Improved Diversity Maximization Algorithms for Matching and Pseudoforest

Sepideh Mahabadi
Microsoft Research, Redmond, WA, USA

Shyam Narayanan[1]
Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

In this work we consider the diversity maximization problem, where given a data set X of n elements, and a parameter k, the goal is to pick a subset of X of size k maximizing a certain diversity measure. Chandra and Halldórsson [11] defined a variety of diversity measures based on pairwise distances between the points. A constant factor approximation algorithm was known for all those diversity measures except "remote-matching", where only an $O(\log k)$ approximation was known. In this work we present an $O(1)$ approximation for this remaining notion. Further, we consider these notions from the perpective of composable coresets. Indyk et al. [23] provided composable coresets with a constant factor approximation for all but "remote-pseudoforest" and "remote-matching", which again they only obtained a $O(\log k)$ approximation. Here we also close the gap up to constants and present a constant factor composable coreset algorithm for these two notions. For remote-matching, our coreset has size only $O(k)$, and for remote-pseudoforest, our coreset has size $O(k^{1+\varepsilon})$ for any $\varepsilon > 0$, for an $O(1/\varepsilon)$-approximate coreset.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Computational geometry

Keywords and phrases diversity maximization, approximation algorithms, composable coresets

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.25

Category APPROX

Related Version *Full Version*: http://arxiv.org/abs/2307.04329

1 Introduction

Diverse Subset Selection is the task of searching for a subset of the data that preserves its diversity as much as possible. This task has a large number of applications in particular while dealing with large amounts of data, including data summarization (e.g. [26, 18, 25]), search and information retrieval (e.g. [5, 2, 24, 13, 31]), and recommender systems (e.g. [33, 1, 34, 32]), among many others (e.g. [17, 29]). Here, given a ground set of n vectors X in a metric space $(\mathcal{X}, \rho)$, representing a data set of objects (e.g. using their feature vectors), and a parameter k, the goal is to choose a subset $S \subseteq X$ of this data set of size k, that maximizes a pre-specified optimization function measuring the diversity.

Many diversity measures have been introduced and used in the literature that fit different tasks. A large number of these measures are defined based on pairwise distances between the vectors in X. In particular the influential work of [11] introduced a taxonomy of pairwise-distance based diversity measures which is shown in Table 1. For example, remote-edge measures the distance of the closest points picked in the subset S, and remote-clique measures the sum of pairwise distances between the points in the subset S. Remote-pseudoforest falls between these two where it wants to ensure that the average distance of a point to its nearest

[1] Work done as an intern at Microsoft Research.

Table 1 This table includes the notions of diversity considered by [11] ($S = S_1|...|S_t$ is used to denote that $S_1 \dots S_t$ is a partition of S into t sets). We also include the best previously-known approximation factors, both in the standard (offline) and coreset setting. If not explicitly stated, the approximation factor holds for both the offline setting and the coreset setting. We note that the previously known $O(1)$-approximate remote-pseudoforest offline algorithm is randomized, whereas the rest of the previously known algorithms are deterministic.

Problem	Diversity of the point set S	Apx factor
Remote-edge	$\min_{x,y \in S} dist(x,y)$	$O(1)$
Remote-clique	$\sum_{x,y \in S} dist(x,y)$	$O(1)$
Remote-tree	$wt(MST(S))$, weight of the minimum spanning tree of S	$O(1)$
Remote-cycle	$\min_C wt(C)$ where C is a TSP tour on S	$O(1)$
Remote t-trees	$\min_{S=S_1\|\dots\|S_t} \sum_{i=1}^{t} wt(MST(S_i))$	$O(1)$
Remote t-cycles	$\min_{S=S_1\|\dots\|S_t} \sum_{i=1}^{t} wt(TSP(S_i))$	$O(1)$
Remote-star	$\min_{x \in S} \sum_{y \in S \setminus \{x\}} dist(x,y)$	$O(1)$
Remote-bipartition	$\min_B wt(B)$, where B is a bipartition (i.e., bisection) of S	$O(1)$
Remote-pseudoforest	$\sum_{x \in S} \min_{y \in S \setminus \{x\}} dist(x,y)$	$O(1)$ (Offline) $O(\log k)$ (Coreset)
Remote-matching	$\min_M wt(M)$, where M is a perfect matching of S	$O(\log k)$

neighbor is large. Remote-matching measures the diversity as the cost of minimum-weight-matching. Various other measures have also been considered: Table 1 includes each of their definitions along with the best known approximation factor for these measures known up to date. In particular, by [11] it was known that all these measures except remote-pseudoforest and remote-matching admit a constant factor approximation. More recently, [7] showed a constant factor randomized LP-based algorithm for remote-pseudoforest. They also showed the effectiveness of the remote-pseudoforest measure on real data over the other two common measures (remote-edge and remote-clique). On the lower bound side, it was known by [20] that for remote-matching, one cannot achieve an approximation factor better than 2. However, despite the fact that there has been a large body of work on diversity maximization problems [8, 3, 9, 10, 14], the following question had remained unresolved for over two decades.

▶ **Question 1.** *Is it possible to get an $O(1)$ approximation algorithm for the remaining notion of remote-matching?*

Later following a line of work on diversity maximization in big data models of computations, [23] presented algorithms producing a composable coreset for the diversity maximization problem under all the aformentioned diversity measures. An α-approximate composable coreset for a diversity objective is a mapping that given a data set X, outputs a small subset $C \subset X$ with the following composability property: given multiple data sets $X^{(1)}, \cdots, X^{(m)}$, the maximum achievable diversity over the union of the composable coresets $\bigcup_i C^{(i)}$ is within an α factor of the maximum diversity over the union of those data sets $\bigcup_i X^{(i)}$. It is shown that composable coresets naturally lead to solutions in several massive data processing models including distributed and streaming models of computations, and this has lead to recent interest in composable coresets since their introduction [28, 6, 27, 22, 4, 15]. [23] showed α-approximate compsable coresets again for all measures of diversity introduced by [11]. They presented constant factor α-approximate composable coresets for all diversity measures except remote-pseudoforest and remote-matching, where they provided only $O(\log k)$-approximations for these measures. Again the following question remained open.

▶ **Question 2.** *Is it possible to get a constant factor composable coreset for the remote-pseudoforest and remote-matching objective functions?*

In this work we answer above two questions positively and close the gap up to constants for these problems.

Our Results

In this work, we resolve a longstanding open question of [11] by providing polynomial-time $O(1)$-approximation algorithms for the remote-matching problem. We also resolve a main open question of [23] by providing polynomial-time algorithms that generate $O(1)$-approximate composable coresets for both the remote-pseudoforest and remote-matching problems. Hence, our paper establishes $O(1)$-approximate offline algorithms *and* $O(1)$-approximate composable coresets for *all* remaining diversity measures proposed in [11]. Specifically, we have the following theorems.

▶ **Theorem 3** (Remote-Matching, Offline Algorithm)**.** *Given a dataset $X = \{x_1, \ldots, x_n\}$ and an integer $k \leq \frac{n}{3}$, w.h.p. Algorithm 4 outputs an $O(1)$-approximate set for remote-matching.*

While we assume n is at least a constant factor bigger than k, this is usually a standard assumption (for instance both the remote-pseudoforest and remote-matching algorithms in [11] assume $n \geq 2k$, and the $O(1)$-approximate remote-pseudoforest algorithm in [7] assumes $n \geq 3k$). We now state our theorems regarding composable coresets.

▶ **Theorem 4** (Pseudoforest, Composable Coreset)**.** *Given a dataset $X = \{x_1, \ldots, x_n\}$ and an integer $k \leq n$, Algorithm 2 outputs an $O(1/\varepsilon)$-approximate composable coreset C for remote-pseudoforest, of size at most $O(k^{1+\varepsilon})$.*

By this, we mean that if we partitioned the dataset X into $X^{(1)}, \ldots, X^{(m)}$, and ran the algorithm separately on each piece to obtain $C^{(1)}, \ldots, C^{(m)}$, each $C^{(i)}$ will have size at most $O(k^{1+\varepsilon})$ and $\max_{Z \subset C : |Z|=k} \mathrm{PF}(Z) \geq \Omega(\varepsilon) \cdot \max_{Z \subset X : |Z|=k} \mathrm{PF}(Z)$, where $C = \bigcup_{i=1}^{m} C^{(i)}$.

▶ **Theorem 5** (Minimum-Weight Matching, Composable Coreset)**.** *Given a dataset $X = \{x_1, \ldots, x_n\}$ and an integer $k \leq n$, Algorithm 3 outputs an $O(1)$-approximate composable coreset C for remote-matching, of size at most $3k$.*

In all of our results, we obtain $O(1)$-approximation algorithms, whereas the previous best algorithms for all 3 problems was an $O(\log k)$-approximation algorithm, meaning the diversity was at most $\Omega(\frac{1}{\log k})$ times the optimum. We remark that our composable coreset in Theorem 4 has size $k^{1+\varepsilon}$, for some arbitrarily small constant ε (to obtain a constant approximation) which is possibly suboptimal. We hence ask an open question as to whether an $O(1)$-approximate composable coreset for remote-pseudoforest, of size $O(k)$, exists.

Finally, as an additional result we also prove an alternative method of obtaining an $O(1)$-approximate offline algorithm for remote-pseudoforest. Unlike [7], which uses primal-dual relaxation techniques, our techniques are much simpler and are based on ε-nets and dynamic programming. In addition, our result works for all $k \leq n$, whereas the work of [7] assumes $k \leq \frac{n}{3}$ and that k is at least a sufficiently large constant. Also, our algorithm is deterministic, unlike [7]. We defer the statement and proof to the full version of the paper on arXiv.

2 Preliminaries

2.1 Definitions and Notation

We use $\rho(x, y)$ to represent the metric distance between two points x and y. For a point x and a set S, we define $\rho(x, S) = \min_{s\in S} \rho(x, s)$. Likewise, for two sets S, T, we define $\rho(S, T) = \min_{s\in S, t\in T} \rho(S, T)$. We define the diameter of a dataset X as $\mathrm{diam}(X) = \max_{x,y\in X} \rho(x, y)$.

Costs and diversity measures

For a set of points $Y = \{y_1, \ldots, y_k\}$, we use $\mathrm{div}(Y)$, as a generic term to denote its diversity which we measure by the following cost functions.

If $k = |Y|$ is even, we define the minimum-weight matching cost $\mathrm{MWM}(Y)$ as the minimum total weight over all perfect matchings of Y. Equivalently,

$$\mathrm{MWM}(Y) := \min_{\text{permutation } \pi:[k]\to[k]} \sum_{i=1}^{k/2} \rho(y_{\pi(2i)}, y_{\pi(2i-1)}).$$

Likewise, we define the pseudoforest cost $\mathrm{PF}(Y)$, also known as the sum-of-nearest-neighbor cost, of Y as

$$\mathrm{PF}(Y) := \sum_{y\in Y} \rho(y, Y\backslash y).$$

Finally, we define the minimum spanning tree cost $\mathrm{MST}(Y)$ of Y as the minimum total weight over all spanning trees of Y. Equivalently,

$$\mathrm{MST}(Y) = \min_{G\text{: spanning tree of } [k]} \sum_{e=(i,j)\in G} \rho(y_i, y_j).$$

The pseudoforest cost and minimum spanning tree cost do not require Y to be even.

▶ **Definition 6** (Diversity maximization). *Given a dataset X, and a parameter k, the goal of the diversity maximization problem is to choose a subset $Y \subset X$ of size k with maximum diversity, where in this work we focus on* $\mathrm{div}(Y) = \mathrm{MWM}(Y)$ *and* $\mathrm{div}(Y) = \mathrm{PF}(Y)$.

We also define $\mathrm{div}_k(X)$ *to be this maximum achievable diversity, i.e.,* $\mathrm{div}_k(X) = \max_{Y\subset X, |Y|=k} \mathrm{div}(Y)$. *In particular we use* $\mathrm{MWM}_k(X)$, *or the* remote-matching *cost of X, as* $\max_{Y\subset X,|Y|=k} \mathrm{MWM}(Y)$, *and define* $\mathrm{PF}_k(X) = \max_{Y\subset X,|Y|=k} \mathrm{PF}(Y)$. *These objectives are also known as* k-matching *and* k-pseudoforest.

For a specific diversity maximization objective div_k (such as MWM_k), an *α-approximation algorithm* ($\alpha \geq 1$) for div is an algorithm that, given any dataset $X = \{x_1, \ldots, x_n\}$, outputs some dataset $Z \subset X$ of size k such that

$$\mathrm{div}(Z) \geq \frac{1}{\alpha} \cdot \mathrm{div}_k(X) = \frac{1}{\alpha} \cdot \max_{Y\subset X:|Y|=k} \mathrm{div}(Y).$$

▶ **Definition 7** (Composable coresets). *We say that an algorithm $\mathcal{A}$ that acts on a dataset X and outputs a subset $\mathcal{A}(X) \subset X$ forms an* α-approximate composable coreset *($\alpha \geq 1$) for* div *if, for any collection of datasets $X^{(1)}, \ldots, X^{(m)}$, we have*

$$\mathrm{div}_k\left(\bigcup_{i=1}^{m} \mathcal{A}(X^{(i)})\right) \geq \frac{1}{\alpha} \cdot \mathrm{div}_k\left(\bigcup_{i=1}^{m} X^{(i)}\right).$$

Algorithm 1 The GMM Algorithm.

Input: data $X = \{x_1, \ldots, x_n\}$, integer k.
y_1 is an arbitrary point in X.
Initialize $Y \leftarrow \{y_1\}$.
for $p = 2$ to k **do**
  $y_p \leftarrow \arg\max_{y \in X} \rho(y, Y)$, i.e., y_p is the furthest point in X from the current $Y = \{y_1, \ldots, y_{p-1}\}$.
  $Y \leftarrow Y \cup \{y_p\}$.
end for
Return Y.

Throughout the paper, for our coreset construction algorithms, we use $X^{(1)}, \cdots, X^{(m)}$ to denote the collection of the data sets. Note that since $\mathcal{A}(X^{(i)}) \subset X^{(i)}$, the diversity of the combined coresets is always at most the diversity of the combined original datasets. We also say that the coreset is of size k' if $|\mathcal{A}(X^{(i)})| \leq k'$ for each $X^{(i)}$. We desire for the size k' to only depend (polynomially) on k, and not on n.

2.2 The GMM Algorithm

The GMM algorithm [19] is an algorithm that was first developed for the k-center clustering, but has since been of great use in various diversity maximization algorithms and dispersion problems, starting with [30]. The algorithm is a simple greedy procedure that finds k points Y in a dataset X that are well spread out. It starts by picking an arbitrary point $y_1 \in X$. Given $y_1, \ldots, y_p$ for $p < k$, it chooses y_{p+1} as the point that maximizes the distance $\rho(y, \{y_1, \ldots, y_p\})$ over all choices of $y \in X$. We provide pseudocode in Algorithm 1.

The GMM algorithm serves as an important starting point in many of our algorithms, as well as in many of the previous state-of-the-art algorithms for diversity maximization. The GMM algorithm has the following crucial property.

▶ **Proposition 8.** *Suppose we run GMM for k steps to produce $Y = \{y_1, \ldots, y_k\}$. Let $r = \max_{x \in X} \rho(x, Y)$. Then, every pair of points y_i, y_j has $\rho(y_i, y_j) \geq r$.*

3 Composable Coreset Constructions

In this section, we design algorithms for constructing $O(1)$-approximate composable coresets for remote-pseudoforest and remote-matching. In this section, we provide a technical overview and pseudocode for the algorithms, but defer the proof (and algorithm descriptions in words) to Appendix A (for remote-pseudoforest) and Section 5 (for remote-matching).

3.1 Coreset Constructions: Technical Overview

For both remote-pseudoforest and remote-matching, we start by considering the heuristic of GMM, where in each group we greedily select k points. For simplicity, suppose we only have one group for now. After picking the set $Y = \{y_1, \ldots, y_k\}$ from GMM, define r to be the maximum distance $\rho(x, Y)$ over all remaining points x. Then, Proposition 8 implies that all distances $\rho(y_j, y_{j'}) \geq r$ for $j, j' \leq k$. Hence, running GMM will ensure we have a set of k points with minimum-weight-matching or pseudoforest cost at least $\Omega(k \cdot r)$. Hence, we only fail to get an $O(1)$-approximation if the optimum remote-matching (or remote-pseudoforest) cost is much larger than $k \cdot r$.

However, in this case, note that every point $x \in X$ satisfies $\rho(x, Y) \leq r$, meaning every point x is within r of some y_i. This suggests that if the optimum cost is $\omega(k \cdot r)$, achieved by some points $z_1, \ldots, z_k$, we could just map each z_i to its closest y_i, and this would change each distance by no more than $O(k \cdot r)$. Hence, we can ostensibly use the GMM points to obtain a cost within $O(k \cdot r)$ of the right answer, which is within an $\Omega(1)$ (in fact a $1 - o(1)$) multiplicative factor! Additionally, this procedure will compose nicely, because if we split the data into m components $X^{(1)}, \ldots, X^{(m)}$, each with corresponding radius $r^{(1)}, \ldots, r^{(m)}$, then each individual coreset has cost at least $r^{(j)}$ (so the combination has cost at least $\max r^{(i)}$), whereas we never move a point more than $r^{(i)} \leq \max r^{(i)}$.

The problem with this, however, is that we may be using each y_j multiple times: for instance, if both z_1 and z_2 are closest to y_1, we would use y_1 twice. Our goal is to find a subset of k points, meaning we cannot duplicate any point.

Note that in this duplication, it is never necessary to duplicate a point more than k times. So, if we could somehow pick k copies of each y_i, we would have a coreset. However, note that it is not crucial for each z_i to be mapped to its closest GMM point y_j: any point within distance r of y_j is also acceptable. Using this observation, it suffices to pick k points among those closest to y_j if possible - if there are fewer than k points, picking all of the points is sufficient. It is also important to choose all of Y, in the case where the optimum cost is only $O(k \cdot r)$. Together, this generates a composable coreset for both remote-matching and remote-pseudoforest, of size only k^2.

3.1.1 Improving the coreset for remote-matching

In the case of remote-matching, we can actually improve this to $O(k)$. The main observation here is to show that if a set Z of size k had two identical points, getting rid of both of them does not affect the minimum-weight-matching cost. (This observation does *not* hold for pseudoforest). One can similarly show that if the two points were close in distance, removing both of the points does not affect the matching cost significantly. This also implies we can, rather than removing both points, move them both to a new location as long as they are close together. At a high level, this means that there must exist a near-optimal k-matching that only has $O(1)$ points closest to each y_j: as a result we do not have to store k points for each y_j: only $O(1)$ points suffice.

3.1.2 Improving the coreset for remote-pseudoforest

In the case of remote-pseudoforest, the improvement is more involved. Consider a single group, and suppose GMM gives us the set $Y = \{y_1, \ldots, y_k\}$. Let X_i represent the set of points in X closest to Y_i. The first observation we make is that if the optimal solution had multiple points in a single X_i, each such point can only contribute $O(r)$ cost. Assuming that the optimum cost is $\omega(k \cdot r)$, it may seem sufficient to simply pick 1 point in each Y_i for the coreset, as we can modify the optimum solution by removing points in X_i if there are two or more of them. While this will allow us to obtain a set with nearly optimum cost, the problem is the set has size less than k. So, we need to add additional points while preventing the pseudoforest cost from decreasing by too much.

To develop intuition for how this can be accomplished, we first suppose that $|X_1|, |X_2| \geq k$. In this case, we could choose the coreset as $X_1 \cup X_2 \cup Y$. We know there is a subset $Z \subset Y$ with pseudoforest cost close to optimum, though $|Z|$ may be much smaller than k. However, since y_1, y_2 are far away from each other (they were chosen first in the greedy procedure of GMM), so all points in X_1 and all points in X_2 are far from each other. That means

Algorithm 2 PFCoreset: $O(1)$-approximate remote-pseudoforest composable coreset algorithm.

Input: data $X = \{x_1, \ldots, x_n\}$, integer k, parameter $\varepsilon \in (0, 1]$.
if $n < 2k^{1+\varepsilon} + k$ **then**
  Return X.
end if
$Y = \{y_1, \ldots, y_k\} \leftarrow \text{GMM}(x_1, \ldots, x_n, k)$.
for $i = 1$ to n **do**
  $\tilde{r}_i \leftarrow (k+1)^{\text{th}}$ largest value of $\rho(x_i, x_j)$ across all $j \le n$.
end for
$\tilde{r} \leftarrow \min_i \tilde{r}_i$, $x \leftarrow \arg\min_i \tilde{r}_i$
$U \leftarrow k$ furthest points in X from x.
$P \leftarrow k$ arbitrary points within distance $\tilde{r}$ of x.
$S, T \leftarrow \text{FindST}(X, k, \varepsilon, \tilde{r})$. {See Algorithm 5}
Return $C \leftarrow P \cup S \cup T \cup U \cup Y$.

that if we randomly choose either X_1 or X_2, and add enough points from the chosen set so that we have k points, each point $y \in Y$, with 50% probability, is not close to the new points added (because every point y must either be far from X_1 or far from X_2). Thus, the expected distance from y to the closest new point is large.

In general, it is not actually important that the points come from X_1 and X_2: we just need two sets S, T of k points such that $\rho(S, T)$, the minimum distance between $s \in S$ and $t \in T$, is large. Then, any point y cannot be close to points in both S and T. This also composes nicely, because to find the final set of k points, we only need there to be two sets S, T throughout the union of the coresets with large $\rho(S, T)$.

To find large S, T with large $\rho(S, T)$, we will require $|X| \ge k^{1+\varepsilon}$ for some small constant ε. For simplicity, we focus on the case when $|X| \ge k^{1.5}$. Suppose all but k points are in some ball B of radius r. If there exists x that is within distance $r/10$ of k points (we can make S these k points), then all points in S must be far away from the furthest k points from x (which we can set as T), or else we could have found a smaller ball B'. Otherwise, there are two options.

1. The majority of points $x \in X$ are within $r/100$ of at least $\sqrt{k}$ other points, but no $x \in X$ is within $r/10$ of at least k other points. Intuitively (we will make this intuition formal in Appendix A), a random set S_0 of size $O(\sqrt{k})$ should be within $r/100$ of at least k other points in total (we can make S these k points), but there are at least $|X| - k \cdot |S_0| \ge k$ points (we can make T these points) that are not within $r/10$ of S_0.
2. The majority of points aren't within $r/100$ of even $O(\sqrt{k})$ points. In this case, we can pick k of these points to form S, and they will not be within $r/100$ of at least $|X| - O(\sqrt{k}) \cdot |S| \ge k$ points. We make this intuition formal and prove the result for the more general $k^{1+\varepsilon}$.

3.2 Algorithm Pseudocode

We provide pseudocode for the remote-pseudoforest coreset in Algorithm 2 and for the remote-matching coreset in Algorithm 3. The proofs, as well as algorithm descriptions in words, are deferred to Section 5 (for remote-matching) and Appendix A (for remote-pseudoforest).

Algorithm 3 MWMCoreset: $O(1)$-approximate remote-matching composable coreset algorithm.

Input: data $X = \{x_1, \ldots, x_n\}$, integer k.
if $n \leq 3k$ **then**
  Return X
else
  $Y = \{y_1, \ldots, y_k\} \leftarrow \text{GMM}(x_1, \ldots, x_n, k)$.
  Initialize $S_1, \ldots, S_k \leftarrow \emptyset$.
  for $i = 1$ to n **do**
    Add i to S_j for $j = \arg\min \rho(x_i, y_j)$.
  end for
  Initialize $C \leftarrow Y$.
  for $i = 1$ to $k/2$ **do**
    Find $x, x' \in X \backslash C$ such that x, x' are in the same S_j.
    $C \leftarrow C \cup \{x, x'\}$.
  end for
  Return C.
end if

4 Offline Remote-Matching Algorithm

In this section, we design $O(1)$-approximate offline algorithms for remote-matching. In this section, we first provide a technical overview, then the algorithm description and pseudocode, and finally we provide the full analysis.

4.1 Technical Overview

The remote-matching offline algorithm first utilizes some simple observations that we made in Section 3.1. Namely, we may assume the largest minimum-weight matching cost of any subset of k points is $\omega(k \cdot r)$, or else GMM provides an $O(1)$-approximation. Next, if the optimum solution was some $Z = \{z_1, \ldots, z_k\}$, we can again consider mapping each z_i to its closest y_j, at the cost of having duplicates. However, as noted in Section 3.1, we may delete a point twice without affecting the matching cost: this means we can keep deleting a point twice until each y_j is only there 0 times (if the total number of z_i's closest to y_j was even) or 1 time (if the total number of z_i's closest to y_j was odd).

However, we have no idea what Z actually is, so we have no idea whether each y_j should be included or not. However, this motivates the following simpler problem: among the k points Y, choose a (even-sized) subset of Y maximizing the matching cost.

One attempt at solving this problem is to choose $\{y_1, \ldots, y_p\}$ for some $p \leq k$: this will resemble an argument in [11]. The idea is that if we define r_p to be the maximum value $\rho(x, \{y_1, \ldots, y_p\})$ over all $x \in X$, the same argument as Proposition 8 implies that all points among $y_1, \ldots, y_p$ are separated by at least r_p. Hence, for the best p we can obtain matching cost $\Omega(\max_{1 \leq p \leq k} p \cdot r_p)$. Conversely, it is known that the minimum-weight-matching cost of any set of points Z is upper-bounded by the cost of the minimum spanning tree of Z. But the minimum spanning tree has cost at most $\sum_{p=1}^{k} r_p$, since we can create a tree by adding an edge from each y_{p+1} to its closest center among $y_1, \ldots, y_p$, which has distance r_p. Since $\max(p \cdot r_p) \geq \Omega\left(\frac{1}{\log k}\right) \cdot \sum_{p=1}^{k} r_p$ (with equality for instance if $r_p = \frac{1}{p}$), we can obtain an $O(\log k)$-approximation.

For simplicity, we focus on the case where $r_p = \Theta(1/p)$ for all $1 \le p \le k$. We would hope that either the minimum spanning tree cost of Y, which we call $\mathrm{MST}(Y)$, is actually much smaller than $\log k$, or there is some alternative selection to obtain matching cost $\Omega(\log k)$ rather than $O(1)$. Suppose that $\mathrm{MST}(Y) = \Omega(\log k)$: furthermore, for simplicity suppose the pth largest edge of the tree has weight $\frac{1}{p}$. If we considered the graph on Y connecting two points if their distance is less than $\frac{1}{p}$, it is well-known that the graph must therefore split into p disconnected components.

Now, for some fixed p suppose that we chose a subset Z of Y such that each connected component in the graph above has an odd number of points in Z. Then, any matching must send at least one point in each $Z \cap CC_j$ (where CC_j is the jth connected component) to a point in a different connected component, forcing an edge of weight at least $\frac{1}{p}$. Since each of p connected components has such a point, together we obtain weight at least 1. In addition, if we can ensure this property for $p = 2, 4, 8, 16 \ldots, k$, we can in fact get there must be at least 2^i edges of weight $1/2^i$, making the total cost $\Omega(\log k)$, as desired.

While such a result may not be possible exactly, it turns out that even a *random* subset of Y satisfies this property asymptotically! Namely, if we choose each point $y \in Y$ to be in Z with 50% probability, each CC_j is odd with 50% probability. So in expectation, for all p, the number of connected components of odd size is $p/2$. Even if we make sure Z has even size, this will still be true, replacing $p/2$ with $\Omega(p)$. Since this is true for all p in expectation, by adding over powers of 2 for p, we will find k points with $\Omega(\log k)$ matching cost in expectation.

4.2 Algorithm Desciption and Pseudocode

Given a dataset $X = \{x_1, \ldots, x_n\}$, we recall that the goal of the remote-matching problem is to find a subset $Z = \{z_1, \ldots, z_k\} \subset X$ of k points, such that the minimum-weight matching cost of Z, $\mathrm{MWM}(Z)$, is approximately maximized. In this subsection, we describe our $O(1)$-approximate remote-matching algorithm. We also provide pseudocode in Algorithm 4.

Algorithm Description

The algorithm proceeds as follows. First, run the GMM algorithm for k steps, to obtain k points $Y = \{y_1, \ldots, y_k\} \subset X$. Define the subsets $S_1, \ldots, S_k$ as a partitioning of X, where $x \in X$ is in S_i if y_i is the closest point to x in Y. (We break ties arbitrarily.) We use the better of the following two options, with the larger minimum-weight matching cost.

1. Simply use $Y = \{y_1, \ldots, y_k\}$.
2. Let $\hat{Z} \subset Y$ be a uniformly random subset of Y. Initialize W to $\hat{Z}$ if $|\hat{Z}|$ is even, and otherwise initialize W to $\hat{Z} \backslash \hat{z}$ for some arbitrary $\hat{z} \in \hat{Z}$. Now, if there exist two points not in $W \cup Y$ but in the same subset S_i, add both of them to W. Repeat this procedure until $|W| = k$.

We will use whichever of Y or W has the larger minimum-weight matching cost. Since minimum-weight matching can be computed in polynomial time, we can choose the better of these two in polynomial time.

In Theorem 3, we assume $n \ge 3k$. Because of this assumption, if $|W| < k$, then $|W \cup Y| \le 2k - 1$, which means $|X \backslash (W \cup Y)| \ge k + 1$. Hence, by Pigeonhole Principle, two of these points must be in the same set S_i, which means that the procedure described above is indeed doable.

Algorithm 4 MWMOffline: $O(1)$-approximate remote-matching algorithm.

1: **Input:** data $X = \{x_1, \ldots, x_n\}$, even integer k.
2: $Y = \{y_1, \ldots, y_k\} \leftarrow \text{GMM}(x_1, \ldots, x_n, k)$.
3: **Initialize** $S_1, \ldots, S_k \leftarrow \emptyset$.
4: **for** $i = 1$ to n **do**
5: Add i to S_j if $j = \arg\min \rho(x_i, y_j)$.
6: **end for**
7: $Z \leftarrow$ random subset of Y.
8: **if** $|Z|$ is odd **then**
9: Remove an arbitrary element from Z
10: **end if**
11: **Initialize** $W \leftarrow Z$.
12: **while** $|W| < k$ **do**
13: Find some $x, x' \in X \backslash (W \cup Y)$, such that x, x' are in the same subset S_j.
14: Add x, x' to W.
15: **end while**
16: **Return** whichever of Y, W has larger minimum-weight matching cost.

4.3 Analysis

The first ingredient in proving Theorem 3 is the following lemma, which shows that assuming the random subset Z we chose in Line 7 of Algorithm 4 is sufficiently good, the algorithm produces an $O(1)$-approximation.

▶ **Lemma 9.** *For some constant $\frac{1}{2} \geq \alpha > 0$, suppose that*

$$\text{MWM}(Z) \geq \alpha \cdot \max_{Z' \subset Y : |Z'| \text{ is even}} \text{MWM}(Z').$$

Then, Algorithm 4 provides a $\frac{4}{\alpha}$-approximation for the remote-matching problem.

Proof. Let M be the optimal remote-matching cost. Let r be the maximum distance from any point in $X \backslash Y$ to its closest point in Y. Note that $\rho(y_i, y_j) \geq r$ for all $i, j \leq k$, by Proposition 8.

First, suppose that $M \leq 2\alpha^{-1} \cdot r \cdot k$. In this case, because every pair in Y has pairwise distance at least r, we have $\text{MWM}(Y) \geq r \cdot \frac{k}{2}$. Hence, $\text{MWM}(Y) \geq \frac{\alpha}{4} \cdot M$, which means we have a $\frac{4}{\alpha}$-approximation.

Alternatively, suppose $M \geq 2\alpha^{-1} \cdot r \cdot k$. Let $W_0 \subset X$ be the set of p points that achieves this, i.e., $\text{MWM}(W_0) = M$. Consider the following multiset $\tilde{W}_0$ of size p in Y, where each point in W_0 is mapped to its closest center in Y (breaking ties in the same way as in the algorithm). Then, every pair of distances between W_0 and $\tilde{W}_0$ changes by at most $2r$. This means *every* matching has its cost change by at most $\frac{k}{2} \cdot 2r = rk$, so $\text{MWM}(\tilde{W}_0) \geq M - rk$.

Now, let $Z_0 \subset Y$ be the set of points where $y_i \in Z_0$ if and only if y_i is in $\tilde{W}_0$ an odd number of times. Then, $\text{MWM}(\tilde{W}_0) = \text{MWM}(Z_0)$. To see why, first note that $\text{MWM}(\tilde{W}_0) \leq \text{MWM}(Z_0)$ since we can convert any matching of Z_0 to a matching of $\tilde{W}_0$ by simply matching duplicate points in $\tilde{W}_0$ until only Z_0 is left. To see why $\text{MWM}(\tilde{W}_0) \geq \text{MWM}(Z_0)$, note that if an optimal matching of $\tilde{W}_0$ connected some copy of y to a point $y' \neq y$ and another copy of y to a point $y'' \neq y$, we can always replace the edges (y, y') and (y, y'') with (y, y) and (y', y''), which by Triangle inequality will never increase the cost. We may keep doing this until a maximal number of duplicate points are matched together, and only one copy of each element in Z_0 will be left. Hence, we have

$$\text{MWM}(Z_0) = \text{MWM}(\tilde{W}_0) \geq M - rk. \tag{1}$$

Similarly, let Z, W be the sets found in the algorithm described above, and let $\tilde{W}$ be the multiset formed by mapping each point in W to its nearest center in Y. As in the case with Z_0 and $\tilde{W}_0$, we have that for Z and $\tilde{W}$, a point z is in Z if and only if z is in $\tilde{W}$ an odd number of times. Hence, $\mathrm{MWM}(\tilde{W}) = \mathrm{MWM}(Z)$. Likewise, each point in $\tilde{W}$ has distance at most r from its corresponding point in W, which means $\mathrm{MWM}(W) \geq \mathrm{MWM}(\tilde{W}) - rk$. Hence, we have

$$\mathrm{MWM}(Z) = \mathrm{MWM}(\tilde{W}) \leq \mathrm{MWM}(W) + rk. \tag{2}$$

Overall, $\mathrm{MWM}(Z) \geq \alpha \cdot \max_{Z \subset Y : |Z| \text{ is even}} \mathrm{MWM}(Z) \geq \alpha \cdot \mathrm{MWM}(Z_0)$, so

$$\begin{aligned}\mathrm{MWM}(W) &\geq \mathrm{MWM}(Z) - rk \geq \alpha \cdot \mathrm{MWM}(Z_0) - rk \\ &\geq \alpha \cdot M - (1+\alpha)rk.\end{aligned}$$

But note that $M \geq 2\alpha^{-1}rk$, which means that $(1+\alpha)rk \leq \frac{\alpha(1+\alpha)}{2} \cdot M \leq \frac{3}{4} \cdot \alpha \cdot M$. Hence, $\mathrm{MWM}(W) \geq \frac{\alpha}{4} \cdot M$, which again means we have a $\frac{4}{\alpha}$-approximation. ◀

The main technical lemma that we will combine with Lemma 9 shows that Z has the desired property. We now state the lemma, but we defer the proof slightly, to Section 4.5. We remark that the proof roughly follows the intuition described at the end of Section 4.1.

▶ **Lemma 10.** *Let $\hat{Z}$ be a random subset of Y where each element is independently selected with probability $1/2$. If $|\hat{Z}|$ is even, we set $Z = \hat{Z}$, and if $|\hat{Z}|$ is odd, we arbitrarily remove 1 element from $\hat{Z}$ to generate Z. Then,*

$$\mathbb{E}[\mathrm{MWM}(Z)] \geq \frac{1}{16} \cdot \max_{Z' \subset Y : |Z'| \text{ is even}} \mathrm{MWM}(Z').$$

Given Lemmas 9 and 10, we explain how combine them to prove Theorem 3.

Proof of Theorem 3. Suppose we generate a random subset Z of Y (possibly removing an element), and suppose that $\mathrm{MWM}(Z) = \alpha \cdot \max_{Z \subset Y : |Z| \text{ is even}} \mathrm{MWM}(Z)$. Then, the output of the algorithm has matching cost at least $\frac{\alpha}{4}$ times the optimum k-matching cost $\mathrm{MWM}_k(X)$, by Lemma 9. However, by Lemma 10, $\mathbb{E}[\alpha] \geq \frac{1}{16}$, which means that the expected matching cost of the output is $\frac{\mathbb{E}[\alpha]}{4} \cdot \mathrm{MWM}_k(X) \geq \frac{1}{64} \cdot \mathrm{MWM}_k(X)$.

If we want this to occur with high probability, note that the matching cost of the output can never be more than $\mathrm{MWM}_k(X)$. Hence, by Markov's inequality, with at least $\frac{1}{64^2} = \frac{1}{4096}$ probability, the output has matching cost at least $\frac{1}{65} \cdot \mathrm{MWM}_k(X)$. If we repeat this $O(1)$ times and return the set with best matching cost, we can find a set of size k with matching cost at least $\frac{1}{65} \cdot \mathrm{MWM}_k(X)$, with probability at least 0.99. ◀

Before proving Lemma 10, we will need some additional preliminaries.

4.4 Preliminaries for Lemma 10

To prove Lemma 10, we will need several preliminary facts relating to the cost of a minimum weight matching, as well as the cost of a minimum spanning tree of a set of points.

First, we have the following fact, bounding the minimum weight matching in terms of the MST.

▶ **Proposition 11** (Classical, see Proof of Lemma 5.2 in [11]). *For any (finite, even sized) set of data points Z in a metric space, $\mathrm{MWM}(Z) \leq \mathrm{MST}(Z)$.*

Next, given a subset Z of Y in a metric space, we can bound the minimum spanning tree cost of Z in terms of the minimum spanning cost of Y.

▶ **Proposition 12** (Classical, see [16]). *Let $Z \subset Y$ be (finite) sets of data points in some metric space. Then,* $\mathrm{MST}(Z) \leq 2 \cdot \mathrm{MST}(Y)$.

Next, we equate the minimum spanning tree of a dataset Y with the number of connected components in a family of graphs on Y. The following proposition essentially follows from the same argument as in [12, Lemma 2.1]. We prove it here for completeness since the statement we desire is not explicitly proven in [12].

▶ **Proposition 13.** *Given a dataset Y in a metric space and a radius $r > 0$, define $G_r(Y)$ to be the graph on Y that connects two data points if and only if their distance is at most r. Define $P_r(Y)$ to be the number of connected components in $G_r(Y)$. Then,*

$$\mathrm{MST}(Y) \in \left[\frac{1}{2}, 1\right] \cdot \left(\sum_{i \in \mathbb{Z}} 2^i \cdot (P_{2^i}(Y) - 1)\right).$$

Proof. Note that if 2^i is at least $\mathrm{diam}(Y)$, the diameter of Y, $P_{2^i}(Y) = 1$, which means we may ignore the summation for i with $2^i > \mathrm{diam}(Y)$. Hence, by scaling by some power of 2, we may assume WLOG that $\mathrm{diam}(Y) < 1$, and that the summation is only over $i < 0$.

Now, for any $t \geq 0$, let $Q_t(Y)$ be the number of edges in the MST of Y with weight at most 2^{-t} and strictly more than $2^{-(t+1)}$ (assuming we run Kruskal's algorithm for MST). Note that $R_t(Y) := \sum_{t' \geq t} Q_{t'}(Y)$ is the number of edges with weight at most 2^{-t}. Note that $R_t(Y)$ is precisely $n - P_{2^{-t}}(Y)$. To see why, note that the $R_t(Y)$ edges form a forest with $n - R_t(Y)$ connected components. In addition, in the graph $G_{2^{-t}}(Y)$, none of the $n - R_t(Y)$ components can be connected to each other, or else there would have been another edge of weight at most 2^{-t} that Kruskal's algorithm would have had to add. Therefore, $\sum_{t' \geq t} Q_{t'}(Y) = n - P_{2^{-t}}(Y)$. By subtracting this equation from the same equation replacing t with $t + 1$, we obtain

$$Q_t(Y) = P_{2^{-(t+1)}}(Y) - P_{2^{-t}}(Y). \tag{3}$$

Now, note that by definition of $Q_t(Y)$, the cost of $\mathrm{MST}(Y)$ is between $\sum_{t \geq 0} 2^{-(t+1)} \cdot Q_t(Y)$ and $\sum_{t \geq 0} 2^{-t} \cdot Q_t(Y)$. Equivalently, it equals $\alpha \cdot \left(\sum_{t \geq 0} 2^{-t} \cdot Q_t(Y)\right)$, for some $\alpha \in [1/2, 1]$. Therefore,

$$\begin{aligned}
\mathrm{MST}(Y) &= \alpha \cdot \left(\sum_{t \geq 0} 2^{-t} \cdot Q_t(Y)\right) \\
&= \alpha \cdot \left(\sum_{t \geq 0} 2^{-t} \cdot (P_{2^{-(t+1)}}(Y) - P_{2^{-t}}(Y))\right) \\
&= \alpha \cdot \left(\sum_{t \geq 0} (2^{-t} - 2^{-(t+1)}) P_{2^{-(t+1)}}(Y) - P_1(Y)\right) \\
&= \alpha \cdot \left(\sum_{t \geq 1} 2^{-t} P_{2^{-t}}(Y) - 1\right) \\
&= \alpha \cdot \left(\sum_{t \geq 1} 2^{-t} (P_{2^{-t}}(Y) - 1)\right).
\end{aligned}$$

The second-to-last line follows since the diameter is at most 1 so $G_1(Y)$ has one connected component, and the last line follows because $\sum_{t \geq 1} 2^{-t} = 1$. ◀

Finally, we need to consider the minimum weight matching cost in a *hierarchically well-separated tree* (HST).

▶ **Definition 14.** *A* hierarchically-well seprated tree (HST) *is a depth-d tree (for some integer $d \geq 1$) with the root as depth 0, and every leaf has depth d. For any node u in the tree of depth $1 \leq t \leq d$, each edge from u to its parent has weight 2^{-t}. For two nodes v, w in the HST, the distance $d_{\mathrm{HST}}(v, w)$ is simply the sum of the edge weights along the shortest path from v to w in the tree.*

Note that for any two leaf nodes v, w in an HST, if their least common ancestor has depth t, the distance between v and w is $2 \cdot \left(2^{-(t+1)} + 2^{-(t+2)} + \cdots + 2^{-d}\right) = 2 \cdot (2^{-t} - 2^{-d})$.

We will make use of the following result about points in an HST metric.

▶ **Proposition 15** ([21], Claim 3)**.** *Let Z be a (finite, even sized) set of points that are leaves in a depth-d HST. Let m_i be the number of nodes at level i with an* odd *number of descendants in Z. Then, with respect to the HST metric, the minimum weight matching cost equals*

$$\mathrm{MWM} = \sum_{i=0}^{d} 2^{-i} \cdot m_i.$$

We remark that the corresponding statement in [21] has an additional additive factor of $n = |Z|$ in the right-hand side. This is because we include the bottom level in our sum (which consists of n nodes each with exactly one descendant), whereas [21] does not.

4.5 Proof of Lemma 10

We are now ready to prove Lemma 10

Proof of Lemma 10. Assume WLOG (by scaling) that the diameter of Y is at most 1. Let Z be a subset of Y with even size. By Proposition 11, $\mathrm{MWM}(Z) \leq \mathrm{MST}(Z)$. By Proposition 12, $\mathrm{MST}(Z) \leq 2 \cdot \mathrm{MST}(Y)$. Combining these together, we have

$$\max_{Z \subset Y : |Z| \text{ even}} \mathrm{MWM}(Z) \leq 2 \cdot \mathrm{MST}(Y). \tag{4}$$

Now, for our dataset Y and any positive real $r > 0$, recall that $G_r(Y)$ is defined as the graph on Y that connects two data points if their distance is at most r. In addition, define $\mathcal{P}_r(Y)$ to be the partitioning of Y into connected components based on $G_r(Y)$, and recall that $P_r(Y) = |\mathcal{P}_r(Y)|$ equals the number of connected components in $G_r(Y)$. Then, Proposition 13 tells us that

$$\mathrm{MST}(Y) \leq \sum_{i \in \mathbb{Z}} 2^i \cdot (P_{2^i}(Y) - 1). \tag{5}$$

Now, consider the following "embedding" of Y into a depth-d hierarchically well-separated tree (where we will choose d later) as follows. By scaling, assume WLOG that the diameter of Y is 1. For each integer $0 \leq t \leq d$, the nodes at level t will be the connected components in $G_{2^{-t}}(Y)$, where the children of any node at depth t, represented by a subset Z of Y, are simply the connected components in $\mathcal{P}_{2^{-(t+1)}}(Y)$ contained in Z.

The distance $d_{\mathrm{HST}}(y_i, y_j)$ between any two vertices y_i, y_j in the HST is precisely $2(2^{-t} - 2^{-d})$ if y_i, y_j have common ancestor at level t of the HST. Note that if $d_{\mathrm{HST}}(y_i, y_j) = 2(2^{-t} - 2^{-d})$, then y_i, y_j are not in the same connected component of $G_{2^{-(t+1)}}$, which means that $\rho(y_i, y_j) > 2^{-(t+1)}$. Importantly, this means that $d_{\mathrm{HST}}(y_i, y_j) \leq 4\rho(y_i, y_j)$ for all

pairs i, j. Hence, for any subset $Z \subset Y$ of even size, the minimum weight matching cost $\mathrm{MWM}_{\mathrm{HST}}(Z)$ with respect to the HST metric is at most 4 times the true minimum weight matching cost, i.e.,

$$\mathrm{MWM}_{\mathrm{HST}}(Z) \leq 4 \cdot \mathrm{MWM}(Z). \tag{6}$$

Finally, we consider selecting a random subset $\hat{Z} \subset Y$, and provide a lower bound for $\mathrm{MWM}_{\mathrm{HST}}(Z)$, where $Z = \hat{Z}$ if $|\hat{Z}|$ is even and otherwise Z equals $\hat{Z}$ after removing a single (arbitrary) element. Note that for each node v of depth t, corresponding to a connected component in $\mathcal{P}_{2^{-t}}(Y)$, the probability that it has an odd number of descendants in $\hat{Z}$ if $\hat{Z}$ is picked at random is precisely $1/2$. This implies that the expectation of $\sum_{i=0}^{d} 2^{-i} \cdot m_i$, where m_i is the number of nodes at level i with an odd number of descendants in $\hat{Z}$, is $\frac{1}{2} \cdot \sum_{i=0}^{d} 2^{-i} \cdot P_{2^{-i}}(Y)$.

Note, however, that $\hat{Z}$ has odd size with $1/2$ probability. In this event, we remove an arbitrary element of $\hat{Z}$, which may reduce each m_i by 1. This only happens with 50% probability, so after this potential removal of a point, the expectation of $\sum_{i=0}^{d} 2^{-i} \cdot m_i(Z)$, where $m_i(Z)$ is the number of nodes at level i with an odd number of descendants in Z, is at least $\frac{1}{2} \cdot \sum_{i=0}^{d} 2^{-i} \cdot (P_{2^{-i}}(Y) - 1)$. Since $\mathrm{diam}(Y)$ is at most 1, this implies $P_{2^i}(Y) - 1 = 0$ for all $i \geq 1$. Also, for $i > d$, $2^{-i} \cdot (P_{2^{-i}}(Y) - 1) \leq 2^{-i} \cdot n$. If we sum this up over all $i > d$, this is still at most $2^{-d} \cdot n$. Hence, we have that

$$\mathbb{E}_Z \left[\sum_{i=0}^{d} 2^{-i} \cdot m_i(Z) \right] \geq \frac{1}{2} \cdot \left(\sum_{i \in \mathbb{Z}} 2^i (P_{2^i}(Y) - 1) \right) - n \cdot 2^{-d}. \tag{7}$$

In summary, we have that

$$\begin{aligned}
\max_{Z' \subset Y : |Z'| \text{ even}} \mathrm{MWM}(Z') &\leq 2 \cdot \mathrm{MST}(Y) && \text{By Equation (4)} \\
&\leq 2 \cdot \sum_{i \in \mathbb{Z}} 2^i \cdot (P_{2^i}(Y) - 1) && \text{By Equation (5)} \\
&\leq 4 \cdot \left(\mathbb{E}_Z \left[\sum_{i=0}^{d} 2^{-i} \cdot m_i(Z) \right] + n \cdot 2^{-d} \right) && \text{By Equation (7)} \\
&= 4 \cdot \left(\mathbb{E}_Z \left[\mathrm{MWM}_{\mathrm{HST}}(Z) \right] + n \cdot 2^{-d} \right) && \text{By Proposition 15} \\
&\leq 16 \cdot \left(\mathbb{E}_Z \left[\mathrm{MWM}(Z) \right] + n \cdot 2^{-d} \right). && \text{By Equation (6)}
\end{aligned}$$

We can choose the depth of the HST to be arbitrarily large, which therefore implies that

$$\mathbb{E}_Z[\mathrm{MWM}(Z)] \geq \frac{1}{16} \cdot \max_{Z' \subset Y : |Z'| \text{ even}} \mathrm{MWM}(Z'). \qquad \blacktriangleleft$$

5 Coreset for Remote-Matching

In this section, we prove why the algorithm given in Algorithm 3 creates an $O(1)$-approximate composable coreset. First, we describe the algorithm in words.

5.1 Algorithm Description

We start by running GMM on the dataset X for k steps, to return k points $Y = \{y_1, \ldots, y_k\}$. Again, let the subsets $S_1, \ldots, S_k$ be a partitioning of X, where $x \in X$ is in S_i if y_i is the closest point in Y to x (breaking ties arbitrarily). Note that $y_i \in S_i$ for all i.

To create our coreset C for X, if $|X| \leq 3k$ we simply define $C = X$. Otherwise, we start by initializing C to be Y, so C currently has size k. Next, for $k/2$ steps, we find any two points in $X \backslash C$ that are in the same partition piece S_i, and add both of them to C. Hence, at the end $|C| = 2k$. Note that this procedure is always doable, since we are assuming $|X| \geq 3k + 1$, which means if we have picked at most $2k$ total elements, there are $k + 1$ remaining elements in X, of which at least 2 must be in the same S_i by the pigeonhole principle.

5.2 Analysis

In this subsection, we prove that the algorithm generates an $O(1)$-approximate composable coreset, by proving Theorem 5.

Proof of Theorem 5. Suppose we run this algorithm for each of m datasets, $X^{(1)}, \ldots, X^{(m)}$, to generate coresets $C^{(1)}, \ldots, C^{(m)}$. We wish to show that the optimum k-matching cost of $C = \bigcup_{j=1}^{m} C^{(j)}$ is at least $\Omega(1)$ times the optimum k-matching cost of $X = \bigcup_{j=1}^{m} X^{(j)}$.

Let $Y^{(j)} = \{y_1^{(j)}, \ldots, y_k^{(j)}\}$ represent the k points we obtained by running GMM on $X^{(j)}$, and let $r^{(j)}$ be the maximum distance from any point in $X^{(j)} \backslash Y^{(j)}$ to its closest point in $Y^{(j)}$. Then, note that all points in $Y^{(j)}$ are pairwise separated by at least $r^{(j)}$. Let $r = \max_{1 \leq j \leq m} r^{(j)}$.

First, suppose that the optimum k-matching cost of X is $M \leq 5r \cdot k$. In this case, for the $r^{(j)}$ that equals r, the GMM algorithm finds k points that are pairwise separated by at least $r^{(j)} = r$. Since $C^{(j)} \supset Y^{(j)}$, this means that the full coreset C contains k points that are pairwise separated by r, which has k-matching cost at least $r \cdot \frac{k}{2}$. Hence, we have a 10-approximate coreset.

Alternatively, the optimum k-matching cost of X is $M \geq 5r \cdot k$. Let $S_i^{(j)}$ represent the set S_i for $X^{(j)}$, and suppose W is an optimal set of k points in X with $\mathrm{MWM}(W) = M$. Let $W^{(j)} = W \cap X^{(j)}$. Also, let $W_i^{(j)} = W \cap S_i^{(j)}$ and $b_i^{(j)}$ be the *parity* of $|W_i^{(j)}|$, i.e., $b_i^{(j)} = 1$ if $|W_i^{(j)}|$ is odd and $b_i^{(j)} = 0$ if $|W_i^{(j)}|$ is even. In addition, let $\tilde{W}$ be the multiset of k points formed by mapping each point in $W_i^{(j)}$ to $y_i^{(j)}$. In other words, $\tilde{W}$ consists of each $y_i^{(j)}$ repeated $|W_i^{(j)}|$ times. Note that since each $W_i^{(j)}$ has distance at most $r^{(j)} \leq r$ from $y_i^{(j)}$, all pairwise distances change by at most $2r$, which means the matching cost difference $|\mathrm{MWM}(\tilde{W}) - \mathrm{MWM}(W)| \leq \frac{1}{2} \cdot 2r \cdot k = rk$. Also, note that $\tilde{W}$ only consists of points of the form $y_i^{(j)}$, with the parity of the number of times $y_i^{(j)}$ appears in $\tilde{W}$ equaling $b_i^{(j)}$.

Next, we create a similar set $W' \subset C$. For each $j \leq m$, define $k^{(j)} = |W^{(j)}|$. We will find a set $(W')^{(j)} \subset C^{(j)}$ of size $k^{(j)}$, such that the parity of $|(W')^{(j)} \cap S_i^{(j)}|$ equals $b_i^{(j)}$ for all $i \leq k$. To do so, first note that if $|X^{(j)}| \leq 3k$, then $C^{(j)} = X^{(j)}$, so we can just choose $(W')^{(j)} = W^{(j)}$. Otherwise, $|X^{(j)}| \geq 3k + 1$, and $C^{(j)}$ consists of $2k$ points. In addition, $C^{(j)} \supset Z^{(j)}$. Now, we start by including in $(W')^{(j)}$ each point $y_i^{(j)}$ such that $b_i^{(j)} = 1$. Since $b_i^{(j)} = 1$ means that $|W_i^{(j)}|$ is odd, for any fixed j the number of $b_i^{(j)} = 1$ is at most $k^{(j)}$ and has the same parity as $k^{(j)}$. Now, as long as $|(W')^{(j)}| < k^{(j)}$, this means $|C^{(j)} \backslash (W')^{(j)}| \geq k + 1$, which means there are two points in $C^{(j)} \backslash (W')^{(j)}$ that are in the same $S_i^{(j)}$, by pigeonhole principle. We can add both of them to $(W')^{(j)}$. We can keep repeating this procedure until $|(W')^{(j)}| = k^{(j)}$, and note that this never changes the parity of each $|(W')^{(j)} \cap S_i^{(j)}|$.

Our set $W' \subset C$ will just be $\bigcup_{j=1}^{m} (W')^{(j)}$. Note that $|W'| = \sum_{j=1}^{m} k^{(j)} = k$, and $|W' \cap S_i^{(j)}|$ has parity $b_i^{(j)}$, just like W. Hence, we can create the multiset $\tilde{W}'$ by mapping each point $w' \in W' \cap S_i^{(j)}$ to $y_i^{(j)}$. Again, each point moves by at most r, so all pairwise distances change by at most $2r$, which means that $|\mathrm{MWM}(\tilde{W}') - \mathrm{MWM}(W')| \leq rk$.

Finally, we will see that $\mathrm{MWM}(\tilde{W}') = \mathrm{MWM}(\tilde{W})$. Note that both $\tilde{W}$ and $\tilde{W}'$ are multisets of $y_i^{(j)}$ points, each repeated an odd number of times if and only if $b_i^{(j)} = 1$. However, we saw in the proof of Lemma 9 that $\mathrm{MWM}(\tilde{W})$ equals the minimum-weight matching cost of simply including each point $y_i^{(j)}$ exactly $b_i^{(j)}$ times. This is because there exists an optimal matching that keeps matching duplicate points together as long as it is possible. The same holds for $\mathrm{MWM}(\tilde{W}')$, which means $\mathrm{MWM}(\tilde{W}) = \mathrm{MWM}(\tilde{W}')$.

Overall, this means that $|\mathrm{MWM}(W') - \mathrm{MWM}(W)| \leq |\mathrm{MWM}(W') - \mathrm{MWM}(\tilde{W}')| + |\mathrm{MWM}(\tilde{W}') - \mathrm{MWM}(\tilde{W})| + |\mathrm{MWM}(\tilde{W}) - \mathrm{MWM}(W)| \leq rk + 0 + rk = 2rk$. But since we assumed that $\mathrm{MWM}(W) = M \geq 5rk$, this means the k-matching for C is at least $M - 2rk \geq \frac{M}{2}$. Hence, we get a 2-approximate coreset.

In either case, we obtain an $O(1)$-approximate coreset, as desired. ◀

References

1 Sofiane Abbar, Sihem Amer-Yahia, Piotr Indyk, and Sepideh Mahabadi. Real-time recommendation of diverse related articles. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1–12, 2013.

2 Sofiane Abbar, Sihem Amer-Yahia, Piotr Indyk, Sepideh Mahabadi, and Kasturi R Varadarajan. Diverse near neighbor problem. In *Proceedings of the twenty-ninth annual symposium on Computational geometry*, pages 207–214, 2013.

3 Zeinab Abbassi, Vahab S Mirrokni, and Mayur Thakur. Diversity Maximization Under Matroid Constraints. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 32–40, 2013.

4 Sepideh Aghamolaei, Majid Farhadi, and Hamid Zarrabi-Zadeh. Diversity maximization via composable coresets. In *CCCG*, pages 38–48, 2015.

5 Albert Angel and Nick Koudas. Efficient diversity-aware search. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 781–792, 2011.

6 Sepehr Assadi and Sanjeev Khanna. Randomized composable coresets for matching and vertex cover. *arXiv preprint*, 2017. `arXiv:1705.08242`.

7 Aditya Bhaskara, Mehrdad Ghadiri, Vahab S. Mirrokni, and Ola Svensson. Linear relaxations for finding diverse elements in metric spaces. In *Advances in Neural Information Processing Systems*, pages 4098–4106, 2016.

8 Allan Borodin, Hyun Chul Lee, and Yuli Ye. Max-sum diversification, monotone submodular functions and dynamic updates. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pages 155–166, 2012.

9 Matteo Ceccarello, Andrea Pietracaprina, Geppino Pucci, and Eli Upfal. Mapreduce and streaming algorithms for diversity maximization in metric spaces of bounded doubling dimension. *arXiv preprint*, 2016. `arXiv:1605.05590`.

10 Alfonso Cevallos, Friedrich Eisenbrand, and Sarah Morell. Diversity maximization in doubling metrics. *arXiv preprint*, 2018. `arXiv:1809.09521`.

11 Barun Chandra and Magnús M. Halldórsson. Approximation algorithms for dispersion problems. *J. Algorithms*, 38(2):438–465, 2001.

12 Artur Czumaj and Christian Sohler. Estimating the weight of metric minimum spanning trees in sublinear time. *SIAM J. Comput.*, 39(3):904–922, 2009.

13 Marina Drosou and Evaggelia Pitoura. Search result diversification. *ACM SIGMOD Record*, 39(1):41–47, 2010.

14 Alessandro Epasto, Mohammad Mahdian, Vahab Mirrokni, and Peilin Zhong. Improved sliding window algorithms for clustering and coverage via bucketing-based sketches. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3005–3042. SIAM, 2022.

15 Alessandro Epasto, Vahab Mirrokni, and Morteza Zadimoghaddam. Scalable diversity maximization via small-size composable core-sets (brief announcement). In *The 31st ACM symposium on parallelism in algorithms and architectures*, pages 41–42, 2019.

16 E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM J. Appl. Math.*, 16:1–29, 1968.

17 Sreenivas Gollapudi and Aneesh Sharma. An axiomatic approach for result diversification. In *Proceedings of the 18th international conference on World wide web*, pages 381–390, 2009.

18 Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha. Diverse sequential subset selection for supervised video summarization. *Advances in neural information processing systems*, 27, 2014.

19 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

20 Magnús M Halldórsson, Kazuo Iwano, Naoki Katoh, and Takeshi Tokuyama. Finding subsets maximizing minimum structures. *SIAM Journal on Discrete Mathematics*, 12(3):342–359, 1999.

21 Piotr Indyk. Algorithms for dynamic geometric problems over data streams. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 373–380. ACM, 2004.

22 Piotr Indyk, Sepideh Mahabadi, Shayan Oveis Gharan, and Alireza Rezaei. Composable core-sets for determinant maximization problems via spectral spanners. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1675–1694. SIAM, 2020.

23 Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab S. Mirrokni. Composable core-sets for diversity and coverage maximization. In Richard Hull and Martin Grohe, editors, *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 100–108. ACM, 2014.

24 Anoop Jain, Parag Sarda, and Jayant R Haritsa. Providing diversity in k-nearest neighbor query results. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 404–413. Springer, 2004.

25 Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 510–520, 2011.

26 Hui Lin, Jeff Bilmes, and Shasha Xie. Graph-based submodular selection for extractive summarization. In *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 381–386. IEEE, 2009.

27 Sepideh Mahabadi, Piotr Indyk, Shayan Oveis Gharan, and Alireza Rezaei. Composable core-sets for determinant maximization: A simple near-optimal algorithm. In *International Conference on Machine Learning*, pages 4254–4263. PMLR, 2019.

28 Vahab Mirrokni and Morteza Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 153–162. ACM, 2015.

29 Julien Pilourdault, Sihem Amer-Yahia, Dongwon Lee, and Senjuti Basu Roy. Motivation-aware task assignment in crowdsourcing. In *EDBT*, 2017.

30 S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi. Facility dispersion problems: Heuristics and special cases. *Algorithms and Data Structures*, 519:355–366, 1991.

31 Michael J Welch, Junghoo Cho, and Christopher Olston. Search result diversity for informational queries. In *Proceedings of the 20th international conference on World wide web*, pages 237–246, 2011.

32 Cong Yu, Laks VS Lakshmanan, and Sihem Amer-Yahia. Recommendation diversification using explanations. In *2009 IEEE 25th International Conference on Data Engineering*, pages 1299–1302. IEEE, 2009.

33 Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107(10):4511–4515, 2010.

34 Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32, 2005.

A Composable Coreset for Remote-Pseudoforest

In this section, we describe and analyze the composable coreset algorithm for remote-pseudoforest.

A.1 Algorithm

In this subsection, we prove why the algorithm given in Algorithm 2 creates an $O(1)$-approximate composable coreset. First, we describe the algorithm in words. We recall that we have m datasets $X^{(1)}, \ldots, X^{(m)}$: we wish to create a coreset $C^{(j)}$ of each $X^{(j)}$ so that $\bigcup_{j=1}^{m} C^{(j)}$ contains a set Z of k points such that $\mathrm{PF}(Z) \geq \Omega(1) \cdot \mathrm{PF}_k\left(\bigcup_{j=1}^{m} X^{(j)}\right)$.

A.1.1 Coreset Construction

Suppose that $|X^{(j)}| \geq 2k^{1+\varepsilon} + k$. Let $\tilde{r}^{(j)}$ represent the smallest value such that there exists a ball $\tilde{B}^{(j)}$ of radius $\tilde{r}^{(j)}$ around some $x^{(j)} \in X^{(j)}$ that contains all but at most k of the points in $X^{(j)}$. The point $\tilde{r}^{(j)}$ and a corresponding $x^{(j)}, \tilde{B}^{(j)}$ can be found in $O(|X^{(j)}|^2)$ time.

Choose $U^{(j)}$ to be the set of k points furthest from $x^{(j)}$. These will either be precisely the k points outside $\tilde{B}^{(j)}$, or all of the points outside $\tilde{B}^{(j)}$ plus some points on the boundary of $\tilde{B}^{(j)}$, to make a total of k points. In addition, we choose some arbitrary set $P^{(j)} \subset X^{(j)}$ of any k points in the ball $B^{(j)}$.

Next, we will choose sets $S^{(j)}, T^{(j)} \subset X^{(j)}$ of size k, such that $\rho(S^{(j)}, T^{(j)}) \geq \frac{\varepsilon}{2} \cdot \tilde{r}^{(j)}$. It is not even clear that such sets exist, but we will show how to algorithmically find such sets in $O(|X^{(j)}|^2)$ time (assuming $|X^{(j)}| \geq 2k^{1+\varepsilon} + k$).

Finally, we run the GMM algorithm on $X^{(j)}$ to obtain $Y^{(j)} = \{y_1^{(j)}, y_2^{(j)}, \ldots, y_k^{(j)}\}$. The final coreset will be $C^{(j)} := P^{(j)} \cup S^{(j)} \cup T^{(j)} \cup U^{(j)} \cup Y^{(j)}$. Note that each of $P^{(j)}, S^{(j)}, T^{(j)}, U^{(j)}, Y^{(j)}$ has size at most k, so $|C^{(j)}| \leq 5k$.

Alternatively, if $|X^{(j)}| < 2k^{1+\varepsilon} + k$, we choose the coreset to simply be $X^{(j)}$. For convenience, in this setting, we define $U^{(j)} := X^{(j)}$ and $P^{(j)}, S^{(j)}, T^{(j)}, Y^{(j)}$ to all be empty.

We have not yet described how to find $S^{(j)}, T^{(j)}$, let alone prove they even exist. We now describe an $O(|X^{(j)}|^2)$ time algorithm that finds $S^{(j)}, T^{(j)} \subset X^{(j)}$ of size k, such that $\rho(S^{(j)}, T^{(j)}) \geq \frac{\varepsilon}{2} \cdot \tilde{r}^{(j)}$.

A.1.2 Efficiently finding $S^{(j)}, T^{(j)}$

Here, we show that we can efficiently find $S^{(j)}, T^{(j)}$ from the coreset construction, as long as $|X^{(j)}| \geq 2k^{1+\varepsilon} + k$. We formalize this with the following lemma.

▶ **Lemma 16.** *Let $\varepsilon > 0$ be a fixed constant, and consider a dataset X of size at least $2k^{1+\varepsilon} + k$, and suppose that no ball of radius smaller than $\tilde{r}$ around any $x \in X$ contains all but at most k points in X. (In other words, for every $x \in X$, there are at least k points in X of distance at least $\tilde{r}$ from x.) Then, we in $O(|X|^2)$ time, we can find two disjoint sets $S, T \subset X$, each of size k, such that $\rho(S, T) \geq \frac{\varepsilon \cdot \tilde{r}}{2}$.*

Proof. The algorithm works as follows. First, define $r' = \frac{\varepsilon \cdot \tilde{r}}{2}$. For each point $x \in X$, and for every nonnegative integer i, define $N_i(x)$ as the set of points in X of distance at most $i \cdot r'$ from x. We can compute the set $N_i(x)$ for all $x \in X$ and $0 \leq i \leq 2/\varepsilon$ in $O(|X|^2)$ time, as ε is a constant. Suppose there exists $x \in X$ such that $|N_{1/\varepsilon}(x)| \geq k$. By our assumption on $\tilde{r}$, there are at least k points of distance at least $\tilde{r} = \frac{2}{\varepsilon} \cdot r'$ from x (or else we could have chosen $\tilde{r}$ to be smaller). Therefore, we can let S be a subset of size k from $N_{1/\varepsilon}(x)$ and T be a subset of size k of points of distance at least $\frac{2}{\varepsilon} \cdot r'$ from x. The minimum distance between any $s \in S$ and $t \in T$ is at least $r' \cdot \left(\frac{2}{\varepsilon} - \frac{1}{\varepsilon}\right) = \frac{\tilde{r}}{2}$.

Alternatively, every $x \in X$ satisfies $|N_{1/\varepsilon}(x)| < k$. Now, consider the following peeling procedure. Let $X_0 := X$: for each $h \geq 1$, we will inductively create $X_h \subsetneq X_{h-1}$ from X_{h-1}, as follows. First, we pick an arbitrary point $x_h \in X_{h-1}$. For any point $x \in X_{h-1}$ and any integer $i \geq 0$, define $N_i(x_h; X_{h-1}) = N_i(x_h) \cap X_{h-1}$ to be the set of points of distance at most $i \cdot r'$ from x_h in X_{h-1}. By our assumption, we have that $|N_{1/\varepsilon}(x_h; X_{h-1})| \leq |N_{1/\varepsilon}(x_h)| < k$, so there exists some $i(h)$ with $0 \leq i(h) \leq \frac{1}{\varepsilon}$, such that $\frac{|N_{i(h)+1}(x_h, X_{h-1})|}{|N_{i(h)}(x_h, X_{h-1})|} \leq k^{\varepsilon}$ and $|N_{i(h)}(x_h, X_{h-1})| \leq k$. Choose such an $i = i(h)$, and let $X_h := X_{h-1} \backslash N_{i(h)}(x_h, X_{h-1})$.

We repeat this process until we have found the first X_ℓ with $|X \backslash X_\ell| \geq k$. Note that each removal process removes at least 1 and at most k elements, so $|X \backslash X_\ell| \leq 2k$. Let $S_h = N_{i(h)}(x_h, X_{h-1}) = X_{h-1} \backslash X_h$ for each $1 \leq h \leq \ell$, so $X \backslash X_\ell = S_1 \cup \cdots \cup S_\ell$. Note, however, that any point within distance r' of some $x \in S_h$ was either in $S_1 \cup \cdots \cup S_{h-1}$, or was in $N_{i(h)+1}(x_h, X_{h-1})$. In other words, every point of distance r' of $x \in S_1 \cup \cdots \cup S_\ell$ is in $\bigcup_{h=1}^{\ell} N_{i(h)+1}(x_h, X_{h-1})$. But this has size at most

$$\sum_{i=1}^{\ell} k^{\varepsilon} \cdot |N_{i(h)}(x_h, X_{h-1})| = k^{\varepsilon} \cdot \sum_{i=1}^{\ell} |S_i| \leq k^{\varepsilon} \cdot 2k = 2k^{1+\varepsilon}.$$

So, assuming that $|X| \geq 2k^{1+\varepsilon} + k$, defining $S := S_1 \cup \cdots \cup S_\ell$, we have that $|S| \geq k$ and there are at least k points in X that are *not* within distance $r' = \frac{\varepsilon \cdot \tilde{r}}{2}$ of S. ◀

We include pseudocode for the algorithm described in the proof of Lemma 16, in Algorithm 5.

A.2 Analysis

In this section, we prove that the algorithm indeed generates an $O(1/\varepsilon)$-approximate composable coreset of size at most $O(k^{1+\varepsilon})$. By making ε an arbitrarily small constant, this implies we can find a constant-approximate composable coreset of size $O(k^{1+\varepsilon})$ for any arbitrarily small constant ε.

Let OPT represent the optimal set of k points in all of $X = \bigcup_{j=1}^{m} X^{(j)}$, that maximizes remote-pseudoforest cost. Our goal is to show that there exists a set of k points in $\bigcup_j (P^{(j)} \cup S^{(j)} \cup T^{(j)} \cup U^{(j)} \cup Y^{(j)})$ with pseudoforest cost at least $\Omega(\text{PF}(\text{OPT}))$.

Let $r^{(j)}$ represent the maximum distance from any point in $X^{(j)}$ to its closest point in $Y^{(j)}$. Note that by Proposition 8, $\text{PF}(Y^{(j)}) \geq k \cdot r^{(j)}$. Hence, if $\text{PF}(\text{OPT}) < 10 \cdot k \cdot \max_j r^{(j)}$, there exists some choice of j with $\text{PF}(Y^{(j)}) \geq 0.1 \cdot \text{OPT}$ and $|Y^{(j)}| = k$. Hence, we get a constant-factor approximation in this case. Otherwise, we may assume that $\text{PF}(\text{OPT}) \geq 10 \cdot k \cdot \max_j r^{(j)}$.

Next, for a fixed $X^{(j)}$, let $X_i^{(j)}$ represent the set of points in $X^{(j)}$ closest to $y_i^{(j)}$ among all points in $Y^{(j)}$. Given the optimal solution OPT of k points, let $\text{OPT}_i^{(j)} = \text{OPT} \cap X_i^{(j)}$.

Algorithm 5 FindST: Find two sets S, T of size k with large $\rho(S,T)$.

Input: data $X = \{x_1, \ldots, x_n\}$, integer k, parameter $\varepsilon \in (0,1]$, radius $\tilde{r}$.
$r' \leftarrow \frac{\varepsilon \cdot \tilde{r}}{2}$.
for x in X **do**
  for $i = 0$ to $2/\varepsilon$ **do**
    $N_i(x) = \{z \in X : \rho(x,z) \leq i \cdot r'\}$.
  end for
  if $|N_{1/\varepsilon}(x)| \geq k$ **then**
    $S \leftarrow$ arbitrary subset of size k in $N_{1/\varepsilon}(x)$.
    $T \leftarrow$ arbitrary k points of distance at least $\tilde{r}$ from x.
    Return S, T.
  end if
end for
$S = \emptyset, X_0 \leftarrow X, h \leftarrow 0$
while $|S| < k$ **do**
  $h \leftarrow h + 1$
  $x_h \in X_{h-1}$ chosen arbitrarily.
  Find $0 \leq i \leq \frac{1}{\varepsilon}$ such that $\frac{|N_{i+1}(x_h) \cap X_{h-1}|}{|N_i(x_h) \cap X_{h-1}|} \leq k^{\varepsilon}$.
  $S_h \leftarrow N_i(x_h) \cap X_{h-1}$.
  $X_h \leftarrow X_{h-1} \backslash S_h$, $S \leftarrow S \cup S_h$
end while
$S \leftarrow$ arbitrary subset of size k in S
$T \leftarrow$ arbitrary subset of k points of distance at least r' from all points in S.
Return S, T.

Now, we will define sets $G_i^{(j)}, {G'_i}^{(j)}$ based on the following cases.

1. If $|X^{(j)}| < 2k^{1+\varepsilon} + k$, define $G_i^{(j)} = {G'_i}^{(j)} = \mathrm{OPT}_i^{(j)}$ for all $i \leq k$.
2. Else, if $y_i^{(j)} \in \mathrm{OPT}_i^{(j)}$, define $G_i^{(j)}$ as $y_i^{(j)} \cup (U^{(j)} \cap \mathrm{OPT}_i^{(j)})$ and ${G'_i}^{(j)} = \mathrm{OPT}_i^{(j)}$.
3. Else, if $\mathrm{OPT}_i^{(j)} \backslash U^{(j)} = \emptyset$ (i.e., all points in $\mathrm{OPT}_i^{(j)}$ happen to be in $U^{(j)}$), define $G_i^{(j)} = {G'_i}^{(j)} = \mathrm{OPT}_i^{(j)}$.
4. Else, define $G_i^{(j)} = y_i^{(j)} \cup (U^{(j)} \cap \mathrm{OPT}_i^{(j)})$, and define ${G'_i}^{(j)}$ as $\mathrm{OPT}_i^{(j)}$ with the slight modification of moving a single (arbitrary) point in $\mathrm{OPT}_i^{(j)} \backslash U^{(j)}$ to $y_i^{(j)}$.

We will define the sets $G = \bigcup_{i,j} G_i^{(j)}$ and $G' = \bigcup_{i,j} {G'_i}^{(j)}$.

Importantly, the following five properties always hold for all $i \leq m, j \leq k$. (They even hold in the setting when $|X^{(j)}| < 2k^{1+\varepsilon} + k$, because we defined $U^{(j)} = X^{(j)}$ and $G_i^{(j)} = {G'_i}^{(j)} = \mathrm{OPT}_i^{(j)}$.)

1. $|{G'_i}^{(j)}| = |\,\mathrm{OPT}_i^{(j)}\,|$. This means that $|G'| = k$.
2. $G_i^{(j)} \subset {G'_i}^{(j)} \subset X_i^{(j)}$. This means that $G \subset G'$.
3. $G_i^{(j)} \subset U^{(j)} \cup Y^{(j)}$. This means that $G \subset \bigcup_j (U^{(j)} \cup Y^{(j)})$.
4. Every point in ${G'_i}^{(j)} \backslash G_i^{(j)}$ is not in $U^{(j)}$. This means that $\bigcup U^{(j)}$ and $G' \backslash G$ are disjoint.
5. If ${G'_i}^{(j)} \backslash G_i^{(j)}$ is nonempty, then $y_i^{(j)} \in G_i^{(j)}$, so $G_i^{(j)}$ is also nonempty.

Now, note that from changing OPT to G', we never move a point by more than $\max_j r^{(j)}$, which means that $|\,\mathrm{PF}(G') - \mathrm{PF}(\mathrm{OPT})| \leq 2k \cdot \max_j r^{(j)}$. As we are assuming that $\mathrm{PF}(\mathrm{OPT}) \geq 10k \cdot \max_j r^{(j)}$, we have $\mathrm{PF}(G') \geq 0.8 \cdot \mathrm{OPT}$. Next, if a point x is in ${G'_i}^{(j)}$ but not in $G_i^{(j)}$, then

$x \in X^{(j)} \backslash U^{(j)}$ and $y_i^{(j)} \in G_i^{(j)}$, which means that the cost of x with respect to G' is at most $\max_j r^{(j)}$. So for any set A, if we define $\text{cost}_A(x)$ for $x \in A$ to denote $\min_{y \in A: y \neq x} \rho(x, y)$, then

$$\sum_{x \in G} \text{cost}_{G'}(x) \geq \text{PF}(G') - \sum_{x \in G' \backslash G} \text{cost}_{G'}(x) \geq \text{PF}(G') - k \cdot \max_j r^{(j)} \geq 0.7 \cdot \text{OPT}\,. \tag{8}$$

We now try to find a set $G'' \supset G$ of size k with large pseudoforest cost, but this time we must ensure that $G'' \subset \bigcup_j (P^{(j)} \cup S^{(j)} \cup T^{(j)} \cup U^{(j)} \cup Y^{(j)})$. In other words, to finish the analysis, it suffices to prove the following lemma.

▶ **Lemma 17.** *There exists $G'' \subset \bigcup_j (P^{(j)} \cup S^{(j)} \cup T^{(j)} \cup U^{(j)} \cup Y^{(j)})$ of size at least k, such that $G'' \supset G$ and $\sum_{x \in G} \text{cost}_{G''}(x) \geq \Omega(\varepsilon) \cdot \text{PF}(\text{OPT})$.*

To see why Lemma 17 is sufficient to prove Theorem 4, since $|G| \leq k$ and $|G''| \geq k$ we can choose a set $\hat{G}$ of size k such that $G \subseteq \hat{G} \subseteq G''$. Then, $\hat{G} \subset \bigcup_j (P^{(j)} \cup S^{(j)} \cup T^{(j)} \cup U^{(j)} \cup Y^{(j)})$ and

$$\text{PF}(\hat{G}) = \sum_{x \in \hat{G}} \text{cost}_{\hat{G}}(x) \geq \sum_{x \in G} \text{cost}_{\hat{G}}(x) \geq \sum_{x \in G} \text{cost}_{G''}(x),$$

where the last inequality holds because the cost of x never increases from $\hat{G}$ to a larger set G''. Finally, by Lemma 17, this means $\text{PF}(\hat{G}) \geq \Omega(\varepsilon) \cdot \text{PF}(\text{OPT})$, as desired.

We now prove Lemma 17. First, we show how to construct G''. Let $g = |G|$: note that $g \leq k$. If $g = k$, then in fact $G = G'$ and we can set $G'' = G$, which completes the proof by (8) and Property 3.

Hence, from now on, we may assume that $g < k$. Recall that $X^{(j)} \backslash U^{(j)}$ is contained in a ball of radius $\tilde{r}^{(j)}$. Next, let $\tilde{r}$ be the radius of $\bigcup_j (X^{(j)} \backslash U^{(j)})$. (Note that for $|X^{(j)}| < 2k^{1+\varepsilon} + k$, $X^{(j)} \backslash U^{(j)}$ is empty.) We claim the following proposition.

▶ **Proposition 18.** *There exist $j, j' \leq m$, possibly equal, such that $\rho(S^{(j)}, T^{(j')}) \geq \frac{\varepsilon}{10} \cdot \tilde{r}$.*

Proof. Let $A \subset [m]$ be the subset of indices j such that $|X^{(j)}| \geq 2k^{1+\varepsilon} + k$. Suppose that $\tilde{r} \leq 5 \cdot \max_{j \in A} \tilde{r}^{(j)}$. Then, by setting $j = j'$ to be $\arg\max_{j \subset A} \tilde{r}^{(j)}$, we have that $\rho(S^{(j)}, T^{(j')}) \geq \frac{\varepsilon}{2} \cdot \max_{j \in A} \tilde{r}^{(j)} \geq \frac{\varepsilon}{10} \cdot \tilde{r}$.

Otherwise, $\tilde{r} > 5 \cdot \max_{j \in A} \tilde{r}^{(j)}$. So, if we pick j arbitrarily, the distance between the center of the ball $\tilde{B}^j$ and the furthest center $\tilde{B}^{j'}$ must be at least $0.8 \cdot \tilde{r}$, or else the ball of radius $0.8 \cdot \tilde{r} + \max_{j \in A} \tilde{r}^{(j)} < \tilde{r}$ around the center of $\tilde{B}^j$ contains all of $\bigcup_j (X^{(j)} \backslash U^{(j)})$. Then, $d(S^{(j)}, T^{(j')}) \geq 0.8 \cdot \tilde{r} - \tilde{r}^{(j)} - \tilde{r}^{(j')} \geq 0.4 \cdot \tilde{r}$, which is at least $\frac{\varepsilon}{10} \cdot \tilde{r}$. ◀

We now prove Lemma 17.

Proof. Recall that we already proved the lemma in the case that $G = G'$. So, we may assume $|G| < k$ and $G' \backslash G$ is nonempty. We claim that we can set G'' to be one of $G \cup P^{(j)}$, $G \cup S^{(j)}$, or $G \cup T^{(j')}$, for j, j' chosen in Proposition 18.

First, note that $P^{(j)}, S^{(j)}$, and $T^{(j')}$ have size k, so all three choices of G'' have size at least k.

First, note that $G' \backslash G$ is assumed to be nonempty, which means it is contained in $\bigcup_j (X^{(j)} \backslash U^{(j)})$ by Property 4, which is contained in the radius $\tilde{r}$ ball. Therefore, $G' \backslash G$ has nonempty intersection with $X \backslash (\bigcup_j U^{(j)})$. Now, fix any point $x \in G$. If $\text{cost}_{G'}(x) \geq 3 \cdot \tilde{r}$, then because $G' \backslash G$ has a point in the radius $\tilde{r}$ ball containing $X \backslash (\bigcup_j U^{(j)})$ (and this point

is not x since $x \in G$), x has distance at least $\tilde{r}$ from the radius $\tilde{r}$ ball. So, $\text{cost}_{G \cup P^{(j)}}(x) \geq \text{cost}_{G'}(x) - 2\tilde{r} \geq \frac{1}{3} \cdot \text{cost}_{G'}(x)$. Alternatively, if $\text{cost}_{G'}(x) < 3 \cdot \tilde{r}$, then $\text{cost}_{G \cup S^{(j)}}(x) \geq \min(\text{cost}_{G'}(x), \rho(x, S^{(j)}))$ and likewise, $\text{cost}_{G \cup T^{(j')}}(x) \geq \min(\text{cost}_{G'}(x), \rho(x, T^{(j')}))$. So,

$$\begin{aligned}\text{cost}_{G \cup S^{(j)}}(x) + \text{cost}_{G \cup T^{(j')}}(x) &\geq \min(\text{cost}_{G'}(x), \rho(S^{(j)}, T^{(j')})) \\ &\geq \min\left(\text{cost}_{G'}(x), \frac{\varepsilon}{10} \cdot \tilde{r}\right) \geq \frac{\varepsilon}{30} \cdot \text{cost}_{G'}(x).\end{aligned}$$

In all cases, we have that

$$\text{cost}_{G \cup P^{(j)}}(x) + \text{cost}_{G \cup S^{(j)}}(x) + \text{cost}_{G \cup T^{(j')}}(x) \geq \frac{\varepsilon}{30} \cdot \text{cost}_{G'}(x),$$

so adding over all $x \in G$ and choosing among the three choices randomly, we have that the total cost in expectation is at least

$$\frac{1}{3} \cdot \left(\sum_{x \in G} \frac{\varepsilon}{30} \cdot \text{cost}_{G'}(x)\right) = \frac{\varepsilon}{90} \cdot \sum_{x \in G} \text{cost}_{G'}(x) \geq \frac{\varepsilon}{90} \cdot 0.7 \cdot \text{PF}(\text{OPT}) \geq \frac{\varepsilon}{150} \cdot \text{PF}(\text{OPT}).$$

Hence, for one of the three choices of G'', the pseudoforest cost is at least $\frac{\varepsilon}{150} \cdot \text{PF}(\text{OPT})$. ◀