Diverse Browsing for Spatial Data

Onur Kucuktunc
The Ohio State University
Department of Computer Science and Engineering
Columbus, OH, USA 43210
kucuktun@cse.ohio-state.edu

Hakan Ferhatosmanoglu
The Ohio State University
Department of Computer Science and Engineering
Columbus, OH, USA 43210
hakan@cse.ohio-state.edu

ABSTRACT

Database search techniques try to obtain the most relevant information and rank it according to the degree of similarity to the queries. However, diversity in query results is also preferred by a variety of applications since very similar results do not give complete view of the queried topic. In this work, we focus on providing diverse query results for the k-nearest neighbor search on spatial data. We make an analogy with the concept of natural neighbors and propose a natural neighbor-based method for 2-d and 3-d data, and an incremental browsing algorithm based on Gabriel graphs for higher dimensional spaces. We also introduce a diverse browsing method based on the popular distance browsing feature of spatial index structures, such as R-trees. The algorithm maintains a priority-queue with the ranks of the objects depending on both relevancy and diversity, and efficiently prunes non-diverse items and nodes. The efficiency and effectiveness of the models and each algorithm are demonstrated with different settings, and compared with the methods found in the literature.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Query formulation, Search process

1. INTRODUCTION

Most similarity search methods in the literature produce results based on the ranked degree of similarity to the query. However, the results are typically unsatisfactory, especially when there is an ambiguity in the query, and when the search results include redundantly repeating similar documents.

To resolve ambiguity, it would be better to answer the query with diverse search results instead of homogeneous results representing similar cases. For example, the query *Barcelona* is ambiguous since the system cannot decide whether it represents a city, a football team, or a movie [2]. One of the reasonable strategies for responding to ambiguous queries is to return a mixture of results covering all aspects of the query. Redundantly repeating search results is another problem of conventional similarity search techniques, particularly for search spaces that include many duplicate data. In this case, similar but homogeneous information will fill up the top results. This situation has been discussed in several application areas, such as recommender systems [26], online shopping [22], and web searches [7].

Similar problems also exist when querying and browsing spatial data. With the popularity of location-based services, most smartphones today come with GPS and solid state compasses, and people use this technology for navigation and searching nearby restaurants, gas stations, etc. A location-based service application that defines user's position with a number of point-of-interests (POI) can be affected adversely with the information overload (see Figure 1). Instead of returning the closest POIs, close but a more diverse result set is preferred for such an application.

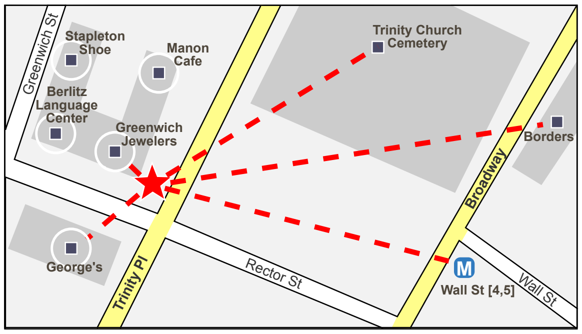

Figure 1: Sample location-based service application that defines user's position (*red star*) with 5 POIs. A conventional similarity search technique (i.e. k-nearest neighbor search) returns redundant results, *white circles*, because of the information overload; on the other hand, diverse browsing can capture spatial distribution around the query point and can provide superior results, *red dashes*.

The relation between relevance and diversity in information retrieval is investigated by Carbonell and Goldstein [5] and they propose the Maximal Marginal Relevance (MMR) method for text retrieval and summarization. Researchers have been trying to apply the same idea to various fields; however, maximizing diversity of a result set is known to be NP-hard. Addressing the aforementioned problem, some studies [14, 24] develop heuristic techniques to optimize the results. Yu *et al.* [24] concentrate on the issue of diversification in recommender systems and introduce two heuristic algorithms to maximize diversity by considering relevance constraints. In the spatial domain, leveraging the advantages of an index structure and its incremental browsing is necessary for an efficient diversification algorithm. Jain *et al.* [14] investigate the k-nearest diverse neighbor search (KNDN) and present greedy approaches based on the distance browsing feature [12] of the R-tree index; however, they use the distance browsing only to retrieve the next nearest neighbor to the query point.

Considering the diversification problem in the spatial domain, it is possible to present an intuitive solution based on clustering. Data can be initially clustered, then representatives of diverse clusters around the query point can be given as the results of a diverse nearest neighbor search. Although clustering can be computationally expensive, there are methods to generate those representatives with tree-based approaches [17]. The problem of clustering-based methods is that initial clusters may be unsatisfactory depending on the settings of the query. Furthermore, if data needs to be clustered for each query, the method is obviously not scalable. Today, most database management systems support a spatial index, i.e. R-tree or one of its variants; therefore, diversity can be obtained by taking advantage of the spatial index without any extra cost of clustering.

In this study, we first give a geometric definition of *diversity* by making an analogy with the concept of natural neighbors and propose a natural neighbor-based method and an incremental browsing algorithm based on Gabriel graph. We also introduce a diverse browsing method based on the popular distance browsing feature of R-tree index structures, which maintains a priority-queue with the ranks of the objects depending on both relevancy and diversity, and efficiently prunes non-diverse items and nodes. Providing a measure that captures both relevancy and diversity, we show that using the advantages of the R-trees and pruning internal nodes with respect to their diversity from the items in the result set, we can achieve more diverse results. Briefly, the contributions of this paper can be summarized as follows:

- We formalize **λ-diverse k-nearest neighbor search** based on angular similarity, and develop measures that evaluate the relevancy and diversity of the retrieved results.
- We propose two geometric diverse browsing approaches for static databases, each of which effectively captures the spatial distribution around a query point, hence gives a diverse set of results.
- Extending the distance browsing feature, we introduce an efficient λ-diverse k-nearest neighbor search algorithm on R-trees, namely diverse browsing, which does not require any change in the index structure, and prove its correctness.
- We conduct experiments on 2-d and high-dimensional datasets to evaluate the performance of the proposed methods.

The rest of the paper is organized as follows: Related work is discussed in Section 2. Problem formulation is given in Section 3. Geometric approaches are defined in 4, and index-based diverse browsing is proposed in Section 5. Experiments are reported in Section 6. Section 7 gives conclusions and future work.

2. RELATED WORK

There are notable works on diverse ranking in the literature. Carbonell and Goldstein [5] describe the Maximal Marginal Relevance (MMR) method for text retrieval and summarization. MMR attempts to find a result set by maximizing the query relevance and also minimizing the similarity between documents in the result set. The proposed method combines relevancy and novelty with a user-defined parameter (λ), which affects the relevancy and diversity of the results.

Since the problem of finding diverse results is known to be NP-hard, Jain *et al.* [14, 11] investigate the k-nearest diverse neighbor search and develop two greedy approaches to optimize the results in terms of both relevancy and diversity. Both proposed methods employ the advantages of an available R-tree index. *Immediate Greedy* (IG) incrementally grows the result set R by including nearest points only if they are diverse enough from the data points already in R. *Buffered Greedy* (BG) tries to resolve some poor choices of IG. They use the R-tree index only for getting the query's nearest neighbors in the dataset. Yu *et al.* [24, 23] address the issue of diversification in recommendation systems and introduce two heuristic algorithms to maximize the diversity by considering relevance constraints. They state that maximizing diversity is about finding a balance between relevance and diversity. The proposed *Swap algorithm* basically tries to swap elements which are less likely to contribute to the set diversity with diverse ones. *Greedy algorithm*, similar to IG in [14], includes the next most relevant item to the result set only if that item is diverse with respect to the items already in the result set.

Some other studies attack on the diversity problem in various ways. Liu and Jagadish [17] employ the idea of clustering to find a solution to the Many-Answers Problem. They suggest that taking one representative from each cluster results in a better answer to this problem. This paper proposes a tree-based approach for finding the representatives efficiently, even if the search space is modified with select conditions at runtime. Halvey *et al.* [10] compare dissimilarity and clustering-based diversity re-ranking methods to introduce diversity in video retrieval results.

The notions of diversity and novelty are generally discussed in information retrieval and recommendation systems. Clarke *et al.* [7] investigate the problems of ambiguity in queries and redundancy in results and propose an evaluation framework. Chen and Karger [6] describe a retrieval method for maximizing diversity, which assigns negative feedback to the documents that are included in the result list. Vee *et al.* [22] present inverted-list algorithms for computing diverse query results in online shopping applications. Ziegler *et al.* [26, 25] present an algorithmic framework to increase the diversity of a top-N list of recommended products. In order to show its efficiency they also introduce a new intra-list similarity metric.

3. PROBLEM FORMULATION

Angular similarity and diverse k-nearest neighbor search is defined in Definition 3.1 and Definition 3.2, respectively.

Definition 3.1. **Angular similarity.** *Given a query point x, two points p_1 and p_2, and an angle θ,* angular similarity *(sim_{ang}) of p_1 with respect to x and another point p_2 is:*

$$sim_{ang}(p_1, x, p_2) = \begin{cases} 1 - \widehat{p_1 x p_2}/\theta & \text{if } \widehat{p_1 x p_2} < \theta \\ 0 & otherwise \end{cases} \quad (1)$$

sim_{ang} results in 0 if the angle $\widehat{p_1 x p_2}$ is greater than θ. It becomes 1 if both of them point the same direction.

Definition 3.2. **λ-diverse k-nearest neighbor search.** *Given a dataset S, a query point x, a diversity ratio λ, and an integer k, the* λ-diverse k-nearest neighbor search *on x retrieves a set of k resulting points denoted by R that minimizes pairwise angular similarity (depending on λ) and maximizes relevancy (depending on 1-λ) of the results.*

4. GEOMETRIC DIVERSE BROWSING

In the spatial domain, diverse nearest neighbor search is conceptually similar to the idea of natural neighbors, which is calculated with Voronoi diagrams (VD) and Delaunay triangulation (DT) [15, 3]. In this section, we first present an analogy of diversity with natural neighbors. Based on the analogy, we propose the NatN-based method along with the techniques that are used to retrieve natural neighbors of a query point efficiently. Although the discussions are mostly on 2-d space, the method can be extended to work on 3-d space. Observing the limitations of NatN-based method in higher dimensional spaces, we present another geometric approach, Gabriel graph-based diverse browsing method.

4.1 Analogy with Natural Neighbors

The *natural neighbors* (NatN) of a point $p \in S$ are the points in S sharing an edge with p in DT. They are also the ones whose Voronoi cells are neighbors of V_p. In the case of a point $x \notin S$, its natural neighbors are the points in S whose Voronoi cells would be modified if x is inserted in VD(S). The insertion of x creates a new Voronoi cell V_x^+ that steals volume from the Voronoi cells of its potential natural neighbors (see Figure 2).

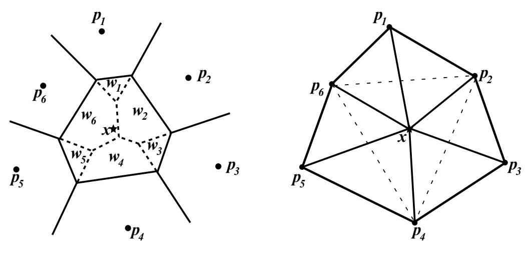

Figure 2: Natural neighbor coordinates w_i of x in 2-d (left), DT with and without x (right).

To capture the influence of each NatN, we use natural neighbor weights/coordinates in natural neighbor interpolation [21].

Let D be the VD(S), and $D^+ = D \cup \{x\}$. The Voronoi cell of a point p in D is defined by V_p, and V_p^+ is its cell in D^+. The natural neighbor weight of x with respect to a point p_i is

$$w_i(x) = \frac{Vol(V_{p_i} \cap V_x^+)}{Vol(V_x^+)} \quad (2)$$

where $Vol(V_{p_i})$ represents the volume of V_{p_i}, and $0 \leq w_i(x) \leq 1$. The natural neighbor weights are affected by both the distance from x to p_i and the spatial distribution of the p_i around x.

4.2 Natural Neighbor-based Method

Based on the property of the natural neighbor concept which captures both the distance to a query point and also the spatial distribution around it, we propose that the natural neighbors of a query point x give a diverse set of similarity search results, where the natural neighbor coordinates $w_i(x)$ are used as ranking measures. The method works as follows: (1) simulate the insertion of x into DT(S), (2) find the natural neighbors of x: $\{p_1, \cdots, p_k\}$ along with the weights $\{w_1, \cdots, w_k\}$, and (3) report results according to the weights in descending order. Details are provided below:

Offline Generation of DT: In the preprocessing stage, DT(S) for all the data points in S is calculated. Although the weights are calculated with the overlapping areas of these cells, and the definition of Natural neighbor is defined (and also easier to understand) in terms of Voronoi cells, performing operations on DT is computationally more efficient.

There are I/O and memory efficient methods for building Delaunay triangulation in 2-d and 3-d [13, 1] which can generate DTs for billions of points very efficiently. There are also publicly available implementations for 2-d [20] and for higher dimensions [4]. We use Qhull implementation [4] to generate DT(S).

Step 1 - Flip-based Incremental Insertion: We simulate the insertion of x into DT(S) with a flip-based insertion algorithm. It is easy to determine the simplex of DT(S) containing x within linear time by inspecting all the triangles.

Let τ be the DT(S) and p_i the natural neighbors of x once it is inserted in DT(S). All the 3 vertices of simplex τ that contain x automatically become natural neighbors of x. Then, the necessary edge flips are carried out until no further edge needs to be flipped.

The number of flips needed to insert x is proportional to the degree of x (the number of incident edges) after its insertion. The average degree of a vertex in a 2-d DT is 6. This number increases along with the dimensionality.

Step 2 - Find NatNs and weights: Vertices p_i adjacent to x in DT($S \cup x$) are the natural neighbors of x. The volume of a d-dimensional Voronoi cell is computed by decomposing it into d-simplices and summing their volumes. The volume of a d-simplex τ is computed with [15]:

$$Vol(\tau) = \frac{1}{d!} \left| det \begin{pmatrix} v^0 \cdots v^d \\ 1 \cdots 1 \end{pmatrix} \right| \quad (3)$$

where v is a d-dimensional vector representing the coordinates of a vertex and *det* is the determinant of the matrix. Weights w_i are then calculated with Equation 2.

Step 3 - Report for Diverse kNN: NatN-based method naturally returns k' results as an answer to query point x. The result set R is ranked according to the weights w_i of each neighbor. If $k' \geq k$, we report top k ranked result. Otherwise, k' points are returned.

Overview of the method is given in Algorithm 1. Note that if the number of natural neighbors is greater than k, the points with smaller weights w_i are eliminated. Otherwise, the method may return less than k results.

Algorithm 1 Algorithm NatNDiversitySearch

```
 1: procedure NatNDiversitySearch($x$,$k$,DT)
 2:     DT$'$ ← Insert(DT,$x$)
 3:     $W \leftarrow \{\}$
 4:     for each point $p_i$ in adj[$x$] do
 5:         $w_i \leftarrow$ CalculateWeight(DT$'$, $p_i$, $x$)
 6:         $W \leftarrow W \cup \{w_i\}$
 7:     end for
 8:     $W' \leftarrow$ Sort($W$)
 9:     if adj[$x$] > k then
10:         $W' \leftarrow W'[1:k]$
11:     end if
12:     return $W'.i$
13: end procedure
```

4.3 Limitations

The drawback of using natural neighbors in the diverse k-nearest neighbor search is that there always is a fixed number of natural neighbors of a point, and the number is proportional to the dimensionality of the space. This can be seen even as an advantage since the parameter k is not specified by the user, it is inherently captured by the process. For browsing purposes, one cannot restrict the search with only natural neighbor results as the user may demand more search results. The search needs to continue incrementally through the neighbors of neighbors with Voronoi cells. Without any assumptions on the distribution of the data, the average degree of a vertex in a 2-d DT is 6 [15]. In this case, diverse k-nearest neighbor search with the NatN-based method for 2-d space may not return a result set with k items. As a result the method is forced to investigate the neighbors of neighbors with Voronoi cells which were not modified with the insertion of x.

For higher dimensional spaces, average degree of a point in DT grows quickly with d (approximately d^d) [8]. The problem of selecting a subset of elements in this set to obtain a diverse set of k items cannot be trivially solved with a NatN-based approach. Because of the disadvantages, NatN-based method is more appropriate for low-dimensional data and small k values.

4.4 Gabriel Neighbor-based Method

In high dimensions, DT is intractable in terms of both construction complexity $O(n^{\lceil d/2 \rceil})$ and browsing efficiency (because the average node degree is $\sim d^d$). For better scalability and browsing capability in high dimensional spaces, we propose using Gabriel graphs instead of DT.

The Gabriel graph [9] is the set of edges e_{ij} subset of DT(S), for which the circle with diameter $[p_i p_j]$ contains no other points from S.

$$\mathrm{GG}(S) = \{e_{ij} \subseteq \mathrm{DT}(S) | \; \forall p_k \in S, |p_k p_i|^2 + |p_k p_j|^2 \geq |p_i p_j|^2\} \tag{4}$$

Observe that the Gabriel graph (GG) contains those edges of DT that intersect their Voronoi faces [18]. Hence, GG can be constructed in $O(n \log n)$ time by first constructing DT and VD, and then adding each edge in DT to GG if it intersects its Voronoi face. Without DT and VD, GG can always be constructed by brute-force in $O(n^3)$ time.

The advantage of working with GG is that both nearest neighbor graph (NNG) and minimum spanning tree (MST) are subgraphs of it; therefore, GG still captures proximity relationships among data points. Furthermore, GG is reasonably sparse and simple: for planar graphs $|GG(S)| \leq 3n - 8$ [18]. As a result, Gabriel graph is particularly popular in constructing power efficient topology for wireless and sensor networks [16].

Our solution for diverse k-nearest neighbor search is to browse GG layer-by-layer, starting from the nearest point p_{nn} to the query point x. Figure 3 shows an example of GG layers connected with B-spline. For efficiency, the query point is not inserted into GG(S), but rather the spatial location of x is imitated with its nearest neighbor.

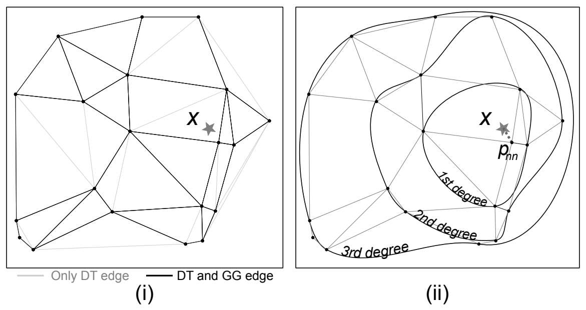

Figure 3: Incremental browsing of Gabriel graph. Delaunay triangulation of the points with Gabriel edges highlighted (i). For a query point x, diverse results are gathered layer-by-layer starting with p_{nn} (ii).

After finding p_{nn}, the algorithm iteratively search the n-degree neighbors of p_{nn} in GG(S), starting with $n = 1$. GGDiversitySearch stops when k or more points are included in R. Note that the resulting points are added layer-by-layer; therefore there is a ranking among layers. However, they are not sorted within layers, since there is no concept similar to natural neighbor weights in Gabriel graphs. In addition, $|R| \geq k$, meaning that the algorithm may return more than k results. The method is given in Algorithm 2.

It is possible to return exactly k results by examining the

Algorithm 2 Algorithm GGDiversitySearch

```
1: procedure GGDiversitySearch($x$,$k$,GG,$S$)
2:     $p_{nn} \leftarrow$ NearestNeighbor(S,$x$)
3:     $R \leftarrow \{\}$
4:     $R' \leftarrow \{p_{nn}\}$
5:     while $|R| < k$ do
6:         $R \leftarrow R \cup R'$
7:         $R'' \leftarrow \{\}$
8:         for each point $p$ in $R'$ do
9:             $R'' \leftarrow R' \cup$ adj[$p$]
10:        end for
11:        $R' \leftarrow (R'' \setminus R)$
12:    end while
13:    return $R$
14: end procedure
```

last layer of points added into R. One method is to choose a subset of points from the last layer, which optimizes the overall diversity of R. We are using a similar approach in the experiments.

5. INDEX-BASED DIVERSE BROWSING

Spatial databases mostly come with an index structure, such as the widely used R-tree [19]. A popular technique called *distance browsing* [12] tries to find the k-nearest neighbors (kNN) of a point in a spatial database that uses the R-tree index. Based on this method, we introduce *diverse browsing* for the diverse k-nearest neighbor search over an R-tree index.

The principal idea of diverse browsing is to use the distance browsing method defined in [12] with a pruning mechanism that omits non-diverse data points and minimum bounding rectangles (MBR). A priority queue is maintained with respect to a *rank*, which is a combination of the *mindist* and the angular similarity for the object (either data point or R-tree index node). In each iteration the closest object is investigated (see Figure 4).

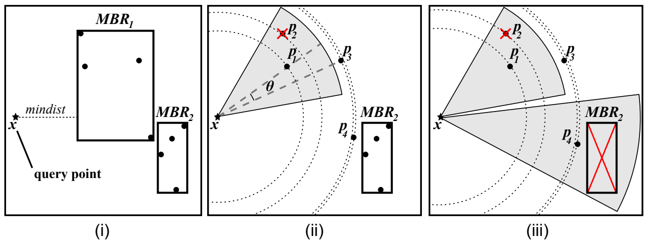

Figure 4: Example for diverse browsing. Suppose MBR_1 and MBR_2 are two internal nodes of the R-tree index (i). When the closest MBR is investigated and the closest point p_1 is added to the result set, p_2 is pruned because of the angular and distance similarity to p_1 (ii). The rank of p_3 is also increased here due to its angular similarity to p_1. Next, p_4 is added to the result set and causes MBR_2 to be pruned since none of the items in MBR_2 can be diverse (iii).

5.1 Pruning and Ranking

When a point p is added to the result set R, we draw an imaginary sector from the query point x in the direction of p with $\theta_{sector} = 2 \times \theta_s$ angle and $r_{sector} = r_s \times |\vec{qp}|$. Every point in this sector ($\circledast$) will eventually be pruned. In default, $r_s = 1 + \lambda$ and $\theta_s = \frac{2\pi}{k+\epsilon}$, unless specified otherwise.

We use the term **rank** as an alternative to *mindist* in distance browsing. Rank of each point and MBR in the priority queue is calculated according to the angular similarity and distance with respect to the elements in R (see Alg 3). Note that the points with ranks closer to 0 are more likely to be included in the result set.

Points without enough angular diversity and distance from another point in R are pruned. Similarly, the algorithm also prunes MBRs only if none of the corners of the object are diverse enough to be in the result set. The advantage here is that all the pruned data can be displayed as *similar results* of each resulting point with a small modification, since we have the information why a point is pruned.

Algorithm 3 Algorithm GetRank

```
1: procedure GetRank($x, obj, R, \lambda$)
2:     $\theta_s \leftarrow \frac{2\pi}{k+\epsilon}$; $r_s \leftarrow 1+\lambda$
3:     $dist \leftarrow mindist(x, obj)$
4:     if $obj$ is a point then
5:         for each point $p$ in $R$ do
6:             $angsim[p] \leftarrow sim_{ang}(obj, x, p)$
7:             if $angsim[p] > 0$ and $dist < |\vec{qp}| \times r_s$ then
8:                 return Prune($obj$)          ▷ $obj$ is inside $\circledast$
9:             end if
10:        end for
11:        $rank \leftarrow \lambda \times max(angsim) + (1-\lambda) \times dist$
12:    else if $e$ is a rectangle then
13:        for each point $p$ in $R$ do
14:            $angsim \leftarrow min_{y \in e.corners}(sim_{ang}(y, x, p))$
15:            if $\forall y \in e.corners$ in $\circledast$ then
16:                return Prune($obj$)
17:            else if $\exists y \in e.corners$ in $\circledast$ then
18:                $dist \leftarrow min(|\vec{qp}| \times r_s, |\vec{qy}|)$
19:            end if
20:            $rrank[p] \leftarrow \lambda \times angsim + (1-\lambda) \times dist$
21:        end for
22:        $rank \leftarrow max(rrank)$
23:    end if
24:    return $rank$
25: end procedure
```

5.2 Maintaining the Priority Queue

The efficiency of the proposed method comes from the *diverse browsing* of R-tree structure. As in incremental nearest neighbor search algorithms and distance browsing [12], a min-priority queue (PQ) is maintained after each operation. However, instead of *MinDist* metric we use the result of GetRank function as the value of each object in PQ. GetRank gives a non-negative value which is the rank of the point or the rectangle with respect to its angular diversity and distance from the query point depending on R and λ. In each iteration, the object with the lowest rank (top of PQ) is investigated.

As the rank of each object in PQ depends on the current state of R, some of the ranks will be obsolete after another point is inserted into R. But instead of updating all the objects in PQ (which would be inefficient), we argue to update only the top of PQ with a *timestamp-based* approach until an up-to-date object is acquired. The timestamp is incremented every time a point is included in the result set. The proposed method based on timestamp-based update of ranks in PQ is proved by Lemma 5.1 and Theorem 5.2.

LEMMA 5.1. *Update operation on an object, which is on top of PQ and has an earlier timestamp, can either increase the rank of the object, or does not affect it at all.*

PROOF. An object (obj) is updated only when $ts[obj] < timestamp$. Since the $timestamp$ increases when a new item is added to R, there are exactly $(timestamp - ts[obj])$ new items (R_{new}) in the result set compared to the time when $rank[obj]$ was calculated.

Suppose the new rank of obj at current timestamp is $rank'[obj]$. If obj is a point, three outcomes of the update are:

1. PRUNE(obj), if it resides in a pruning sector $\circledast$ of a new point in R,
2. $rank'[obj] > rank[obj]$, if $\exists p \in R_{new}$, $sim_{ang}(obj, x, p) > max_{r\in R}(sim_{ang}(obj, x, r))$
3. $rank'[obj] = rank[obj]$, if $\forall p \in R_{new}$, $sim_{ang}(obj, x, p) \leq max_{r\in R}(sim_{ang}(obj, x, r))$

On the other hand, if obj is a leaf or internal node, the rank depends on the corners of the MBR:

1. PRUNE(obj), if $\forall y \in obj.corners$ in a pruning sector $\circledast$ of a new point in R,
2. $rank'[obj] > rank[obj]$, if $\exists p \in R_{new}, y \in obj.corners$, $sim_{ang}(y, x, p) > max_{r\in R}(sim_{ang}(obj, x, r))$
3. $rank'[obj] = rank[obj]$, if $\forall p \in R_{new}, y \in obj.corners$, $sim_{ang}(y, x, p) \leq max_{r\in R}(sim_{ang}(obj, x, r))$

We have shown that the updated object can be pruned. Otherwise its rank either increases or stays the same. Therefore, update operation never decreases the rank of an object. □

THEOREM 5.2. *An object on top of PQ with the current timestamp provides the lower-bound for the ranks of all the objects in PQ, even if there are other objects in PQ with earlier timestamps.*

PROOF. Suppose obj is on top of PQ with the current timestamp, and let obj' be another object in PQ with a prior timestamp ($ts[obj'] < timestamp$). Following Lemma 5.1, even if the rank of obj' is updated, it is either pruned or $rank^+[obj'] \geq rank[obj']$. Since obj' is not on top of PQ, $rank[obj'] \geq rank[obj]$. Hence $rank^+[obj'] \geq rank[obj]$, therefore $rank[obj]$ is still a lower-bound for the ranks of the objects in PQ. □

5.3 Incremental Browsing

After extending the distance browsing feature of R-trees with diverse choices, incremental browsing of an R-tree gives diverse results depending on λ. Details of the method are given in Algorithm 4, excluding the specific conditions, i.e., when $\lambda = 1$ and PQ is empty.

Algorithm 4 Algorithm DiverseKNNSearch

```
1: procedure DiverseKNNSearch($x, k, \lambda, R\text{-}tree$)
2:     $R \leftarrow \{\}$
3:     $ts \leftarrow 0$
4:     $PQ \leftarrow$ MinPriorityQueue()
5:     Enqueue($PQ$, <R-tree.root, $ts$, 0>)
6:     while $|R| < k$ and not IsEmpty($PQ$) do
7:         while Top($PQ$).ts $< ts$ do
8:             $e \leftarrow$ Dequeue($PQ$)                ▷ update top
9:             Enqueue($PQ$, <$e, ts$, GetRank($e$)>)
10:        end while
11:        $e \leftarrow$ Dequeue($PQ$)
12:        if $e$ is a point then                  ▷ add point, inc. ts
13:            $R \leftarrow R \cup \{e\}$; $ts \leftarrow ts + 1$
14:        else                                    ▷ leaf or internal node
15:            for each $obj$ in node $e$ do
16:                Enqueue($PQ$, <$obj, ts$, GetRank($obj$)>)
17:            end for
18:        end if
19:    end while
20:    return $R$
21: end procedure
```

The proposed algorithm has the following properties:

Property 1. Diverse k-nearest neighbor results obtained by the diverse browsing method always contain p_{nn}, the nearest neighbor of the query point x.

PROOF. Initially $R =$, therefore the rank of every object $o_i \in PQ$ is calculated solely depending on the *mindist* of the nodes and the points (see Algorithm 3). The algorithm's behavior is similar to that of distance browsing method at this stage. When the first point p is dequeued from PQ, it is included to R. p is also the point with the minimum distance to x, hence $p_{nn} \in R$. □

Property 2. Diverse browsing can capture the set P_c, which comprises k points uniformly distributed around x with the same distances as p_{nn}.

PROOF. The method selects the points in P_c without pruning any of them. We guarantee that the points in P_c are retrieved without any assumptions on the order. In addition, $\min \widehat{p_i x p_j} = 2\pi/k$ where $p_i, p_j \in P_c$, therefore $[2\pi/k] \geq [2\pi/(k+\epsilon)] = \theta_s$, therefore no point in P_c is pruned. □

Dissimilarity-based diversification methods (e.g. [14]) do not support this property since they are likely to prune some points in P_c, especially when $k > 6$.

6. EXPERIMENTS

We define the evaluation measures in Section 6.1. Real and synthetic datasets used in the experiments are summarized in Section 6.2. Evaluation and discussion of the methods for spatial and high-dimensional datasets are given in Section 6.3.

6.1 Evaluation Measures

In order to measure how well the methods capture the relevancy and the spatial distribution around the query point, evaluation measures are given in Definition 6.1, 6.2, and 6.3.

Definition 6.1. **Angular diversity.** *Given a query point q and a set of results R, angular diversity measures the spatial diversity around the query point:*

$$DIV(q,R) = 1 - \frac{\left\| \sum_{p_i \in R} \frac{\vec{qp_i}}{\|\vec{qp_i}\|} \right\|}{|R|} \tag{5}$$

The intuition behind this measure is that each of the points in R tries to influence the overall result in the direction of the point itself. If the result set is fully diverse, sum of these 'forces' will be closer to the center; therefore the average influence on the query point gives an idea of how diverse the result set is. This measure can be easily applied to higher dimensions since it consists of simple vector additions and normalization. See Figure 5 for the angular diversity of the points in Figure 2.

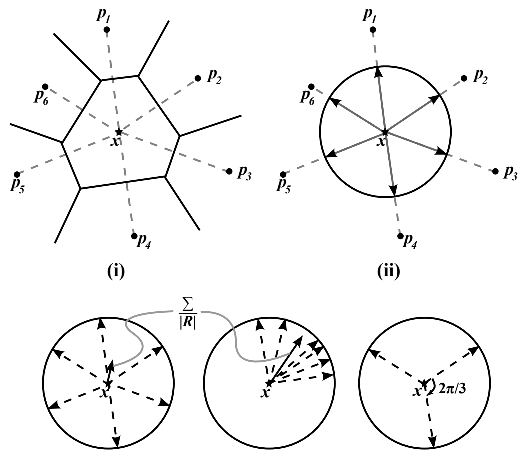

Figure 5: Angular diversity measure. Suppose diverse 6-nearest neighbor search for x retrieves $\{p_1, \cdots, p_6\}$ (i). Angular diversity of the result set is calculated by the sum of vectors $\vec{qp_i}$ on a unit circle/sphere (ii). When the result set is diverse (iii), the average of these vectors will be close to the center; otherwise, as in (iv), the average will be close to the circle. For example, maximum angular diversity ($DIV = 1$) in 2-d for $k = 3$ can be achieved with points which have an angle of $2\pi/3$ pairwise (v).

However, the angular diversity measure is not adequate to evaluate the diversity of a result set, since an algorithm can always return a better set of items more distant than the nearest neighbors if the distance factor is omitted.

Definition 6.2. **Relevance measure.** *Given a query point q, its k-nearest neighbors KNN, and a set of results R, relevance measure calculates the normalized average distance of the points in R with respect to the k-nearest neighbors:*

$$REL(q,R) = \frac{\sum_{p_j \in KNN} \|\vec{qp_j}\|}{\sum_{p_i \in R} \|\vec{qp_i}\|} \tag{6}$$

The result of REL is in the interval [0,1]. Magnitude of each vector is calculated in Euclidean space; although any metric distance measure can be applied to the function, as long as it is consistent with the one used for the calculation of the kNN.

Definition 6.3. **Diverse-relevance measure.** *Given a query point q, a set of results R, and a parameter λ specifying the importance of diversity over relevancy, diverse-relevance measures both relevancy and angular diversity of the results:*

$$DIVREL(q,R) = \lambda \times DIV(q,R) + (1-\lambda) \times REL(q,R) \tag{7}$$

$DIVREL$ is based on the Maximal Marginal Relevance (MMR) method [5]. When $\lambda = 0$, the measure evaluates the relevance of the results excluding the diversity. The aim of our methods is to maximize the diverse-relevance of a result set depending on the value of λ.

6.2 Datasets

We conduct our experiments on both real datasets of points of interests (2-d) and synthetic high-dimensional datasets. The datasets are summarized in Table 1.

Real spatial datasets. We deploy three real-life datasets in our experiments. ROAD is the latitude and longitude data of road crossings in Montgomery County MD, with 63,830 cardinality. 10% of ROAD dataset is randomly selected as query points. North East (NE) dataset contains 123,593 postal addresses, which represent three metropolitan areas (New York, Philadelphia and Boston) [1]. CAL dataset consists of 104,770 points of interest in California [2]. To avoid querying outside of the data region, 500 points from each of two datasets (NE and CAL) are randomly selected as queries.

Synthetic high-dimensional datasets. We generate four synthetic high-dimensional datasets to evaluate the efficiency and the effectiveness of the proposed methods. NORM is 6-d dataset with 500,000 points generated with Normal distribution ($\mu = 0, \sigma^2 = 1$). UNI is the dataset with 6-d and 500,000 cardinality generated with Uniform distribution. SKEW is 6-d dataset with 500,000 points generated with Skew Normal distribution ($\mu = 0, \sigma^2 = 1, \alpha = 1$). HDIM is a 10-$d$ dataset, also generated with Uniform distribution, but has 1,000,000 data points. For each of those

[1]http://www.rtreeportal.org

[2]http://www.cs.fsu.edu/~lifeifei/SpatialDataset.htm

synthetic datasets, 200 query points are produced with the same distribution and parameters.

Table 1: Description of the datasets.

Dataset	d	Card.	Description
ROAD	2	63,830	road crossings, Montgomery County
NE	2	123,593	postal addresses, northeast of USA
CAL	2	104,770	points of interest in California
NORM	6	500K	normal distribution ($\mu = 0, \sigma^2 = 1$)
UNI	6	500K	uniform distribution
SKEW	6	500K	skew normal dist. (μ=0,σ^2=1,α=1)
HDIM	10	1M	uniform distribution

6.3 Evaluations

Spatial real data. We compare the results of geometric approaches (NatN-based and GG-based) with diverse browsing of R-tree and KNDN [14] on ROAD, NE, and CAL datasets. In order to be consistent, the results of the NatN-based method is obtained first. Depending on the number of natural neighbors of each query point, we run other algorithms for each query with $k = \#NatN$. R-trees are built with a page size of 512 bytes (which holds 64 data points) and fill factor of 0.5. Immediate greedy approach of the KNDN method is adopted for the spatial domain: threshold parameter $MinDiv$ is updated according to the value of λ. Note that when $\lambda = 0$, both KNDN and diverse browsing on R-tree methods reduce to kNN.

Similar results for all real datasets (see Figure 6) proves that geometric methods naturally produce diverse results in terms of DIV measure. Since the natural and Gabriel neighbors of a point are fixed in a dataset, these methods are the most efficient ones, only if (1) the purpose of the search is angular diversity, and (2) the spatial database is stable. GG-based method has an advantage over NatN-based method, while it enables for incremental diverse browsing.

However, life itself is always changing. In most cases a spatial index is used to represent certain points-of-interests, and those databases are updated constantly. Roads are built, new restaurants are opened, old buildings are replaced by the new ones, etc. Because geometric methods require a preprocessing time for building the entire DT or GG, they are not appropriate for dynamic databases. In addition, users may want to adjust how diverse vs. relevant the search results are. Index-based methods provide such flexibility. We will discuss the advantages and disadvantages of each method in Section 6.4.

If we focus on index-based search methods, both diverse browsing and KNDN start with $DIVREL \approx 1$ when $\lambda = 0$, and they try to adjust their results as the user asks for more diversity. It is seen that diverse browsing performs better producing a more diverse result set for the spatial domain compared to KNDN (about 20% improvement for $\lambda = 1$, 10% improvement overall). Diverse browsing method also gives a high diverse-relevant set of results as user seek diversity in the results (about 15% to 25% improvement).

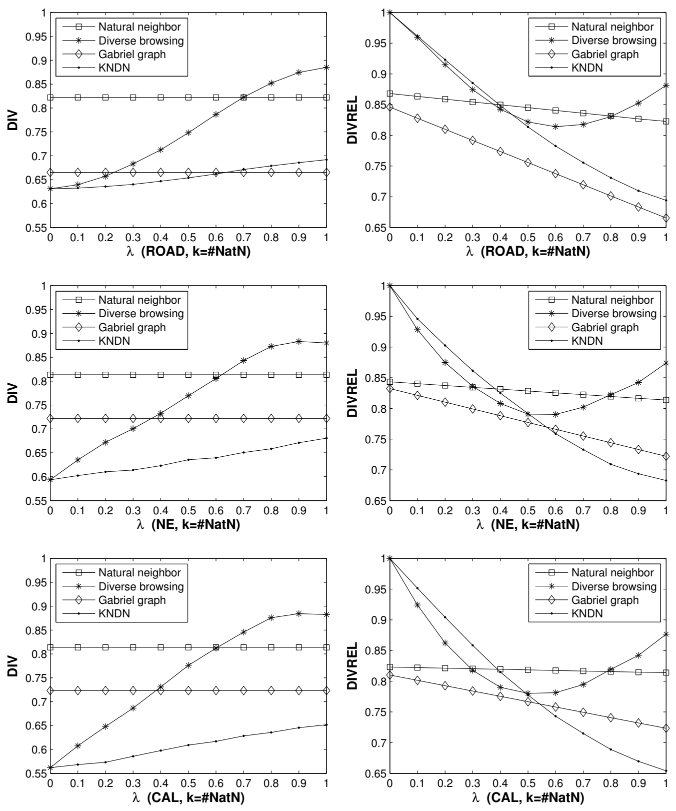

Figure 6: Comparison of the algorithms on ROAD (a,b), NE (c,d), and CAL (e,f) datasets. Aim of the methods is to maximize the diverse-relevance (DIVREL) of the results. Angular diversity (DIV) of the geometric approaches are stable, because natural and Gabriel neighbors of a point are fixed.

Synthetic high-dimensional data. The purpose of experimenting in high-dimensional space is to show the efficiency of each compared algorithm and the diverse-relevance of the results. Figure 7 shows the comparison of diverse browsing, GG-based and KNDN methods on synthetic 6-d datasets. NatN-based method is omitted, because it is not scalable to high dimensions due to its high average degree (see Section 4.3). In order to measure the efficiency of each index-based method, we spot the page accesses when $\lambda = 1$, for which the algorithms investigate the highest number of internal nodes.

For the queries where relevance is preferred over diversity (i.e., $\lambda < 0.5$), diverse browsing and KNDN perform better that GG-based method, since they are both based on distance browsing feature of R-trees. On the other hand, Gabriel graph-based method is extremely powerful for diversity-dominant queries (i.e., $\lambda \geq 0.5$) in terms of both computational efficiency and the diverse-relevance of the results. After retrieving the nearest neighbor p_{nn} in the database, it only takes page accesses equal to the number of layers $l_{GG}(k)$ required to obtain k Gabriel neighbors. From our observations, $l_{GG}(k) \leq 2$ for $k = O(d^2)$. GG-based method also improves the diverse-relevance of the results up to 25% when $\lambda \geq 0.5$.

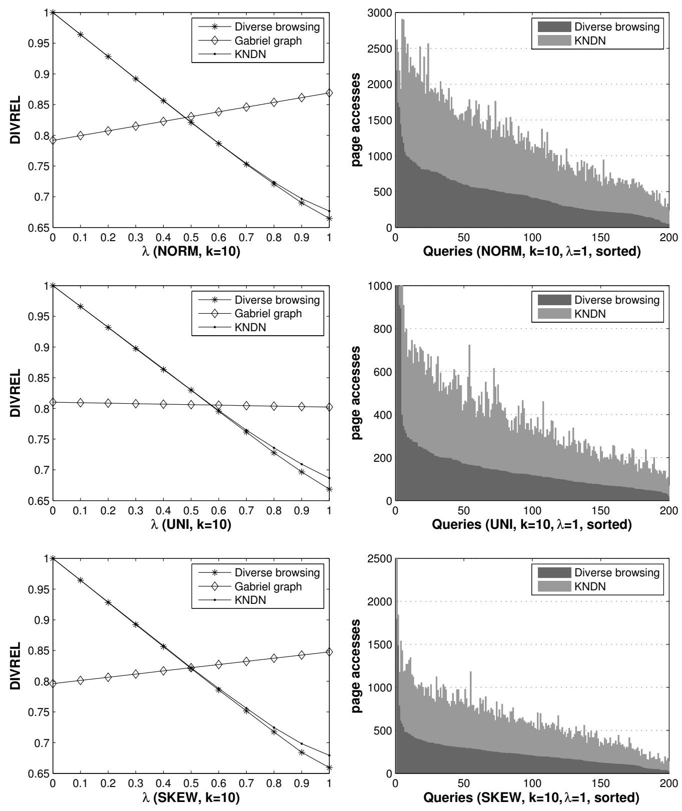

Figure 7: Comparison of the algorithms on NORM (a,b), UNI (c,d), and SKEW (e,f) datasets.

The most challenging dataset we experiment on is 10-d HDIM dataset, which consists of 1M entries. Because generating the GG efficiently in high-dimensions is not the concern of this paper, we decided to extract only the necessary Gabrial-edges for this experiment. Figure 8 suggest that GG-based method is highly effective for diversity-intended queries, where index-based methods return similar results for different λ values. This is obviously because Euclidean distance in higher dimensions may not accurately measure the similarity. But still, diverse browsing method is able to produce the same results as KNDN with 36% less page accesses.

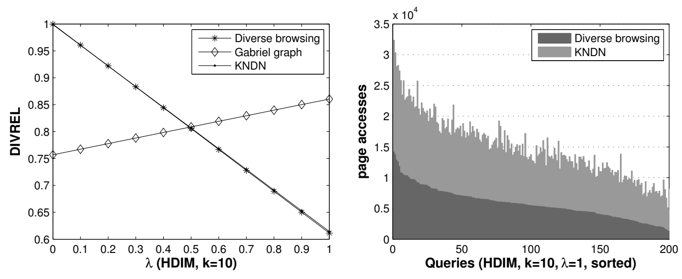

Figure 8: Comparison of the algorithms on HDIM (a,b) dataset.

6.4 Discussions

Proposed geometric and an index-based diverse browsing methods have their own advantages in terms of preprocessing, querying, flexibility, and scalability. A summary of the proposed methods are given in Table 2.

Preprocessing. The advantage of index-based diverse browsing method is that it does not require any preprocessing, and ready to execute on any spatial database that use data partitioning method, such as R-tree, R*-tree, etc. On the other hand, geometric methods require to build DT or GG, which can be very complex depending on dimensionality and cardinality. As a result, we suggest index-based diverse browsing for a dynamic database, which is more likely based an index that handles insert, delete and update operations efficiently; and geometric methods for static databases, which would not cause DT and GG to be calculated frequently.

Querying. As mentioned, NatN-based method naturally returns a result set with a fixed number of points. If the user does not specify k and the purpose is to find a perfectly balanced diverse and relevant set of results (see DIVREL graphs at $\lambda \approx 0.5$), NatN-based method is appropriate. However, diverse browsing is more suitable for diverse k-NN search, which requires exactly k results returned. If the query asks for at least k results, GG-based method can be used as well.

Flexibility. We can investigate this property in two different ways. First is the flexibility of setting the importance of diversity over relevance. Only diverse browsing method adjusts itself for various λ values, since the natural and Gabriel neighbors are fixed in a graph. Second is the flexibility of incremental diverse browsing, where the user demands more search results. Both index-based diverse browsing and GG-based methods enable the retrieval of additional diverse results.

Scalability. For high dimensional spaces, NatN-based method is intractable (see Section 4.3). Since data partitioning methods are shown to be inefficient for high dimensional data, Gabriel graph-based method can be preferred over index-based diverse browsing. Experiments (see Figure 8) show that GG-based method is in fact very efficient ($\sim$10K vs. $l_{GG}(k)$ page accesses) and effective (0.6 vs. 0.85 DIVREL for $\lambda = 1$) in high dimensional datasets.

Table 2: Comparison of the methods.

Method	Results	Ordered	Prep.	Incremental
NatN	#natn	by w_i	build DT	NO
GG	$\geq k$	by layer	build GG	YES
Diverse Browsing	k	by *Rank*	NO	YES

7. CONCLUSIONS

In this work, we investigate the diversification problem in the spatial domain. Because the diverse k-nearest neighbor search is conceptually similar to the idea of natural neighbors, we give a definition of *diversity* by making an analogy with the concept of natural neighbors and propose a natural neighbor-based method. Observing the limitations of NatN-based method in high dimensions, we then present another geometric approach based on Gabriel graphs. For spatial databases, we introduce an index-based diverse browsing method, which maintains a priority-queue with the ranks of the objects depending on both relevancy and diversity, and efficiently prunes non-diverse items and nodes in order to get the diverse nearest neighbors efficiently. To evaluate the diversity of a given result set to a query point, a measure that captures both the relevancy and angular diversity is presented. We experiment on spatial and high dimensional, real and synthetic datasets to observe the efficiency and effectiveness of each proposed method, and compare with another R-tree-based technique found in the literature.

Results suggest that geometric approaches are suitable for static, index-based diverse browsing is for dynamic databases. In addition, Gabriel graph-based method performed well in high dimensions, which can be investigated more and applied to other research fields where search in high dimensional space is required.

Since there are numerous application areas of diverse k-nearest neighbor search, we plan to extend our method to work with different types of data and distance metrics.

8. REFERENCES

[1] P. K. Agarwal, L. Arge, and K. Yi. I/O-Efficient construction of constrained Delaunay triangulations. In *ESA'05*, pages 355–366, 2005.

[2] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM'09*, pages 5–14, 2009.

[3] F. Aurenhammer, T. U. Graz, R. Klein, F. Hagen, and P. I. Vi. Voronoi diagrams. In *Handbook of Computational Geometry*, pages 201–290, 2000.

[4] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, 1996.

[5] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR'98*, pages 335–336, 1998.

[6] H. Chen and D. R. Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *SIGIR'06*, pages 429–436, 2006.

[7] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR'08*, pages 659–666, 2008.

[8] R. A. Dwyer. Higher-dimensional voronoi diagrams in linear expected time. In *SCG'89*, pages 326–333, New York, NY, USA, 1989. ACM.

[9] K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18(3):259–278, 1969.

[10] M. Halvey, P. Punitha, D. Hannah, R. Villa, F. Hopfgartner, A. Goyal, and J. M. Jose. Diversity, assortment, dissimilarity, variety: A study of diversity measures using low level features for video retrieval. In *ECIR'09*, pages 126–137, 2009.

[11] J. R. Haritsa. The KNDN problem: A quest for unity in diversity. *IEEE Data Eng. Bull.*, 32(4):15–22, 2009.

[12] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM Trans. Database Syst.*, 24(2):265–318, 1999.

[13] M. Isenburg, Y. Liu, J. Shewchuk, and J. Snoeyink. Streaming computation of Delaunay triangulations. In *SIGGRAPH'06*, pages 1049–1056, 2006.

[14] A. Jain, P. Sarda, and J. R. Haritsa. Providing diversity in k-nearest neighbor query results. In *PAKDD'04*, pages 404–413, 2003.

[15] H. Ledoux and C. Gold. An efficient natural neighbour interpolation algorithm for geoscientific modelling. In *Proceedings of 11th International Symposium on Spatial Data Handling*, pages 23–25, 2004.

[16] X.-Y. Li, P.-J. Wan, Y. Wang, and O. Frieder. Sparse power efficient topology for wireless networks. In *HICSS'02*, page 296.2, Washington, DC, USA, 2002. IEEE Computer Society.

[17] B. Liu and H. V. Jagadish. Using trees to depict a forest. *Proc. VLDB Endow.*, 2(1):133–144, 2009.

[18] D. Matula and R. Sokal. Properties of gabriel graphs relevant to geographical variation research and the clustering of points on the plane. *Geographical Analysis*, 12:205–222, 1980.

[19] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *SIGMOD'95*, pages 71–79, 1995.

[20] J. R. Shewchuk. *Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996.

[21] R. Sibson. A brief description of natural neighbor interpolation. *Interpolating Multivariate Data*, pages 21–36, 1981.

[22] E. Vee, U. Srivastava, J. Shanmugasundaram, P. Bhat, and S. A. Yahia. Efficient computation of diverse query results. In *ICDE'08*, pages 228–236, 2008.

[23] C. Yu, L. Lakshmanan, and S. Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *EDBT'09*, pages 368–378, 2009.

[24] C. Yu, L. V. S. Lakshmanan, and S. Amer-Yahia. Recommendation diversification using explanations. In *ICDE'09*, pages 1299–1302, 2009.

[25] C.-N. Ziegler and G. Lausen. Making product recommendations more diverse. *IEEE Data Eng. Bull.*, 32(4):23–32, 2009.

[26] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *WWW'05*, pages 22–32, 2005.