Diverse near neighbor problem

Diverse Near Neighbor Problem

[Extended Abstract]

Sofiane Abbar
QCRI, Doha, Qatar
sabbar@qf.org.qa

Sihem Amer-Yahia
CNRS, LIG, Paris, France
Yahia@imag.fr

Piotr Indyk
MIT, Cambridge, MA, USA
indyk@mit.edu

Sepideh Mahabadi
MIT, Cambridge, MA, USA
mahabadi@mit.edu

Kasturi R. Varadarajan
U Iowa, Iowa City, IA, USA
kvaradar@cs.uiowa.edu

ABSTRACT

Motivated by the recent research on diversity-aware search, we investigate the k-diverse near neighbor reporting problem. The problem is defined as follows: given a query point q, report the *maximum diversity* set S of k points in the ball of radius r around q. The diversity of a set S is measured by the minimum distance between any pair of points in S (the higher, the better).

We present two approximation algorithms for the case where the points live in a d-dimensional Hamming space. Our algorithms guarantee query times that are sub-linear in n and only polynomial in the diversity parameter k, as well as the dimension d. For low values of k, our algorithms achieve sub-linear query times even if the number of points within distance r from a query q is linear in n. To the best of our knowledge, these are the first known algorithms of this type that offer provable guarantees.

Categories and Subject Descriptors

F.2.2 [**Nonnumerical Algorithms and Problems**]: Geometrical problems and computations; E.2 [**Data Storage Representations**]: Hash-table representations

Keywords

Near Neighbor, Diversity, Core-set, Sub-linear

1. INTRODUCTION

The near neighbor reporting problem (a.k.a. range query) is defined as follows: given a collection P of n points, build a data structure which, given any query point, reports all data points that are within a given distance r to the query. The problem is of major importance in several areas, such as databases and data mining, information retrieval, image and video databases, pattern recognition, statistics and data analysis. In those application, the features of each object of interest (document, image, etc) are typically represented as a point in a d-dimensional space and the distance metric is used to measure similarity of objects. The basic problem then is to perform indexing or similarity searching for query objects. The number of features (i.e., the dimensionality) ranges anywhere from tens to thousands.

One of the major issues in similarity search is how many answers to retrieve and report. If the size of the answer set is too small (e.g., it includes only the few points closest to the query), the answers might be too homogeneous and not informative [6]. If the number of reported points is too large, the time needed to retrieve them is high. Moreover, long answers are typically not very informative either. Over the last few years, this concern has motivated a significant amount of research on diversity-aware search [8, 18, 5, 12, 17, 16, 7] (see [6] for an overview). The goal of that work is to design efficient algorithms for retrieving answers that are both *relevant* (e.g., close to the query point) and *diverse*. The latter notion can be defined in several ways. One of the popular approaches is to cluster the answers and return only the cluster centers [6, 8, 14, 1]. This approach however can result in high running times if the number of relevant points is large.

Our results.

In this paper we present two efficient approximate algorithms for the *k-diverse near neighbor* problem. The problem is defined as follows: given a query point, report the *maximum diversity* set S of k points in the ball of radius r around q. The diversity of a set S is measured by the minimum distance between any pair of points in S. In other words, the algorithm reports the approximate solution to the k-center clustering algorithm applied to the list points that are close to the query. The running times, approximation factors and the space bounds of our algorithms are given in Table 1. Note that the Algorithm A is dominated by Algorithm B; however, it is simpler and easier to analyze and implement, and has been used in applications before for diverse news retrieval [1].

The key feature of our algorithms is that they guarantee query times that are sub-linear in n and polynomial in the diversity parameter k and the dimension d, while at the same time providing constant factor approximation guarantees[1] for the diversity objective. Note that for low values of k our

SoCG'13, June 17-20, 2013, Rio de Janeiro, Brazil.

[1]Note that approximating the diversity objective is inevitable, since it is NP-hard to find a subset of size k which

	Algorithm A	Algorithm B
Distance approx factor	$c > 2$	$c > 1$
Diversity approx factor	6	6
Space	$O((n \log k)^{1+\frac{1}{c-1}} + nd)$	$O(\log k \cdot n^{1+\frac{1}{c}} + nd)$
Query Time	$O((k^2 + \frac{\log n}{r})d \cdot (\log k)^{\frac{c}{c-1}} \cdot n^{\frac{1}{c-1}})$	$O((k^2 + \frac{\log n}{r})d \cdot \log k \cdot n^{1/c})$

Table 1: Performance of our algorithms

algorithms have sub-linear query times *even if* the number of points within distance r from q is linear in n. To the best of our knowledge, these are the first known algorithms of this type with provable guarantees. One of the algorithms (Algorithm A) is closely related to algorithms investigated in applied works [1, 14]. However, those papers did not provide rigorous guarantees on the answer quality.

1.1 Past work

In this section we present an overview of past work on (approximate) near neighbor and diversity aware search that are related to the results in this paper.

Near neighbor.

The near neighbor problem has been a subject of extensive research. There are several efficient algorithms known for the case when the dimension d is "low". However, despite decades of intensive effort, the current solutions suffer from either space or query time that is exponential in d. Thus, in recent years, several researchers proposed methods for overcoming the running time bottleneck by using approximation. In the approximate near neighbor reporting/range query, the algorithm must output *all* points within the distance r from q, and can also output *some* points within the distance cr from q.

One of the popular approaches to near neighbor problems in high dimensions is based on the concept of *locality-sensitive hashing* (LSH) [10]. The idea is to hash the points using several (say L) hash functions so as to ensure that, for each function, the probability of collision is much higher for objects which are close to each other than for those which are far apart. Then, one can solve (approximate) near neighbor reporting by hashing the query point and retrieving all elements stored in buckets containing that point. This approach has been used e.g., for the E2LSH package for high-dimensional similarity search [4].

The LSH algorithm has several variants, depending on the underlying distance functions. In the simplest case when the dis-similarity between the query points is defined by the Hamming distance, the algorithm guarantees that (i) each point within the distance r from from q is reported with a constant (tunable) probability and (ii) the query time is at most $O(d(n^{1/c} + |P_{cr}(q)|))$, where $P_R(q)$ denotes the set of points in P with the distance R from q. Thus, if the size of the answer set $P_{cr}(q)$ is large, the efficiency of the algorithm decreases. Heuristically, a speedup can be achieved [14] by clustering the points in each bucket and retaining only the cluster centers. However, the resulting algorithm did not have any guarantees (until now).

maximizes the diversity with approximation factor $a < 2$ [13].

Diversity.

In this paper we adopt the "content-based" definition of diversity used e.g., in [6, 8, 14, 1, 7]. The approach is to report k answers that are "sufficiently different" from each other. This is formalized as maximizing the minimum distance between any pair of answers, the average distance between the answers, etc. In this paper we use the minimum distance formulation, and use the greedy clustering algorithm of [9, 13] to find the k approximately most diverse points in a given set.

To the best of our knowledge, the only prior work that explicitly addresses our definition of the k-diverse near neighbor problem is [1]. It presents an algorithm (analogous to Algorithm A in this paper, albeit for the Jaccard coefficient as opposed to the Hamming metric) and applies it to problems in news retrieval. However, that paper does not provide any formal guarantees on the accuracy of the reported answers.

1.2 Our techniques

Both of our algorithms use LSH as the basis. The key challenge, however, is in reducing the dependence of the query time on the size of the set $P_{cr}(q)$ of points close to q. The first algorithm (Algorithm A) achieves this by storing only the k most diverse points per each bucket. This ensures that the total number of points examined during the query time is at most $O(kL)$, where L is the number of hash functions. However, proving the approximation guarantees for this algorithm requires that no outlier (i.e., point with distance $> cr$ from q) is stored in any bucket. Otherwise that point could have been selected as one of the k diverse points for that bucket, replacing a "legitimate" point. This requirement implies that the algorithm works only if the distance approximation factor c is greater than 2.

The 6-approximation guarantee for diversity is shown by using the notion of *coresets* [2]. It is easy to see that the maximum k-diversity of a point set is within a factor of 2 from the optimal cost of its $(k-1)$-center clustering cost. For the latter problem it is known how to construct a small subset of the input point set (a *coreset*) such that for any set of cluster centers, the costs of clustering the coreset is within a constant factor away from the cost of clustering the whole data set. Our algorithm then simply computes and stores only a coreset for each LSH bucket. Standard coreset properties imply that the union of coresets for the buckets touched by the query point q is a coreset for all points in those buckets. Thus, the union of all coresets provides a sufficient information to recover an approximately optimal solution to all points close to q.

In order to obtain an algorithm that works for arbitrary $c > 1$, we need the algorithms to be robust to outliers. The standard LSH analysis guarantees that that the number of outliers in all buckets is at most $O(L)$. Algorithm B achieves the robustness by storing a *robust* coreset [11, 3], which can

tolerate some number of outliers. Since we do not know a priori how many outliers are present in any given bucket, our algorithm stores a sequence of points that represents a coreset robust to an increasing number of outliers. During the query time the algorithm scans the list until enough points have been read to ensure the robustness.

2. PROBLEM DEFINITION

Let $(\Delta, dist)$ be a d-dimensional metric space. We start from two definitions.

Definition 1. For a given set $S \in \Delta$, its **diversity** is defined as the minimum pairwise distance between the points of the set, i.e., $div(S) = \min_{p,p' \in S} dist(p, p')$

Definition 2. For a given set $S \in \Delta$, its k**-diversity** is defined as the maximum achievable diversity by choosing a subset of size k, i.e., $div_k(S) = \max_{S' \subset S, |S'|=k} div(S')$. We also call the maximizing subset S' the **optimal** k**-subset** of S. Note that k-diversity is not defined in the case where $|S| < k$.

To avoid dealing with k-diversity of sets of cardinality smaller than k, in the following we adopt the convention that all points p in the input point set P are duplicated k times. This ensures that for all non-empty sets S considered in the rest of this paper the quantity $div_k(S)$ is well defined, and equal to 0 if the number of distinct points in S is less than k. It can be easily seen that this leaves the space bounds of our algorithms unchanged.

The k**-diverse Near Neighbor Problem** is defined as follows: given a query point q, report a set S such that: (i) $S \subset P \cap B(q, r)$, where $B(q, r) = \{p | dist(p, q) \leq r\}$ is the ball of radius r, centered at q; (ii) $|S| = k$; (iii) $div(S)$ is maximized.

Since our algorithms are approximate, we need to define the **Approximate** k**-diverse Near Neighbor Problem**. In this case, we require that for some approximation factors $c > 1$ and $\alpha > 1$: (i) $S \subset P \cap B(q, cr)$; (ii) $|S| = k$; (iii) $div(S) \geq \frac{1}{\alpha} div_k(P \cap B(q, r))$.

3. PRELIMINARIES

3.1 GMM Algorithm

Suppose that we have a set of points $S \subset \Delta$, and want to compute an optimal k-subset of S. That is, to find a subset of k points, whose pairwise distance is maximized. Although this problem is NP-hard, there is a simple 2-approximate greedy algorithm [9, 13], called GMM.

In this paper we use the following slight variation of the GMM algorithm [2]. The algorithm is given a set of points S, and the parameter k as the input. Initially, it chooses some arbitrary point $a \in S$. Then it repeatedly adds the next point to the output set until there are k points. More precisely, in each step, it greedily adds the point whose minimum distance to the currently chosen points is maximized. Note that the convention that all points have k duplicates implies that if the input point set S contains less than k distinct points, then the output S' contains all of those points.

[2]The proof of the approximation factor this variation achieves is virtually the same as the proof in [13]

Algorithm 1 GMM

Input S: a set of points, k: size of the subset
Output S': a subset of S of size k.

```
1: S' ← An arbitrary point a
2: for i = 2 → k do
3:    find p ∈ S \ S' which maximizes min_{x∈S'} dist(p, x)
4:    S' ← S' ∪ {p}
5: end for
6: return S'
```

LEMMA 1. *The running time of the algorithm is $O(k \cdot |S|)$, and it achieves an approximation factor of at most 2 for the k-diversity $div_k(S)$.*

3.2 Coresets

Definition 3. Let $(P, dist)$ be a metric. For any subset of points $S, S' \subset P$, we define the k-center cost, $KC(S, S')$ as $\max_{p \in S} min_{p' \in S'} dist(p, p')$. The **Metric** k**-center Problem** is defined as follows: given S, find a subset $S' \subset S$ of size k which minimizes $KC(S, S')$. We denote this optimum cost by $KC_k(S)$.

k-diversity of a set S is closely related to the cost of the best $(k-1)$-center of S. That is,

LEMMA 2. $KC_{k-1}(S) \leq div_k(S) \leq 2KC_{k-1}(S)$

PROOF. For the first inequality, suppose that S' is the optimal k-subset of S. Also let $a \in S'$ be an arbitrary point and $S'_- = S' \setminus \{a\}$. Then for any point $b \in S \setminus S'$, we have $min_{p \in S'_-} dist(b, p) \leq min_{p \in S'_-} dist(a, p)$, otherwise b was a better choice than a, i.e., $div(b \cup S'_-) > div(S')$. Therefore, $KC(S, S'_-) \leq div_k(S)$ and the inequality follows.

For the second part, let $C = \{a_1, \cdots, a_{k-1}\}$ be the optimum set of the $(k-1)$-center for S. Then since S' has size k, by pigeonhole principle, there exists $p, p' \in S'$ and a, such that

$$a = \arg\min_{c \in C} dist(p, c) = \arg\min_{c \in C} dist(p', c)$$

and therefore, by triangle inequality we get

$$\begin{aligned} div_k(S) = div(S') \leq dist(p, p') &\leq dist(p, a) + dist(a, p') \\ &\leq 2KC_{k-1}(S) \end{aligned}$$

□

Definition 4. Let $(P, dist)$ be our metric. Then for $\beta \leq 1$, we define a β**-coreset** for a point set $S \subset P$ to be any subset $S' \subset S$ such that for any subset of $(k-1)$ points $F \subset P$, we have $KC(S', F) \geq \beta KC(S, F)$.

Definition 5. Let $(P, dist)$ be our metric. Then for $\beta \leq 1$ and an integer ℓ, we define an ℓ**-robust** β**-coreset** for a point set $S \subset P$ to be any subset $S' \subset S$ such that for any set of outliers $O \subset P$ with at most ℓ points, $S' \setminus O$ is a β-coreset of $S \setminus O$.

3.3 Locality Sensitive Hashing

Locality-sensitive hashing is a technique for solving approximate near neighbor problems. The basic idea is to

hash the data and query points in a way that the probability of collision is much higher for points that are close to each other, than for those which are far apart. Formally, we require the following.

Definition 6. A family $\mathcal{H} = h : \Delta \to U$ is (r_1, r_2, p_1, p_2)-sensitive for $(\Delta, dist)$, if for any $p, q \in \Delta$, we have

- if $dist(p, q) \leq r_1$, then $Pr_{\mathcal{H}}[h(q) = h(p)] \geq p_1$
- if $dist(p, q) \leq r_2$, then $Pr_{\mathcal{H}}[h(q) = h(p)] \leq p_2$

In order for a locality sensitive family to be useful, it has to satisfy inequalities $p_1 > p_2$ and $r_1 < r_2$.

Given an LSH family, the algorithm creates L hash functions $g_1, g_2, \cdots, g_L$, as well as the corresponding hash arrays $A_1, A_2, \cdots, A_L$. Each hash function is of the form $g_i =< h_{i,1}, \cdots, h_{i,K} >$, where $h_{i,j}$ is chosen uniformly at random from $\mathcal{H}$. Then each point p is stored in bucket $g_i(p)$ of A_i for all $1 \leq i \leq L$. In order to answer a query q, we then search points in $A_1(g_1(q)) \cup \cdots \cup A_L(g_L(q))$. That is, from each array, we only have to look into the single bucket which corresponds to the query point q.

In this paper, for simplicity, we consider the LSH for the Hamming distance. However, similar results can be shown for general LSH functions. We recall the following lemma from [10].

LEMMA 3. *Let $dist(p, q)$ be the Hamming metric for $p, q \in \Sigma^d$, where Σ is any finite alphabet. Then for any $r, c \geq 1$, there exists a family $\mathcal{H}$ which is $(r, rc, p_1, p_2) - sensitive$, where $p_1 = 1 - r/d$ and $p_2 = 1 - rc/d$. Also, if we let $\rho = \frac{\log 1/p_1}{\log 1/p_2}$, then we have $\rho \leq 1/c$. Furthermore, by padding extra zeros, we can assume that $r/d \leq 1/2$.*

4. ALGORITHM A

The algorithm (first introduced in [1]) is based on the LSH algorithm. During the preprocessing, LSH creates L hash functions $g_1, g_2, \cdots, g_L$, and the arrays $A_1, A_2, \cdots, A_L$. Then each point p is stored in buckets $A_i[g_i(p)]$, for all $i = 1 \cdots L$. Furthermore, for each array A_i, the algorithm uses GMM to compute a 2-approximation of the optimal k-subset of each bucket, and stores it in the corresponding bucket of A'_i. This computed subset turns out to be a 1/3-coreset of the points of the bucket.

Given a query q, the algorithm computes the union of the buckets $Q = A'_1(g_1(q)) \cup \cdots \cup A'_L(g_L(q))$, and then it removes from Q all **outlier** points, i.e., the points which are not within distance cr of q. In the last step, the algorithm runs GMM on the set Q and returns the approximate optimal k-subset of Q.

The pseudo codes are shown in Algorithm 2 and 3. In the next section we discuss why this algorithm works.

4.1 Analysis

In this section, first we determine the value of the parameters L and K in terms of n and $\rho \leq 1/c$, such that with constant probability, the algorithm works. Here, L is the total number of hash functions used, and K is the number of hash functions $h_{i,j}$ used in each of the g_i. We also need to argue that limiting the size of the buckets to k, and storing only the approximate k most diverse points in A', works well to achieve a good approximation. We address these issues in the following.

Algorithm 2 Preprocessing

Input $G = \{g_1, \cdots, g_L\}$: set of L hashing functions, P: collection of points, k
Output $A' = \{A'_1, \cdots, A'_L\}$

```
for all points p ∈ P do
    for all hash functions g_i ∈ G do
        add p to the bucket A_i[g_i(p)]
    end for
end for
for A_i ∈ A do
    for j = 1 → size(A_i) do
        A'_i[j] = GMM(A_i[j], k) // only store the approximate k-diverse points in each bucket
    end for
end for
```

Algorithm 3 Query Processing

Input q: The query point, k
Output Q : The set of k-diverse points.

```
Q ← ∅
for i = 1 → L do
    Q ← Q ∪ A'_i[g_i(q)]
end for
for all p ∈ Q do
    if dist(q, p) > cr then
        remove p from Q // since it is an outlier
    end if
end for
Q ← GMM(Q, k)
return Q
```

LEMMA 4. *For $c > 2$, There exists hash functions $g_1, \cdots, g_L$ of the form $g_i =< h_{i,1}, \cdots, h_{i,K} >$ where $h_{i,j} \in \mathcal{H}$, for $\mathcal{H}$, p_1 and p_2 defined in 3, such that by setting $L = (\log(4k)/p_1)^{1/(1-\rho)} \times (4n)^{\rho/(1-\rho)}$, and $K = \lceil \log_{1/p_2}(4nL) \rceil$, the following two events hold with constant probability:*

- *$\forall p \in Q^* : \exists i$ such that $p \in A_i[g_i(q)]$, where Q^* denotes the optimal solution (the optimal k-subset of $P \cap B(q, r)$).*
- *$\forall p \in \bigcup_i A_i[g_i(q)] : dist(p, q) \leq cr$, i.e., there is no outlier among the points hashed to the same bucket as q in any of the hash functions.*

See Appendix A for a proof.

COROLLARY 1. *Since each point is hashed once in each hash function, the total space used by this algorithm is at most*

$$nL = n(\log(4k)/p_1(4n)^{\rho})^{1/(1-\rho)} = O((\frac{n \log k}{1 - r/d})^{\frac{1}{1-\rho}})$$

$$= O((n \log k)^{1+\frac{1}{c-1}})$$

where we have used the fact that $c > 2$, $\rho \leq 1/c$, and $r/d \leq 1/2$. Also we need $O(nd)$ space to store the points.

COROLLARY 2. *The query time is $O(((\log n)/r + k^2) \cdot (\log k)^{\frac{c}{c-1}} \cdot n^{\frac{1}{c-1}} d)$.*

PROOF. The query time of the algorithm for each query is bounded by $O(L)$ hash computation each taking $O(K)$

$$\begin{aligned} O(KL) &= O((\log_{1/p_2}(4nL)) \cdot L) = O(\frac{\log n}{\log(1/p_2)} L) \\ &= O(\frac{d}{r} \log n \cdot (\frac{\log k}{1-r/d})^{\frac{c}{c-1}} \cdot n^{\frac{1}{c-1}}) \\ &= O(\frac{d}{r} (\log k)^{\frac{c}{c-1}} n^{\frac{1}{c-1}} \log n) \end{aligned}$$

Where we have used the approximation $\log p_2 \approx 1 - p_2 = \frac{cr}{d}$, $c \geq 2$ and $r/d \leq 1/2$.

Also in the last step, we need to run the GMM algorithm for at most kL number of points in expectation. This takes

$$\begin{aligned} O(k^2 Ld) &= O(k^2 \cdot (\log k/p_1(4n)^{\rho})^{1/(1-\rho)} d) \\ &= O(k^2 (\log k)^{\frac{c}{c-1}} \cdot n^{\frac{1}{c-1}} d) \end{aligned}$$

□

LEMMA 5. *$GMM(S, k)$ computes a 1/3-coreset of S.*

PROOF. Suppose that the set of k points computed by GMM is S'. Now take any subset of $k-1$ points $F \subset P$. By pigeonhole principle there exist $a, b \in S'$ whose closest point in F is the same, i.e., there exists $c \in F$, such that

$$c = \arg\min_{f \in F} dist(a, f) = \arg\min_{f \in F} dist(b, f)$$

and therefore, by triangle inequality we get

$$div(S') \leq dist(a, b) \leq dist(a, c) + dist(b, c) \leq 2KC(S', F)$$

Now take any point $s \in S$ and let s' be the closest point of S' to s and f be the closest point of F to s'. Also let $a \in S'$ be the point added in the last step of the GMM algorithm. Then from definitions of s' and a, and the greedy choice of GMM, we have

$$dist(s, s') \leq \min_{p \in S' \setminus \{a\}} dist(p, s) \leq \min_{p \in S' \setminus \{a\}} dist(p, a) \leq div(S')$$

and thus by triangle inequality,

$$\begin{aligned} dist(s, f) &\leq dist(s, s') + dist(s', f) \leq div(S') + KC(S', F) \\ &\leq 3KC(S', F) \end{aligned}$$

Since this holds for any $s \in S$, we can infer that $KC(S, F) \leq 3KC(S', F)$ which completes the proof. □

LEMMA 6. *Suppose $S_1, \cdots, S_m$ are subsets of P, and let $S = \bigcup_i S_i$. Also suppose that $T_i = GMM(S_i, k)$ is the 2-approximation of the optimal k-subset of S_i which is achieved by running the GMM algorithm on S_i. Also define $T = \bigcup_i T_i$, and let $T' = GMM(T, k)$ be the 2-approximation of the optimal k-subset of T. Then we have $div(T') \geq \frac{1}{6} div_k(S)$.*

PROOF. Let S' denote the optimal k-subset of S. Also let $a \in T'$ be the added point at the last step of algorithm $GMM(T, k)$ and $T'_- = T' \setminus \{a\}$. By pigeonhole principle there exists $p, p' \in S'$ and $c \in T'_-$ such that

$$c = \arg\min_{t \in T'_-} dist(t, p) = \arg\min_{t \in T'_-} dist(t, p')$$

Therefore, by triangle inequality, lemma 5, and the fact that union of β-coresets of S_i is a β-coreset for the union S, we have

$$\begin{aligned} div_k(S) = div(S') &\leq dist(p, p') \leq dist(p, c) + dist(p', c) \\ &\leq 2KC(S, T'_-) \leq 6KC(T, T'_-) \end{aligned}$$

And since for any $b \in T$ we have

$$\min_{t \in T'_-} dist(t, b) \leq \min_{t \in T'_-} dist(t, a) \leq div(T')$$

And thus, $KC(T, T'_-) \leq div(T')$ and the lemma follows. □

COROLLARY 3. *With constant probability, the approximation factor achieved by the algorithm is 6.*

PROOF. Let $S_i = A_i[g_i(q)]$ and $S = \bigcup_i S_i$. Also let $T_i = A'_i[g_i(q)]$ and $T = \bigcup_i T_i$. Furthermore define Q^* be the optimal k-subset of $P \cap B(q, r)$, $T' = \text{GMM}(T, k)$ and Q be our returned set. From the description of the algorithm, it is obvious that $Q \subset B(q, cr)$. So, we only have to argue about its diversity.

By Theorem 4, with constant probability the two following statements hold:

- $S \subset B(q, cr)$. Therefore, we have $T \subset B(q, cr)$, which shows that when we run the Algorithm 3 for q, since T contains no outlier, the GMM algorithm in Line 10 is called on the set T itself and thus, $Q = T'$.
- $Q^* \subset S$. So we have $div_k(S) \geq div_k(Q^*) = div(Q^*)$

Therefore, by Lemma 6 we have

$$div(Q) \geq \frac{1}{6} div_k(S) \geq \frac{1}{6} div(Q^*)$$

□

5. ALGORITHM B

In this section, we introduce and analyze a modified version of Algorithm A which also achieves a constant factor approximation. Suppose that we knew the total number of outliers in any bucket is at most ℓ. Then, we can store for each single bucket of the array A, an ℓ-robust $\frac{1}{3}$-coreset in the corresponding bucket of array A'. First we show in Algorithm 4, how to find an ℓ-robust β-coreset if we know how to find a β-coreset. This is the algorithm of [3] that "peels" coresets β-coresets, and its analysis follows [15].

Algorithm 4 (ℓ, β)-coreset

Input S: set of points
Output S': An array which is a (ℓ, β)-coreset of S

1: $S' \leftarrow \emptyset$
2: **for** $i = 1 \rightarrow (\ell + 1)$ **do**
3: $\quad R_i \leftarrow \beta$-coreset of S
4: $\quad$ Append R_i to the end of S'
5: $\quad S \leftarrow S \setminus R_i$
6: **end for**
7: **return** S'

LEMMA 7. *Let $O \subset P$ be the set of outliers and S'_j denote set of the first (kj) points in S' which is S' after the jth round of the algorithm. Then for any $0 \leq j \leq \ell$ that satisfies $|S'_{j+1} \cap O| \leq j$, we have that $S'_{j+1} \setminus O$ is a β-coreset for $S \setminus O$.*

PROOF. Let $F \subset P$ be any subset of $(k-1)$ points, and q be the furthest point from F in $(S \setminus O)$, i.e.,

$$q = \arg\max_{p \in S \setminus O} \min_{f \in F} dist(p, f)$$

Now for any $i \leq (j+1)$, if q is a point in R_i, then the lemma holds since $KC(S,F) = KC(S'_{j+1},F)$. Otherwise, because q has not been chosen in any of the first $(j+1)$ rounds, each of the R_i's (for $i \leq j+1$) contains an r_i such that $KC(\{r_i\},F) \geq \beta KC(\{q\},F)$. Of these $(j+1)r_i$'s, at least one is not in O and therefore, $KC(S'_{j+1} \setminus O, F) \geq \beta KC(S \setminus O, F)$. □

COROLLARY 4. *Algorithm 4 computes the (ℓ,β)-coreset of S correctly.*

PROOF. Note that here for any set of outliers $O \subset P$ such that $|O| \leq \ell$, the condition in lemma 7 is satisfied for $j = \ell$. Thus when the algorithm returns, it has computed an ℓ-robust β-coreset correctly. □

Since by lemma 5, we know that GMM(S,k) computes a $\frac{1}{3}$-coreset of size k for S, it is enough to replace line 3 of the algorithm 4 with $R \leftarrow GMM(S,k)$, in order to achieve an ℓ-robust $\frac{1}{3}$-coreset of size $k(\ell+1)$ for the set S.

Then the only modification to the preprocessing part of Algorithm A is that now, each bucket of A'_i keeps an ℓ-robust $\frac{1}{3}$-coreset of the corresponding bucket of A_i. So the line 8 of Algorithm 2 is changed to $A'_i[j] = (\ell,\beta)$-coreset$(A_i[j], \ell = 3L, \beta = 1/3)$.

The pseudo-code of processing a query is shown in Algorithm 5. For each bucket that corresponds to q, it tries to find the smallest value of ℓ such that the total number of outliers in the first $k(\ell+1)$ elements of $A'_i[g_i(q)]$ does not exceed ℓ. It then adds these set of points to T and returns the approximate optimal k subset of non-outlier points of T.

Algorithm 5 Query Processing

Input q: The query point
Output Q : The set of k-diverse points.

```
1: T ← ∅
2: O ← set of outliers
3: for i = 1 → L do
4:    for ℓ = 0 → 3L do
5:       U_i^{ℓ+1} = the first k(ℓ+1) points of A'_i[g_i(q)]
6:       if |U_i^{ℓ+1} ∩ O| ≤ ℓ then
7:          ℓ_i ← ℓ
8:          T_i ← U_i^{ℓ+1}
9:          break
10:      end if
11:   end for
12:   T ← T ∪ T_i
13: end for
14: Q ← GMM(T \ O, k)
15: return Q
```

Note that the inner loop (lines 4 to 11) of Algorithm 5 can be implemented efficiently. Knowing the number of outliers in U_i^j, there are only k more elements to check for being outliers in U_i^{j+1}. Also, each point can be checked in $O(d)$ if it is an outlier. So in total the inner loop takes $O(k\ell_i d)$.

5.1 Analysis

We first state the following theorem which is similar to Theorem 4. The proof is very similar to the original proof of correctness of the LSH algorithm given in [10], and hence omitted.

THEOREM 1. *There exists hash functions $g_1,\cdots,g_L$ of the form $g_i =< h_{i,1},\cdots,h_{i,K} >$ where $h_{i,j} \in \mathcal{H}$, for $\mathcal{H}$, p_1 and p_2 defined in 3 such that by setting $L = \log(4k)\times n^\rho/p_1$, and $K = \lceil \log_{1/p_2} n \rceil$, with constant probability the following two events hold:*

- *$\forall p \in Q^*$: $\exists i$ such that $p \in A_i[g_i(q)]$, where Q^* denotes the optimal solution (the optimal k-subset of $P \cap B(q,r)$).*
- *$|\{p \in \bigcup_i A_i[g_i(q)] : dist(p,q) > cr\}| \leq 3L$, i.e. the number of outliers among points hashed to the same bucket as q, is at most $3L$.*

COROLLARY 5. *Since each point is hashed once in each hash function and each bucket of A'_i is a subset of the corresponding bucket of A_i, the total space used by this algorithm is at most*

$$nL = n\log(4k)\cdot n^\rho/p_1 = O(\log k \cdot n^{1+1/c})$$

where we have used the fact that $\rho \leq 1/c$, and that $p_1 = 1 - r/d \geq 1/2$. Also we need $O(nd)$ space to store the points.

THEOREM 2. *With constant probability, the approximation factor achieved by the algorithm is 6.*

PROOF. First we start by defining a set of notations which is useful in the proof.

- Let $S_i = A[g_i(q)]$, and $S = \bigcup_i S_i$.
- Let O be the set of outliers, i.e., $O = S \setminus B(q,cr)$. We know that with constant probability $|O| \leq 3L$
- Let S' be the optimal k-subset of $S \setminus O$.
- Let $U_i = A'_i[g_i(q)]$ and U_i^j be the first jk elements of U_i (note that since Algorithm 4 returns an array, the elements of U are ordered). We define $T_i = U_i^{(\ell_i+1)}$ where ℓ_i is chosen such that for any $\ell'_i < \ell_i$, we have $\left|U_i^{(\ell'_i+1)} \cap O\right| > \ell'_i$. This is exactly the T_i variable in Algorithm 5. Moreover, let $T = \bigcup_i T_i$.
- Define Q^* to be the optimal k-subset of $P \cap B(q,r)$, and Q be our returned set, i.e., $Q = GMM(T \setminus O, k)$. Let $a \in Q$ be the added point at the last step of GMM, then define $Q_- = Q \setminus \{a\}$.

From the description of algorithm it is obvious that $Q \subset B(q,cr)$. Also by Theorem 1, with constant probability we have $Q^* \subset S$, and that the total number of outliers does not exceed $3L$. Thus we have $div_k(Q^*) = div(Q^*) \leq div_k(S \setminus O)$, and therefore it is enough to prove that under these conditions, $div(Q) \geq div_k(S \setminus O)/6 = div(S')/6$.

By pigeonhole principle, since $|S'| = k$ and $|Q_-| = k-1$, then there exist $p, p' \in S'$ whose closest point in Q_- is the same, i.e., there exists $c \in Q_-$ such that $KC(\{p\},Q_-) = dist(p,c)$ and $KC(\{p'\},Q_-) = dist(p',c)$. Therefore, by triangle inequality, we have

$$\begin{aligned} div(S') &\leq dist(p,p') \leq dist(p,c) + dist(p',c) \\ &\leq 2KC(S \setminus O, Q_-) \end{aligned} \tag{1}$$

By lemma 7 $T_i \setminus O$ is a $\frac{1}{3}$-coreset for $S_i \setminus O$, and therefore their union $T \setminus O$, is a $\frac{1}{3}$-coreset for $S \setminus O$, and thus we have

$$KC(S \setminus O, Q_-) \leq 3KC(T \setminus O, Q_-) \tag{2}$$

Now note that a is chosen in the last step of GMM$(T \setminus O, k)$. Thus for any point $b \in (T \setminus O) \setminus Q$, since it is not chosen by GMM, b should be closer to Q_- than a, i.e., we should have $KC(\{b\}, Q_-) \leq KC(\{a\}, Q_-)$. This means that

$$KC(T \setminus O, Q_-) \leq KC(\{a\}, Q_-) \leq div(Q) \quad (3)$$

Putting together equations 1, 2 and 3 finishes the proof. □

LEMMA 8. *With constant probability the query time is* $O((k^2 + \frac{\log n}{r})d \cdot \log k \cdot n^{1/c})$

PROOF. The query time of the algorithm for each query, has three components. First there are $O(L)$ hash computations each taking $O(K)$

$$O(KL) = O((\log_{1/p_2} n) \cdot L) = O(\frac{d}{r} \log n \cdot \frac{\log k}{1 - r/d} \cdot n^{1/c})$$
$$= O(\log k \cdot n^{1/c} \cdot \log n \cdot \frac{d}{r})$$

Where we have used the approximation $\log p_2 \approx 1 - p_2 = \frac{cr}{d}$, $c \geq 1$ and $r/d \leq 1/2$. Second, in the last step of Algorithm 5, with constant probability the total number of outliers is at most $3L$. Therefore, we need to run the GMM algorithm for at most $O(kL)$ number of points, i.e., $|T| \leq 3L$. Then GMM takes

$$O(k^2 Ld) = O(d \cdot k^2 \log k \cdot n^{1/c})$$

Finally, as mentioned before, the inner loop (steps $4 - 11$) of the algorithm 5 can be implemented incrementally such that the total time it takes is $O(k\ell_i d)$.Thus the total running time of the loop is $O(kd \sum_i \ell_i) = O(kLd)$. □

6. ACKNOWLEDGMENTS

This work was supported a grant from Draper Lab, an NSF CCF-1012042 award, MADALGO project and the Packard Foundation.

7. REFERENCES

[1] S. Abbar, S. Amer-Yahia, P. Indyk, and S. Mahabadi. Efficient computation of diverse news. In *WWW*, 2013.

[2] P. K. Agarwal, S. Har-peled, and K. R. Varadarajan. Geometric approximation via coresets. In *Combinatorial and Computational Geometry*, volume 52, pages 1–30, 2005.

[3] P. K. Agarwal, S. Har-peled, and H. Yu. Robust shape fitting via peeling and grating coresets. In *In Proc. 17th ACM-SIAM Sympos. Discrete Algorithms*, pages 182–191, 2006.

[4] A. Andoni. LSH algorithm and implementation (E2LSH). http://www.mit.edu/ andoni/LSH/.

[5] A. Angel and N. Koudas. Efficient diversity-aware search. In *SIGMOD*, pages 781–792, 2011.

[6] M. Drosou and E. Pitoura. Search result diversification. *SIGMOD Record*, pages 41–47, 2010.

[7] P. Fraternali, D. Martinenghi, and M. Tagliasacchi. Top-k bounded diversification. In *SIGMOD*, pages 421–432, 2012.

[8] S. Gollapudi and A. Sharma. An axiomatic framework for result diversification. In *WWW*.

[9] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance.

[10] S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory Of Computing*, 8:321–350, 2012.

[11] S. Har-peled and Y. Wang. Shape fitting with outliers. *SIAM J. Comput.*, 33(2):269–285, 2004.

[12] A. Jain, P. Sarda, and J. R. Haritsa. Providing diversity in k-nearest neighbor query results. In *PAKDD*, pages 404–413, 2004.

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

[14] Z. Syed, P. Indyk, and J. Guttag. Learning approximate sequential patterns for classification. *Journal of Machine Learning Research.*, 10:1913–1936, 2009.

[15] K. Varadarajan and X. Xiao. A near-linear algorithm for projective clustering integer points. In *SODA*, pages 1329–1342, 2012.

[16] M. J. Welch, J. Cho, and C. Olston. Search result diversity for informational queries. In *WWW*, pages 237–246, 2011.

[17] C. Yu, L. V. S. Lakshmanan, and S. Amer-Yahia. Recommendation diversification using explanations. In *ICDE*, pages 1299–1302, 2009.

[18] C.-N. Ziegler, S. M. Mcnee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *WWW*, pages 22–32, 2005.

APPENDIX

A. PROOF OF LEMMA 4

PROOF. For the first argument, consider a point $p \in Q^*$. By Definition 6 the probability that $g_i(p) = g_i(q)$ for a given i, is bounded from below by

$$p_1^K \geq p_1^{\log_{1/p_2}(4nL)+1} = p_1(4nL)^{-\frac{\log 1/p_1}{\log 1/p_2}} = p_1(4nL)^{-\rho}$$

Thus the probability that no such g_i exists is at most

$$\zeta = (1 - p_1(4nL)^{-\rho})^L \leq (1/e)^{L \cdot \frac{p_1}{(4nL)^\rho}} = (1/e)^{L^{(1-\rho)} \cdot \frac{p_1}{(4n)^\rho}}$$
$$= (1/e)^{(\log(4k)/p_1(4n)^\rho) \cdot \frac{p_1}{(4n)^\rho}} \leq \frac{1}{4k}$$

Now using union bound, the probability that $\forall p \in Q^* : \exists i$, such that $p \in A_i[g_i(q)]$ is at least $\frac{3}{4}$.

For the second part, note that the probability that $g_i(p) = g_i(q)$ for $p \in P \setminus B(q, cr)$ is at most $p_2^K = \frac{1}{4nL}$. Thus, the expected number of elements from $P \setminus B(q, cr)$ colliding with q under fixed g_i is at most $\frac{1}{4L}$, and the expected number of collisions in all g functions is at most $\frac{1}{4}$. Therefore, with probability at least $\frac{3}{4}$, there is no outlier in $\bigcup_i A_i[g_i(q)]$.

So both events hold with probability at least $\frac{1}{2}$. □