 Preprocess

```
Input: points in $P_i$ for $i \in [p]$
Output: $k$ centers
   ▷ parallel in all servers
1: $T_i$ = the centers returned by GMM-$k$ $(P_i)$ for $i \in [p]$
   ▷ sequential in one server
2: $T = (c_1, \cdots, c_k)$ = the centers in the order found
   by GMM-$k$ $(\cup_{i=1}^{p} T_i)$
3: return $T$
```

3 Lower Bound

In this section we give a lower bound for approximation factor of r-gather clustering in the MapReduce model.

Theorem 1 *There are no α-approximation r-gather clusterings in MPC for $\alpha < 2, k > \log n$, where k is the number of clusters.*

Proof. Let G be a graph whose vertices are grouped into $2k$ subsets called $A_i \in A$ for $i \in [k]$ and $B_i \in B$ for $i \in [k]$. Each subset in A has $l \in [\frac{r}{2}, r)$ points and each subset $B_i \in B$ has at least $\frac{r}{2}$ points. Points inside each $A_i \in A$ have distance 1 from each other and points in $B_i \in B$ have distance 2. For each A_i and B_i, for $i \in [k]$, there is an optimal center in set A_i called o_i that has distance 1 from all points of B_i. All other edges have weight 2. So the radius of r-gather is 1 if and only if A_i and B_i, for $i \in [k]$, become clusters and o_i is selected as the center, as depicted in Figure 3. Also, assume all subsets $A_i \in A$ are in the same server and each of subsets $B_i \in B$ is in a different server excluding the one containing subsets of A.

Let ALG be an r-gather algorithm. Since for each $i \in [k]$ points of A_i and B_i are not in the same server, ALG cannot find o_i knowing only A. To find $o_i \in A_i$, ALG needs all points of A_i and at least one point of B_i to be in the same server. Since ALG cannot differentiate between the points of $B_j \in B$ without knowledge of A, in the worst case, ALG has to send all points of A_i to all servers to find o_i, which takes one MPC round. This will only give one point o_i, if $|A_i| = \theta(m)$. So finding the k optimal points requires at least k rounds. □

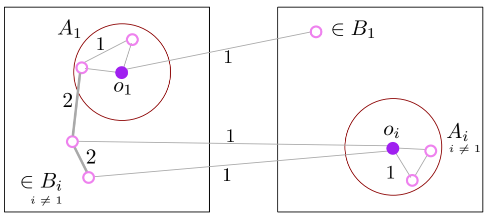

Figure 1: Lower-bound for r-gather (Theorem 1)

4 r-Gather in MapReduce

We propose an algorithm (Algorithm 4) for r-gather. It first finds an lower bound on the optimal radius R using **Bounds(P)** subroutine, which computes a 16-approximation algorithm for r-gather (algorithm 5).

Then it multiplies the lower bound by factor $1+\epsilon$ and tests the resulting radius with an r-gather decider algorithm **Decider(R,P)** (algorithm 6) that finds an r-gather of radius $3R$ if an r-gather or radius R exists, otherwise it returns *FAIL*.

Algorithm 4 r-Gather 3-approximation

```
Input: $P$ a set of points distributed in servers
Output: clusters
1: $UB = Bounds(P), R = \frac{UB}{16}$
2: while $R \le UB$ do
3:     if $Decider(R,P)$ returns $FAIL$ then
4:         $R \leftarrow (1+\epsilon)R$
5:     else
6:         return the output of $Decider(R,P)$
7:     end if
8: end while
```

4.1 r-Gather Bounds

Algorithm 5 finds a constant factor approximation for r-gather. Assume the value of k is selected such that $pk^2 \le m$ and k is greater than the number of clusters in the optimal r-gather. **Preprocess**() is algorithm 3.

Algorithm 5 Bounds

```
Input: points in $P_i$ for $i \in [p]$
Output: r-gather clusters
1: $T = \textbf{Preprocess}(\{P_i\}_{i\in[p]})$
2: send $T$ to all servers
   ▷ parallel in all servers:
3: $C = \emptyset, j = 1$
4: for $j \in [k]$ do
5:     $C \leftarrow C \cup \{c_j\}$
6:     assign points to their nearest center in $C$
7:     $n(i,c,j) =$ the sizes of clusters centered at $c \in C$
8: end for
9: send $n(i,c,j)$ to one server for $i \in [p], j = [k], c \in T$
   ▷ sequentially in one server:
10: find the minimum $k' \in [k]$ such that exists a center
    $c \in T$, $\sum_{i=1}^{p} n(i,c,k') < r$
11: return $\{c_1, \cdots, c_{k'-1}\}$
```

Lemma 2 *In algorithm 5, each cluster has at least r points.*

Proof. According to line 10 of the algorithm, k' is the first step that the number of points assigned to a center becomes less than r. So, in step $k'-1$ each center in $\{c_1, \cdots, c_{k'-1}\}$ has at least r points. □

Theorem 3 *Algorithm 5 is a 16-approximation for r-gather.*

Proof. Based on Lemma 2, each cluster has at least r points.

Consider a point s that was reassigned at step k'. Let c_i be the center of the cluster that contained s at step $k'-1$. Since points are assigned to their nearest centers, we have $d(c_j,s) \le d(c_i,s)$, and using triangle inequality $d(c_i,c_j) \le d(c_i,s) + d(c_j,s)$. So, $d(c_i,c_j) \le 2d(c_i,s)$.

Each point $c_i \in C$ covers all points of the optimal cluster containing c_i, with radius $2R^*$, where R^* is the radius of the optimal r-gather of S, i.e. $d(c_i,s) \le 2R^*$. Putting the two previous inequalities together gives the bound $d(c_i,c_j) \le 4R^*$.

GMM-k adds the farthest point as a center each time, so the cluster containing c_j has the maximum radius R' among the clusters. Let c_k be the nearest center to c_j, i.e. $d(c_k,c_j) \le d(c_i,c_j)$ for all $i \ne k$. Since c_j is assigned to its nearest center, $R' = d(c_k,c_j)$. So, we have $R' \le 4R^*$.

Algorithm 3 is a 4-approximation for k-Center, so $R \le 4R'$, where R is the radius of k-Center from first round. The last two inequalities yield $R \le 4R' \le 16R^*$. □

In algorithm 5, for each server we send k^2 points to one server, so we have $pk^2 \le m$ points in total. Since we had condition $pk^2 \le m$, algorithm 5 follows the constraints of MPC model.

4.2 r-Gather Decider

r-Gather decider takes R as input and gives an r-gather clustering with radius $3R$ if an r-gather with radius R exists, otherwise it returns *FAIL*. Assume we have the order of centers added by GMM-k algorithm used in r-gather bounds algorithm.

4.2.1 Cluster Re-assignments via Max. Flow

Two sets of clusters D and S with the number of points assigned to them, and $\mathcal{R}$ an upper bound for the radius of clustering are given. For each $p \in D \cup S$ we denote the number of points assigned to p with n_p. The goal is to re-assign the points of set S to centers of clusters in D using a flow network.

Definition 4.1 *Consider the graph $G = (V,E)$ with $V = \{s,t\} \cup D \cup S$ and $E = E_s \cup E_m \cup E_t$, where E_s, E_m, E_t are defined as follows:*

- $E_s = \{(s,v,r-n_v) \mid v \in D\}$
- $E_m = \{(u,v,n_v) \mid u \in D, v \in S, d(u,v) \le \mathcal{R}\}$
- $E_t = \{(u,t,n_u) \mid u \in S\}$

and where (u, v, c) denotes an edge from vertex u to vertex v with capacity c.

Let $MaxFlow(D, S, \{n_c\}_{c \in D \cup S}, \mathcal{R})$ be the function that if the max. flow of G in the flow network of Definition 4.1 is less than $\sum_{(u,v,c) \in E_s} c$, returns ($FAIL$, $\emptyset$) and otherwise returns ($SUCCESS$, $\{(u, v, c) \in E_m\}$).

Figure 2 shows the original clustering, the flow network and the clusters after re-assignment.

Algorithm 6 Decider

```
Input: points in $P_i$ for $i \in [p]$, radius $R$
Output: $r$-gather clustering or $FAIL$
    ▷ parallel in all servers
 1: $T_i, \{n_c\}_{c \in T_i}$ = GMM-$R$ $(P_i, R)$ for $i \in [p]$
 2: send $T_i$ for $i \in [p]$ to one server
    ▷ sequentially in one server:
 3: $D$ = centers with $\geq r$ points assigned to it
 4: $S$ = centers with $< r$ points assigned to it
 5: for $c_s \in S$ do
 6:     $N(c_s, R)$ = centers in radius of $R$ of $c_s$
 7:     if all centers $c \in N(c_s, 2R)$ are in $S$ then
 8:         $D \leftarrow D \cup \{c_s\}$
 9:     end if
10: end for
11: $(flag, E) = MaxFlow(D, S, \{n_c\}_{c \in D \cup S}, 2R)$
12: if $flag = FAIL$ then
13:     return $FAIL$
14: end if
15: send $E$ to all servers
16: according to $E$, assign points to centers in $D$
    ▷ parallel in all servers
17: return centers in set $D$ and their assigned points
```

4.2.2 Analysis

Lemma 4 *In algorithm 6, each cluster has at least r points.*

Proof. In the flow network of definition 4.1, there is an edge with capacity $r - n_{c_d}$ from node s to every center $c_d \in D$. From the definition of $MaxFlow(.,.,.,.)$ we know the max. flow is $r|D| - \sum_{c \in D} n_c$, which means c_d is assigned $r - n_{c_d}$ new points. Since c_d had n_{c_d} existing assigned points, in total it has r assigned points. □

Theorem 5 *Algorithm 6 is 3-approximation.*

Proof. By lemma 4 in the solution of the algorithm, each cluster has at least r points.

According to definition 4.1 each center $c_d \in D$ has edges to centers $c_s \in S$ within radius of $2R$. So c_s only can gets new points from c_s at distance at most $2R$. From line 1 of algorithm 6 for each point p assigned to c_s we have $d(c_s, p) \leq R$. By triangle inequality we have $d(c_d, p) \leq d(c_d, c_s) + d(c_s, p) \leq 3R$.

From line 8 of the algorithm we know for each $c_s \in S$ there exists a center $c_d \in D$ within distance of $2R$. By the triangle inequality, all assigned points of c_s can be covered by c_d with radius at most $3R$. So all of the input points can be covered with *dense* centers with radius at most $3R$. □

In algorithm 6, for each server we send k points to one server, so we have pk points in total. Since we had the condition $pk^2 \leq m$, algorithm 6 runs in MPC model.

4.3 Analysis

Theorem 6 *Algorithm 4 takes $O(\frac{1}{\epsilon})$ rounds in MapReduce and gives a $(3 + \epsilon)$-approximation for r-gather.*

Proof. According to algorithm 4, r-gather 3-approximation algorithm runs r-gather bounds once at line 1. By theorem 3, we know r-gather bounds takes 2 rounds in MapReduce model. In algorithm 4 line 2, each while iteration runs algorithm 6 one time. By theorem 5, we know r-gather decision takes 2 rounds in the MapReduce model. In line 2 while statement iterates at most $O(\frac{1}{\epsilon})$ so total algorithm takes $O(\frac{1}{\epsilon})$ rounds in the MapReduce model.

Let R be the greatest radius which r-gather decision algorithm returns $FAIL$. So we have $R \leq R^* \leq R(1 + \epsilon)$ where R^* is the optimal radius of clustering. By theorem 5 we know r-gather decision with $R(1 + \epsilon)$ as input returns a r-gather clustering with radius $3R(1+\epsilon)$. So we have $3R(1+\epsilon) \leq 3R^*(1+\epsilon)$ and the approximation factor is $3 + \epsilon$.

By theorems 3 and 5 we know algorithms 5 and 6 runs in MPC model. So algorithm 4 runs in MPC model too. □

5 (r, ϵ)-Gather in MapReduce

In this section, we provide an algorithm which takes the optimal radius R as input and finds a (r, ϵ)-gather clustering of radius $7R$ if one exists, or returns $FAIL$ otherwise. Also, we give an algorithm to find the lower bound and upper bound for the radius of clustering (algorithm 8 denoted by $(r, \epsilon) - GLB(.)$), which we then use with a 7-approximation algorithm as its decision subroutine to solve the problem.

Algorithm 7 (r, ϵ)-Gather

```
Input: $P$ a set of points distributed in servers
Output: clusters
1: $R = (r, \epsilon) - GLB(P)$
2: while algorithm 9 $(R, S)$ returns $FAIL$ do
3:     $R \leftarrow (1 + \epsilon)R$
4: end while
5: return the output of algorithm 9
```

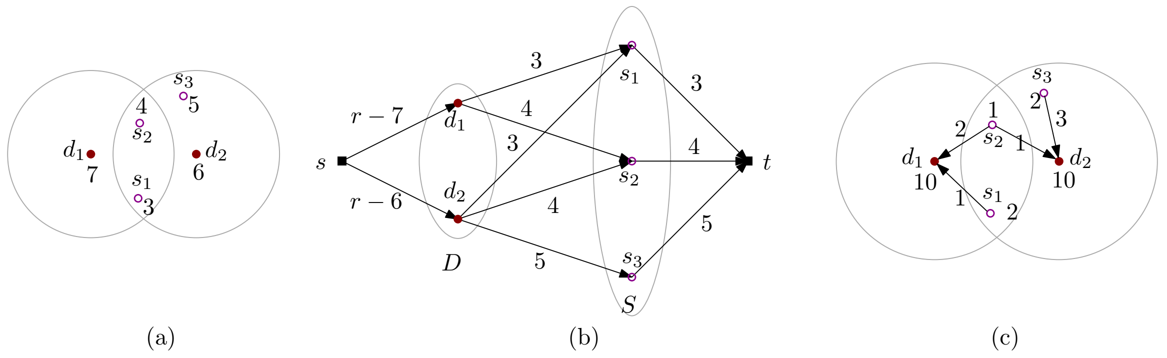

Figure 2: (a) a clustering, (b) the network flow of (a), and (c) a reassignment of points to centers for $r = 10$

5.1 (r, ϵ)-Gather Lower-Bound ((r, ϵ)-GLB)

The inputs are a set of points distributed among the servers and a parameter ϵ which means at most $n\epsilon$ points can be left unclustered. Assume we select k such that $pk^2 \leq m$, and k is greater than the number of clusters in the optimal (r, ϵ)-gather. In the first round of algorithm 9, we use algorithm 3.

Algorithm 8 (r, ϵ)-GLB

```
Input: points in $P_i$ for $i \in [p]$
Output: lower bound for $(r, \epsilon)$-gather clustering
 1: $T$ = Preprocess with inputs $P_i$ for $i \in [p]$
 2: send centers in $T$ to all servers
    ▷ parallel in all servers
 3: $C = \emptyset$
 4: for $j \in [k]$ do
 5:     $C \leftarrow C \cup \{c_j\}$
 6:     each point assigns to the nearest center $c \in C$
 7:     $R_c$ = radius of the cluster centered at $c$
 8:     $n(i, c, R, j)$ = the number of input points within
    distance $R$ of $c$ in server $i$
 9: end for
10: collect set $C$ with $\{n(i, c, R_c, j)\}$ and $\{n(i, c, \frac{R_c}{2}, j)\}$
    for $i \in [p], j \in [k], c \in C$
    ▷ sequential in one server
11: $C_T = \emptyset$, $L = \emptyset$
12: for $j \in [k]$ do
13:     $n(c, R, j) = \sum_{i=1}^{p} n(i, c, R, j)$ for $c \in C_T$
14:     $C_T \leftarrow C_T \cup \{c_j\}$
15:     if $n(c_j, R_c, j) < r$ then
16:         $L \leftarrow L \cup \{c_j\}$
17:     end if
18:     if $\sum_{c \in L} n(c, \frac{R_c}{2}, j) > n\epsilon$ then
19:         break
20:     end if
21: end for
22: $R = min_{c \in C_T} R_c$
23: return $\frac{R}{4}$
```

Lemma 7 *The value returned by algorithm 8 is a lower bound within a constant factor of the optimal radius of (r, ϵ)-gather.*

Proof. In line 18 of algorithm 8 the algorithm terminates if the points assigned to the centers in L become more than $n\epsilon$. Let outlier candidates (set OC) be the set of centers in L and their assigned points. Since $|OC| > n\epsilon$, there is a point $p \in OC$ that must be clustered. Suppose the nearest center to p is a center called c, then we have $d(c, p) \leq \frac{R_c}{2}$. Since $c \in L$, we know c does not have r points within radius of R_c. For each point q outside the radius R_c around c, by triangle inequality, we have $d(c, q) \leq d(c, p) + d(p, q)$. So we have $R_c \leq \frac{R_c}{2} + d(p, q)$ and $\frac{R_c}{2} \leq d(p, q)$. So p does not have more than r points within radius $\frac{R_c}{2}$. We know in any (r, ϵ)-gather clustering with radius R' each point can cover all points of its cluster with radius $2R'$. Since p must be clustered in the optimal (r, ϵ)-gather clustering, the cluster contains p cannot have radius less than $\frac{R_c}{4}$. So we have $R^* \geq \frac{R_c}{4}$ where R^* is the optimal radius. □

5.2 (r, ϵ)-Gather Decision

The inputs are R the radius of (r, ϵ)-gather clustering and ϵ a parameter which indicates at most $n\epsilon$ points can be left unclustered. (r, ϵ)-Gather decision algorithm gives an (r, ϵ)-gather with radius $7R$ if there exists a (r, ϵ)-gather clustering with radius R and returns *FAIL* otherwise.

Lemma 8 *All the points that are clustered in the optimal (r, ϵ)-gather will also be clustered by (r, ϵ)-gather decision algorithm.*

Proof. Based on line 6 of algorithm 9, centers that do not have r points within radius of $3R$ of themselves are outliers. Let c be a center which has been removed as an outlier, and p be a point assigned to c, so p has been removed as an outlier, too. Let q be any point such that $d(c, q) \geq 3R$. Because of triangle inequality, we have $d(c, q) \leq d(c, p) + d(p, q)$. Line 1 of the algorithm

Algorithm 9 (r,ϵ)-Gather Decision

```
Input: points in $P_i$ for $i \in [p]$, radius $R$
Output: $(r,\epsilon)$-Gather clustering or $FAIL$
    ▷ parallel in all servers
 1: $T_i, \{n_c\}_{c\in T_i}$ = GMM-$R$ $(P_i, R)$
 2: assign points in $P_i$ to their nearest center in $T_i$
 3: collect sets $T_i$ with $\{n_c\}_{c\in T_i}$
    ▷ sequentially in one server:
 4: for $c \in \cup_{i=1}^{p} T_i$ do
 5:     $N(c,R)$ =set of centers within radius of $R$ of $c$
 6:     remove $c$ and its assigned points as outliers if $\sum_{q\in N(c,3R)} n_q < r$
 7: end for
 8: $D = \emptyset$, $S = \emptyset$, $U = \cup_{i=1}^{p} C_i$
 9: for $c_u \in U$ do
10:     $N_U(c,R) = U \cap N(c,R)$
11:     $n(N_U(c,R)) = \sum_{t\in N_U(c,R)} n_t$
12:     if $n(N_U(c_u,3R)) \geq r$ then
13:         $U \leftarrow U \setminus N_U(c_u,3R)$
14:         $D \leftarrow D \cup \{c_u\}$
15:         $S \leftarrow S \cup (N_U(c_u,3R) \setminus \{c_u\})$
16:     end if
17: end for
18: $(flag, E) = MaxFlow(D, S, \{n_c\}_{c\in D\cup S}, 3R)$
19: if $flag = FAIL$ then
20:     return $FAIL$
21: end if
22: send $E$ to all servers
    ▷ parallel in all servers
23: according to $E$ assign points of assigned to $S$ to centers in $D$
24: assign all of points assigned to centers in $U$ to their nearest center in $D$
25: return centers in set $D$ and their assigned points
```

guarantees $d(c,q) \leq R$, so $3R \leq R + d(p,q)$ and $d(p,q) \geq 2R$. This means in any (r,ϵ)-gather, points p and q cannot belong to the same cluster. Since c does not have r points within distance $3R$ of itself, no point within distance R of c can be clustered in the optimal (r,ϵ)-gather. □

Theorem 9 *The approximation factor of (r,ϵ)-gather is 7.*

Proof. In the flow network of line 18, there are only edges between *dense* centers $c_d \in D$ and *sparse* centers $c_s \in S$ such that $d(c_d, c_s) \leq 3R$. So c_d can only get new points from points $c_s \in S$ with distance at most $3R$ from c_d. Let p be a point assigned to c_s, so $d(c_s,p) \leq R$. By triangle inequality we have $d(c_d,p) \leq d(c_d,c_s) + d(c_s,p) \leq 4R$. So all centers $c_s \in S$ and their assigned points can be clustered with radius $4R$ with c_d as center.

For each unclustered center $c_u \in U$, since c_u was not removed at line 6, we know c_u has at least r points within distance $3R$ of itself. Since c_u remained an *unclustered* center, from line 12 we know there are less than r *unclustered* points within distance $3R$ of c_u. So there exists at least one *sparse* center $c_s \in S$ within distance $3R$ of c_u. From line 15 we know there exists a *dense* center in distance of at most $3R$ of c_s. By triangle inequality we have $d(c_d,c_u) \leq d(c_d,c_s) + d(c_s,c_u) \leq 6R$. Let p be a point assigned to c_u, so using triangle inequality we have $d(c_d,p) \leq d(c_d,c_u) + d(c_u,p) \leq 7R$. □

We send at most k points per server to take their union in one server, which is at most pk points in total. Since we had the assumption $pk^2 \leq m$, algorithm 9 follows the memory constraints of the MPC model.

5.3 Analysis

Theorem 10 *Algorithm 7 takes $O(\frac{1}{\epsilon})$ rounds in MapReduce and gives (r,ϵ)-gather clustering with approximation factor $7+\epsilon$.*

Proof. According to algorithm 7, (r,ϵ)-gather 7-approximation algorithm runs (r,ϵ)-GLB once at line 1. From algorithm 8, we know (r,ϵ)-GLB takes 2 rounds in the MapReduce model. By theorem 5, we know (r,ϵ)-gather decision takes 2 rounds in the MapReduce model. In line 2 of the algorithm, the while loop iterates at most $O(\frac{1}{\epsilon}\log\frac{OPT}{LB})$ times, for $LB = (r,\epsilon) - GLB(P)$. So, the total round complexity of the algorithm is $O(\frac{1}{\epsilon})$.

Let R be the maximum radius for which (r,ϵ)-gather decision algorithm returns *FAIL*. So we have $R \leq R^* \leq R(1+\epsilon)$ where R^* is the optimal radius of clustering. By theorem 5, we know (r,ϵ)-gather decision algorithm with $R(1+\epsilon)$ as input returns a (r,ϵ)-gather clustering with radius $7R(1+\epsilon)$. So we have $7R(1+\epsilon) \leq 7R^*(1+\epsilon)$ and the approximation factor is $7+\epsilon$. □

6 Conclusion and Open Problems

We solved two minimum radius covering problems with lower bounds on the number of members in a cluster, and gave a constant factor approximation for these problems in MapReduce. While most MapReduce capacitated clustering algorithms are based on linear programming, we used an algorithm based on maximum flow.

Improving the round complexity or the approximation factor of our algorithm remains an open problem. Also, removing the assumption that $k \leq m$ will make our algorithm more scalable.

Finding similar results for other clusterings with ℓ_p-based costs such as k-means and k-median is also interesting.

References

[1] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Approximation algorithms for k-anonymity. *Journal of Privacy Technology (JOPT)*, 2005.

[2] G. Aggarwal, R. Panigrahy, T. Feder, D. Thomas, K. Kenthapadi, S. Khuller, and A. Zhu. Achieving anonymity via clustering. *ACM Transactions on Algorithms (TALG)*, 6(3):49, 2010.

[3] S. Aghamolaei and M. Ghodsi. A composable coreset for k-center in doubling metrics. In *30th Canadian Conference on Computational Geometry (CCCG)*, 2018.

[4] S. Ahmadian and C. Swamy. Approximation algorithms for clustering problems with lower bounds and outliers. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[5] M. Bateni, A. Bhaskara, S. Lattanzi, and V. Mirrokni. Distributed balanced clustering via mapping coresets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2591–2599, 2014.

[6] P. Beame, P. Koutris, and D. Suciu. Communication steps for parallel query processing. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI symposium on Principles of database systems*, pages 273–284. ACM, 2013.

[7] C. Benjamin, M. Fung, K. Wang, R. Chen, and P. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4):1–53, 2010.

[8] M. Ceccarello, A. Pietracaprina, and G. Pucci. Solving k-center clustering (with outliers) in mapreduce and streaming, almost as accurately as sequentially. *arXiv preprint arXiv:1802.09205*, 2018.

[9] D. J. De Witt, R. Ramakrishnan, et al. Mondrian multidimensional k-anonymity. In *Proc. of the 22nd IEEE International Conference on Data Engineering (ICDE)*, 2006.

[10] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[11] H. Ding, L. Hu, L. Huang, and J. Li. Capacitated center problems with two-sided bounds and outliers. In *Workshop on Algorithms and Data Structures*, pages 325–336. Springer, 2017.

[12] A. Ene, S. Har-Peled, and B. Raichel. Fast clustering with lower bounds: No customer too far, no shop too small. *arXiv preprint arXiv:1304.7318*, 2013.

[13] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[14] P. Indyk, S. Mahabadi, M. Mahdian, and V. S. Mirrokni. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 100–108. ACM, 2014.

[15] H. Karloff, S. Suri, and S. Vassilvitskii. A model of computation for mapreduce. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 938–948. SIAM, 2010.

[16] G. Malkomes, M. J. Kusner, W. Chen, K. Q. Weinberger, and B. Moseley. Fast distributed k-center clustering with outliers on massive data. In *Advances in Neural Information Processing Systems*, pages 1063–1071, 2015.

[17] J. McClintock. Parallel algorithms for k–center clustering. In *4TH ANNUAL DOCTORAL COLLOQUIUM*, page 33, 2016.

[18] H. Park and K. Shim. Approximate algorithms for k-anonymity. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 67–78. ACM, 2007.

[19] I. Roy, S. T. Setty, A. Kilzer, V. Shmatikov, and E. Witchel. Airavat: Security and privacy for mapreduce. In *NSDI*, volume 10, pages 297–312, 2010.

[20] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, technical report, SRI International, 1998.