Towards Spatially- and Category-Wise k-Diverse Nearest Neighbors Queries

Camila F. Costa(✉) and Mario A. Nascimento

Department of Computing Science, University of Alberta, Edmonton, Canada
{camila.costa,mario.nascimento}@ualberta.ca

Abstract. k-nearest neighbor (k-NN) queries are well-known and widely used in a plethora of applications. In the original definition of k-NN queries there is no concern regarding diversity of the answer set, even though in some scenarios it may be interesting. For instance, travelers may be looking for touristic sites that are not too far from where they are but that would help them seeing different parts of the city. Likewise, if one is looking for restaurants close by, it may be more interesting to return restaurants of different categories or ethnicities which are nonetheless relatively close. The interesting novel aspect of this type of query is that there are competing criteria to be optimized. We propose two approaches that leverage the notion of linear skyline queries in order to find spatially- and category-wise diverse k-NNs w.r.t. a given query point and which return all optimal solutions for any linear combination of the weights a user could give to the two competing criteria. Our experiments, varying a number of parameters, show that our approaches are several orders of magnitude faster than a straightforward approach.

1 Introduction

k-Nearest neighbor (k-NN) queries [14] have been extensively studied by researchers in several areas, such as databases and data mining, information retrieval, pattern recognition and statistics. Namely, given a query point q, a set of points P and a distance metric, a k-NN query finds the set of k points $P' \in P$ such that no other point in $P \setminus P'$ is closer to q than those in P' according to that metric.

One potential drawback of k-NN queries is the fact that the k-NNs are determined based *only* on their distance to the query and no assumption is made on how they relate to each other. Providing homogeneous result sets, i.e., ones where the elements are very similar to each other, may not add much information to the query result as whole. One such scenario is document ranking [3], where returning k documents that are close to the query, but diverse among themselves is likely to increase the amount of overall information one gathers.

This concern has motivated some research to incorporate diversity into similarity searches, i.e., to find elements that are as close as possible to the query,

This research has been partially supported by NSERC, Canada and CNPq's Science Without Borders program, Brazil.

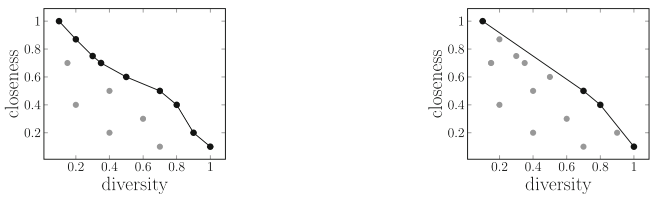
DOI: 10.1007/978-3-319-64367-0_9

while, at the same time, being as diverse as possible. The relation between diversity and closeness has been mainly investigated within the domain of information retrieval, e.g., [3,5], recommendation systems, e.g., [19], web retrieval, e.g., [13] and spatial query processing [11,20].

In this paper we investigate k-Diverse NNs (kDNNs) queries by considering two different notions of diversity, namely spatial and categorical diversities, in typical spatial k-NN queries. In *spatial diversity* the diversity between two data points is given by the distance between them. For instance, a tourist visiting a city may want to explore points-of-interest that are not too far from his/her location, but that at the same time cover different parts of the city. Regarding *categorical diversity*, the diversity is modeled by the difference between categories (or labels) of data points. As an example, consider a user looking for restaurants close by, in which case it may be interesting to return different types of restaurant so that the user could make a decision based on diversified options.

As we shall discuss shortly, previous works that dealt with similar problems, i.e., balancing closeness and diversity, suffer from two main shortcomings: (1) they rely on an user-provided linear combination of the relative importance of closeness over diversity, and (2) they do not provide optimal solutions, given the problem is NP-hard [4]. The approaches we propose in this paper overcome both such shortcomings. In order to find *all* results that are *optimal* under *any* given arbitrary combination of two competing criteria (e.g., closeness and diversity) we rely on the notion of skyline queries [2], more specifically linear skylines [15]. The result set of a skyline query contains elements which are not dominated by any other element. In the context of kDNN queries, a solution is not dominated if there is no one with higher closeness *and* diversity than its own. Linear skyline queries have been proposed as a way to reduce the amount of results to a more intuitive set. Two important properties of linear skylines is that they typically return (1) a much smaller subset of the conventional skyline which (2) are optimal under any linear combination of the competing criteria. The linear skyline obtained from the conventional skyline shown in Fig. 1a can be visualized in Fig. 1b.

The main contributions of this paper are two algorithms to find optimal answers for spatially- and category-wise kDNN queries using linear skylines.



(a) Conventional skyline (b) Linear skyline

Fig. 1. Illustration of conventional vs linear skylines, where the linked dark dots denote the frontier of non-dominated solutions returned by each approach.

In our proposed algorithms the candidate subsets are generated in decreasing order of closeness to the query point, i.e., increasing order of distance. This facilitates checking whether a candidate is part of the skyline or not, since all posterior subsets must have higher diversity than the previous ones in order to be non-dominated. Our extensive experiments, investigating several different parameters, show that our proposals find all optimal solutions for any given linear combination of the optimization criteria several orders of magnitude faster than a straightforward approach.

The remainder of this paper is structured as follows. In Sect. 2 we present a brief discussion of related work. Next we formally define kDNN queries. Our proposed solutions are presented in Sect. 4. The experimental evaluation and results are shown and discussed in Sect. 5. Finally, Sect. 6 presents a summary of our findings and suggestions for further work.

2 Related Work

The concept of incorporating diversity into similarity searches has its origins in information retrieval. The Maximal Marginal Relevance (*MMR*) model [3] is one of the earliest proposals to consider diversity to re-rank documents in the answer set. At each step, the element with higher marginal relevance is selected. A document has high marginal relevance if it is both relevant to the query and has minimal similarity to previously selected documents.

In [18] Vieira *et al.* survey several other approaches, but since they claim that their proposed methods, Greedy Marginal Contribution (*GMC*) and Greedy Randomized with Neighborhood Expansion (*GNE*), are superior (and also due to restricted space) we will review those two only. Similarly to *MMR*, *GMC* constructs the result set incrementally and the elements are ranked based on their maximum marginal contribution (a concept similar to maximal marginal relevance in [3]). In the *GNE* approach, a predefined number of result sets is found and the one that maximizes the optimization function is returned. Differently from *GMC*, in each iteration the element chosen is a random element among the top ranked ones. For each result set R constructed, the algorithm then performs swaps between elements in R and elements that offer a great chance to be more diverse. If in any of these swaps lead to a better result, according to the optimization function, then it is set as the best solution. We choose to compare our proposed approaches to *GNE* because, although it is an approximate solution, it is the one that presented the highest precision w.r.t. the optimal solution (obtained through an exhaustive brute-force approach). Moreover, it can support both types of diversity considered in this paper.

In the context of spatial diversity, several other approaches have been proposed. Jain *et al.* [10] consider the problem of providing diversity in the results of k-NN queries. The result set is constructed incrementally and the elements are analyzed in decreasing order of closeness. Lee *et al.* [12] presented the Nearest Surrounder query, which aims at finding the nearest objects from a query point from different angles. The work proposed by Abbar *et al.* [1] strives to find the most diverse set within a predefined radius in a Hamming space. Kucuktunc *et al.* [11] investigated the diversified k-NN problem based on angular similarity. Finally, Zhang *et al.* [20] studied the problem of diversified spatial keyword search on road networks. These works focused on spatial diversity only, i.e. do not support the categorical diversity dimension considered in this paper. Moreover, all the approaches presented above propose approximate solutions to the problem of optimizing a given linear combination of similarity and diversity. We, on the other hand, find the set of all optimal solutions for any linear combination of these criteria, i.e., the linear skyline, introduced next.

The skyline operator was first introduced in [2]. Given a d-dimensional data set, the skyline query returns the points that are not dominated by any other point. In the context of kDNN queries, a solution S is not dominated if there is no solution S' with higher closeness and diversity than S. One interesting aspect of skyline queries is that the user does not need to determine beforehand weights for closeness and diversity. The skyline query provides the user with a set of multiple and equally interesting solutions in the sense they are all non-dominated, for arbitrary weights. The users then make their decision based on varied options. A drawback of skyline queries is that it may return a large number of points, which may make it difficult for the user to interpret the results and choose the solution that better suits their preference. To overcome this problem [9,16,17] have proposed to reduce the amount of results by focusing on finding the k skyline points that best diversify the skyline result. On the other hand, [15] proposes the more pragmatic concept of linear skyline queries. A linear skyline consists of a relatively small subset of the conventional skyline which is optimal under *all* linear combination functions of the criteria to be optimized. Thus, in this paper we rely on the notion of linear skylines.

3 Problem Definition

The k-Diverse Nearest Neighbor (kDNN) problem incorporates diversity into the traditional k-NN problem. The goal is to find a set of solutions with k elements that are close to the query q while being as diverse among them as possible.

Let $P = \{p_1, p_2, \dots p_n\}$ be a set of data points. Throughout this paper we refer to the closeness of a point $p_i \in P$ w.r.t the query q as the opposite of their Euclidean distance $d(p_i, q)$. Thus maximizing the closeness of p_i w.r.t q is the same as minimizing $d(p_i, q)$. Using this definition, the distance from q to a subset $S \subseteq P$ is defined as the maximum distance between q and any point in S, i.e., $d(q, S) = \max\limits_{s_i \in S}\{d(q, s_i)\}$.

We consider two different notions of diversity in this paper: spatial and categorical. In spatial diversity, also adopted in [7,20], the diversity between two points $p_i, p_j \in P$ is given by the distance between them, i.e., $div(p_i, p_j) = d(p_i, p_j)$. The idea is to find points that are well spatially distributed and consequently cover different areas.

Regarding categorical diversity, we assume that there is a set of categories $G = \{g_1, g_2, ..., g_{|G|}\}$ and that each point $p_i \in P$ is labeled with a category denoted by $g(p_i) \in G$. Note that categories are non-exclusive, i.e., more than one point may belong to the same category. We model the diversity between categories as a matrix $M_{|G|\times|G|} = (m_{p,q})$, and for simplicity we assume that $0 \leq m_{p,q} \leq 1$ and in particular $m_{p,q} = 0$ if $p = q$. We can then represent the diversity between two categories g_p and g_q by the element $m_{p,q}$ in M. The diversity between a pair of points $p_i, p_j \in P$ is then given by $m_{g(p_i),g(p_j)}$, i.e., the diversity between their categories. Finally, the diversity within a set $S \subseteq P$ is defined as the minimum pairwise diversity between the points of S, i.e., $div(S) = \min_{(p_i,p_j)\in S \wedge (p_i \neq p_j)} \{div(p_i, p_j)\}$.

Let us now define the notions of conventional and linear skyline domination.

Definition 1. *Let R and T be subsets $R, T \subseteq P$ of size k. Then, in the context of kDNN queries, R dominates T, denoted as $R \prec T$, if*

$$\begin{aligned} d(q,R) < d(q,T) \quad &and \quad div(R) \geq div(T) \quad or \\ d(q,R) \leq d(q,T) \quad &and \quad div(R) > div(T) \end{aligned}$$

That is, R is better in one criteria and at least as good as T in the other one. Thus the conventional skyline, i.e., the set of non-dominated subsets, is given by

$$\{S \subseteq P \ s.t. \ \nexists R \subseteq P : R \prec S, |S| = |R| = k\}.$$

In our proposed approaches we first find the conventional skyline and then, in a posterior step, we extract the linear skyline from the conventional one.

Definition 2. *Let SK be a conventional skyline. A subset $SK' \subseteq SK$ linearly dominates a solution $T \in SK$, denoted as $SK' \prec_{lin} T$, iff*

$$\forall w \in \mathbb{R}^2 \ \exists R \in SK' : w^T r > w^T t$$

where w is a 2-dimensional weight vector and r (t) is a vector representing the closeness and diversity of R (T). The maximal set of linearly non-dominated solutions is referred to as linear skyline [15].

Note that in contrast to conventional domination, testing for linear domination requires comparing a solution to more than one other solution.

Finally, the problem addressed in this paper can now be defined as follows.

Problem Definition. *Given a query q, a positive integer k and a set of data points P, the k-Diverse NN (kDNN) query aims at finding the set of all k-sized linearly non-dominated subsets $S \subseteq P$.*

4 Proposed Solutions

As our main contribution in this paper we present two algorithms to the optimal kDNN problem using linear skylines. Both algorithms find the conventional skyline with the linear skyline being extracted in a posterior step. They analyze the candidate subsets in increasing order of $d(q, S)$, which means that the following subsets necessarily need to have a higher diversity than the previous ones in order to be non-dominated. This allows us to establish a lower bound to the diversity of the following non-dominated result sets. Throughout this paper this lower bound is denoted *minDiv*. Both proposed solutions, referred to as *Recursive Range Filtering* (*RRF*) and *Pair Graph* (*PG*), use such lower bound to filter out dominated subsets. *RRF* works recursively by combining a partial solution S' with points in a candidate set C, one at time, that have a higher diversity to S' than *minDiv*. *PG* strives to reduce the number of generated combinations by pruning subsets that contain pairs of elements $(p_i, p_j) \subseteq P$ such that $div(p_i, p_j) \leq minDiv$ and therefore could not be in a non-dominated solution.

Before introducing *RRF* and *PG* we establish as a comparison baseline a straightforward, brute-force algorithm (*BF*). *BF* generates all possible subsets $S \subseteq P$ of size k in an arbitrary order. If S is not dominated by any other solution in the *skyline* set, then S is added to *skyline*. When a new solution S is added to *skyline*, we check if no previous solutions in *skyline* are dominated by S. If such solutions exist, they are removed from *skyline*.

Our proposed solutions, *RRF* and *PG*, are general in the sense that they work for both categorical and spatial diversity. However, note that for categorical diversity only the closest points from each category are needed in order to find the whole skyline set. Farthest points would lead to solutions with lower closeness and that would not have higher diversity, since the diversity between two points is determined by their categories. Therefore, such solutions would be dominated. This means that the algorithms can stop once the closest points from each category have been found. *RRF* and *PG* are both based on Algorithm 1, presented next. They differ in how they find the set with maximum diversity for a fixed closeness, which is done by the method *findSetMaxDiv* used within Algorithm 1. Since each of our proposed solutions has its own implementation of *findSetMaxDiv*, they are presented in turn next.

Algorithm 1 takes as input a query point q, the size of the answer sets k and the set of data points P. The first subset S generated is the closest one, i.e., the one with minimum $d(q, S)$ (line 1). The elements $s_i \in S$ are removed from P (line 2) and added to the set of previously examined points P' (line 3). S is then added to *skyline*, since it is guaranteed to be a non-dominated solution, and $minDiv$ is set to $div(S)$. The variable $minDiv$ represents the minimum diversity that the following solutions must have in order to be non-dominated. All subsets S' with $div(S') \leq minDiv$ are discarded since they are dominated solutions and do not belong to the *skyline*. Next, the remaining elements $p_d \in P$ are examined in increasing order of $d(q, p_d)$ (lines 6 and 7). For each p_d, the method $findSetMaxDiv$ looks for the set S with maximum $div(S) > minDiv$ such that $p_d \in S$ and $S \setminus p_d \subseteq C$. Where C is a subset of P' containing all $p_i \in P'$ such that $div(p_d, p_i) > minDiv$ (line 9). This step eliminates the elements that can not be combined with p_d in order to obtain a non-dominated solution.

Note that $d(q, S) = d(q, p_d)$ since p_d is the farthest point from q in S. Also, for a particular p_d, all subsets examined by *findSetMaxDiv* have the same distance, since they necessarily contain p_d. This implies that for each p_d there is up to one non-dominated solution, which is the one with highest diversity. At each step, the current p_d is removed from P and added to P'. Moreover, if a non-dominated solution S is found by *findSetMaxDiv*, S is added to *skyline* and *minDiv* is updated to $div(S)$ (lines 11 to 14).

Algorithm 1. MinDistMaxDiv

```
   Data: Query $q$, set of data points $P$ and result set size $k$
   Result: All non-dominated solutions $S \subseteq P$, $|S| = $ k
1  $S \leftarrow k\text{-}NN(q, P)$;
2  $P \leftarrow P \setminus S$;
3  $P' \leftarrow S$;
4  $skyline \leftarrow S$;
5  $minDiv \leftarrow div(S)$;
6  while $|P|$ ¿ 0 do
7      $p_d \leftarrow \operatorname{argmin}_{p_i \in P}(dist(q, p_i))$;
8      $P \leftarrow P \setminus p_d$;
9      $C \leftarrow$ all $p_i \in P'$ such that $div(p_d, p_i) > minDiv$;
10     $S \leftarrow findSetMaxDiv(args)$;
11     if $S$ is not null then
12         $skyline \leftarrow skyline \cup S$;
13         $minDiv \leftarrow div(S)$;
14     end
15     $P' \leftarrow P' \cup p_d$;
16 end
17 return skyline;
```

Next, we prove that the set of solutions generated by Algorithm 1 is correct and complete. For this we assume that the *findSetMaxDiv* method is correct and finds the corresponding non-dominated solution for a given $p_d \in P$, if such solution exists. We prove such assumption, i.e., the correctness of the *findSetMaxDiv* method used in each of our proposed solutions in the following subsections, when *RRF* and *PG* are presented.

Theorem 1. *The skyline set does not contain dominated solutions.*

Proof. The first solution S found by Algorithm 1 is the closest one and thus is not dominated. All consecutive solutions can not dominate previous solutions in terms of closeness, since they are found in increasing order of distance. Therefore, those solutions must have a higher diversity than minDiv, the highest diversity found so far, in order to be non-dominated. For each point $p_d \in P$, a new solution S is returned by the $findSetMaxDiv$ method only if $div(S) > minDiv$.

Moreover, $findSetMaxDiv$ finds the corresponding non-dominated solution for p_d, i.e., the one with highest diversity, if such solution exists. Therefore, no dominated solutions can be part of skyline. □

Theorem 2. *All non-dominated solutions are in the skyline set.*

Proof. Let us suppose by contradiction that there is a non-dominated solution S that is not part of the skyline. Assuming that $d(q, S) = d(q, p_d)$, $p_d \in P$, implies that S was not found by the $findSetMaxDiv$ method when p_d was the current element with minimum distance in P. This is a contradiction since $findSetMaxDiv$ is guaranteed to find the corresponding non-dominated solution for any $p_d \in P$ if such solution exists. (This last statement is proven in the following, in the context of each proposed method.) □

4.1 Recursive Range Filtering (RRF)

The *findSetMaxDiv* method used in the *RRF* solution, which in this context we denote as *RRF-findSetMaxDiv* to remove any ambiguity, assumes that p_d, the point in P with minimum distance, is part of the non-dominated result set S, if such set exists. Also, as explained above, p_d is the farthest element from q in S. A candidate set C, containing points that can be combined with the partial set $S' = \{p_r\}$ in order to obtain a non-dominated solution, is given as input to *RRF-findSetMaxDiv*. One element c_i from C is chosen at a time to also be part of S'. Elements that can not be combined with c_i are removed from C and the algorithm is recursively called for the new partial set S' and candidate set C.

Algorithm 2 shows how the *RRF-findSetMaxDiv* method works. It takes as input a partial set S', which initially only contains the point p_d, a set C of candidate points that can be combined with S', the number n of elements that need to be added to S' to form a set of size k and the current set with maximum diversity $setMaxDiv$, which is initially empty. If $n > 1$ (line 9), the candidates c_i in C are examined one at a time. More specifically, we check whether the candidate set C_i obtained after adding c_i to S' contains at least $n - 1$ points (line 14). If so, the function is recursively called for the new partial set S'_i and candidate set C_i (line 16). Otherwise, it is not possible to obtain a non-dominated solution S such that $S'_i \subseteq S$ and therefore S'_i can be discarded. Note that we avoid creating the same subset more than once by not adding previously examined candidate points $c_j \in C$ such that $1 \leq j < i$ to C_i (lines 11 and 12). That is because all the possible sets containing $S' \cup c_j$ have already been created, including the ones that contain c_i, and adding c_j to the candidate set C_i would just create duplicates and make the algorithm less efficient.

In the base case (line 1), only one point still need to be added to S'. We then simply create a solution S for each $c_i \in C$ and set the one with highest diversity to $setMaxDiv$. When no other set of size k can be built, the algorithm returns $setMaxDiv$ (line 21).

Figure 2 illustrates one iteration of Algorithm 2 for $k = 4$, $p_d = p_6$ and considering spatial diversity. The points $P' = \{p_1, p_2, p_3, p_4, p_5\}$ have already been

Algorithm 2. *RRF-findSetMaxDiv*

```
Data: Partial set $S'$, candidate set $C$, $n = k - |S'|$, $setMaxDiv$ and $minDiv$
1  if $n$=1 then
2      for $c_i \in C$ do
3          $S \leftarrow S'$;
4          $S \leftarrow S \cup c_i$;
5          if $div(S) > div(setMaxDiv)$ then
6              $setMaxDiv \leftarrow S$;
7          end
8      end
9  else
10     for $c_i \in C$ do
11         if $|C| - i - 1 \geq n - 1$ then
12             $C_i \leftarrow C[i+1, |C|]$;
13             $C_i \leftarrow$ all $c_j \in C_i$ such that $div(c_i, c_j) > minDiv$;
14             if $|C_i| \geq n - 1$ then
15                 $S'_i \leftarrow S' \cup c_i$;
16                 $setMaxDiv \leftarrow$
                     $findSetMaxDiv(S'_i, C_i, n-1, setMaxDiv, minDiv)$;
17             end
18         end
19     end
20 end
21 return $setMaxDiv$;
```

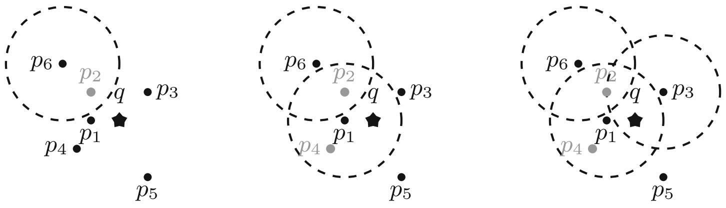


Fig. 2. *RRF-findSetMaxDiv* execution for $p_d = p_6$, $k = 4$ and spatial diversity.

examined and the following skyline solutions have been generated $skyline = \{(\{p_1, p_2, p_3, p_4\}, d = 0.9, div = 0.5), (\{p_2, p_3, p_4, p_5\}, d = 1.12, div = 1)\}$. Therefore at this point $minDiv = 1$.

The method takes as input $S' = \{p_6\}$, $C = \{p_1, p_3, p_4, p_5\}$, $n = 3$ $(k-1)$ and $setMaxDiv = \emptyset$. Note that, as shown in Fig. 2, p_2 can not be combined with p_6 and thus is not part of C. The first candidate point p_1 is added to S', which discards p_4 from C. The algorithm is then recursively called for $S' = \{p_6, p_1\}$, $C = \{p_3, p_5\}$. The candidate point p_3 is added to S' and the only remaining candidate is $C = \{p_5\}$. The base case, where $n = 1$, is reached and a solution is created for each point in C. The solution $S_1 = \{p_6, p_1, p_3, p_5\}$ with $div(S_1) = 1.12$ is built and becomes the current $setMaxDiv$. When the second candidate point of $C = \{p_3, p_5\}$, i.e., p_5, is considered to be added to $S' = \{p_6, p_1\}$ (omitted in the example), the corresponding candidate set will be empty. As explained above, this avoids creating the same set S_1 twice. Next, the second candidate from $C = \{p_1, p_3, p_4, p_5\}$ is added to $S' = \{p_6\}$, creating the new candidate set $C = \{p_4, p_5\}$. Note that p_1 is not added to C because all the possible sets containing p_1 have already been generated. The solution $S_2 = \{c_3, c_4, c_5, c_6\}$ with $div(S_2) = 1.35$ is found and becomes the current *setMaxDiv*. When p_4 and p_5 are considered to be added to $S' = \{p_6\}$ (omitted in the example), they are pruned because there are not enough points to be combined with them in order to create a non-dominated solution that has not yet been generated.

Theorem 3. *For any p_d, the element in P with minimum distance, the method RRF-findSetMaxDiv finds the corresponding non-dominated solution S, where $dist(q, S) = dist(q, p_d)$, if such a solution exists.*

Proof. Let us assume that S exists and the method does not find it. This means that at least one $s_i \in S$, $s_i \neq p_d$ has been discarded by RRF-findSetMaxDiv. Let us analyze the two possible cases:

1. *$div(s_i, s'_j) \leq minDiv$ holds for at least one $s'_j \in S'$, $S' \subseteq S \setminus s_i$.*
   This is a contradiction to the assumption that S is a non-dominated solution and thus $div(s_i, s_j) > minDiv$ for all $s_i, s_j \in S$.
2. *There is no subset $S' \subseteq S \setminus s_i$ such that $S' \cup s_i$ can be combined with at least $n = k - |S'| - 1$ points.*
   This is also a contradiction to the fact that S is a solution with k points and thus $S' \cup s_i$ can be combined with $S \setminus \{S' \cup s_i\}$.

Thus, such set S must be found by the method RRF-findSetMaxDiv. □

4.2 Pair Graph Solution (PG)

The idea behind the *PG* solution is to combine pairs of elements (p_i, p_j) such that $div(p_i, p_j) > minDiv$ in order to construct sets of size k that can potentially be a non-dominated solution. Two pairs can be combined if they have an element in common. For instance, (p_i, p_j) can be combined with (p_j, p_l) if $div(p_i, p_j) > minDiv$ and $div(p_j, p_l) > minDiv$, obtaining the set $S_3 = \{p_i, p_j, p_l\}$ of size three. Similarly, S_3 can be combined with a pair (p_l, p_m) to obtain the set $S_4 = \{p_i, p_j, p_l, p_m\}$ of size four if $div(p_l, p_m) > minDiv$. However, note that there is no guarantee that $div(S_4) > minDiv$, unless $div(p_i, p_l) > minDiv$, $div(p_i, p_m) > minDiv$ and $div(p_l, p_m) > minDiv$.

A non-dominated solution $S \subseteq P$ for which the candidate point p_d is the farthest one from q, can only contain previously examined elements $p_i \in P'$ such that $div(p_d, p_i) > minDiv$, i.e., the points in the candidate set C. Therefore, the set of pairs $\mathcal{P}$ needed for following the strategy aforementioned are all (p_d, p_i) for $p_i \in C$ and all (p_i, p_j) for $p_i, p_j \in C$.

In order to facilitate the construction of sets of size greater than two by combining the pairs in $\mathcal{P}$, we build a graph $G(V, E)$, where the set of vertices is $V = C \cup p_d$. Also, for each pair $(p_i, p_j) \in \mathcal{P}$, we create an edge (p_i, p_j) and add it to the set of edges E. The cost of an edge (p_i, p_j) is given by $div(p_i, p_j)$. Finding sets of size k is then equivalent to finding paths with k vertices in G.

Figure 3 shows the graph G built for the example shown in Fig. 2, i.e., for $p_d = p_6$, $k = 4$ and spatial diversity. Since all the sets of size $k = 4$ must contain p_6, in order to find paths in G that may represent a non-dominated solution, we assume that the starting point is p_6. Note that the possible paths of size 4 starting from p_6 are $PT_1 = (p_6, p_1, p_3, p_5)$, $PT_2 = (p_6, p_3, p_4, p_5)$ and $PT_3 = (p_6, p_1, p_3, p_4)$. The diversity of PT_1 is 1.14 because this is the minimum cost of an edge between any two vertices of PT_1. Since $1.14 > 1 = minDiv$, PT_1 is a candidate to be added to skyline. Similarly, the diversity of PT_2 is 1.35, which makes PT_2 a better candidate than PT_1. However, PT_3 does not have diversity higher than $minDiv$, since there is no edge (c_1, c_4) in E and thus $div(c_1, c_4) \leq minDiv$. Therefore, PT_2 represents the set with highest diversity.

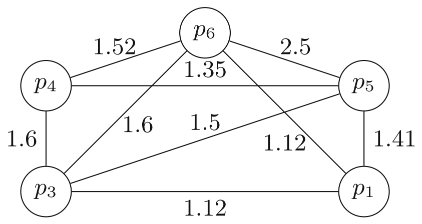


Fig. 3. Graph G built for the example shown in Fig. 2.

Algorithm 3 shows all steps of the *PG-findSetMaxDiv* method. In order to find paths of a given size k, we use a solution based on Dijkstra's classical algorithm. First, all edges $(u, v) \in E$, representing paths of size 2, are added to a priority *queue* (lines 5 to 7). The paths PT in *queue* are ordered by decreasing order of an upper bound $ub(PT)$ that represents an optimistic value to the diversity of a set of size k that contains the vertices of PT. Let us assume that the size of PT is $p \leq k$ and that $l = k - p$. As exemplified above, in order to obtain a non-dominated solution of size k that contains the vertices of PT, PT can only be extended by adding l vertices that are neighbors of all vertices of PT. Let MN be this set of mutual neighbors. If $|MN| < l$, then it is not possible to find a non-dominated solution that contains the vertices of PT, therefore PT can be discarded. Otherwise, for each $pt_i \in PT$ let $div^l(pt_i, mn_j)$ be its l-th largest diversity to an element $mn_j \in MN$. We refer to the minimum of these values as $div(PT, MN)$. Since $div(PT, MN)$ is an upper bound to the diversity of PT to l other elements, if $div(PT, MN) \leq minDiv$, PT can also be discarded. The same occurs if $div(PT) \leq minDiv$. Therefore, the upper bound $ub(PT)$ to the diversity of a set S of size k containing the vertices of PT is given by $\min\{div(PT), div(PT, MN)\}$.

At each step the path PT with highest $ub(PT)$ is removed from *queue* and expanded if it contains less than k vertices (lines 9 and 12). Let v be the last

Algorithm 3. *PG-findSetMaxDiv*

```
Data: Candidate set $C$, result set size $k$, point $p_d$ and $minDiv$
1  $\mathcal{P} \leftarrow$ all pairs $(p_d, p_i)$, $p_i \in C$;
2  $\mathcal{P} \leftarrow \mathcal{P} \cup$ all pairs $(p_i, p_j) \subset C$, $div(p_i, p_j) > minDiv$;
3  $G(V, E) \leftarrow buildGraph(\mathcal{P})$;
4  $queue \leftarrow \emptyset$;
5  for $(v, u) \in E$ do
6      $queue$.add($(v, u)$);
7  end
8  while $|queue|$ ¿ $0$ do
9      $PT \leftarrow queue.dequeue()$;
10     if $|PT| = k$ then
11         return $PT$;
12     else if $|PT|$ ¡ $k$ then
13         $v \leftarrow PT[|PT|]$;
14         for $(v, u) \in E$ do
15             if $ub(PT \cup u) > minDiv$ then
16                 $queue$.add($PT \cup u$);
17             end
18         end
19     end
20 end
21 return $null$;
```

vertex in the path PT. For each neighbor u of the vertex v, PT is extended by adding u. If $ub(PT \cup u) > minDiv$ then the path $(PT \cup u)$ is added to the queue (lines 15 and 16). If PT has size k (line 10), PT is returned. Note that no other posterior path PT' of size k could have higher diversity than PT. Since PT' is not the first set of size k removed from *queue*, then $ub(PT') \leq ub(PT)$, which means that in an optimistic scenario the diversity of PT' is at most equal to $div(PT)$. If no path of size k is found, the algorithm returns *null*.

Theorem 4. *For any p_d, PG-findSetMaxDiv finds the non-dominated solution S, where $d(q, S) = d(q, p_d)$, if such a solution exists.*

Proof. We divide this proof into two parts. First we prove that the algorithm does not remove any pair of points necessary for finding S, if S exists. Then we show that given all the necessary pairs, PG-findSetMaxDiv finds S.

Let us suppose that S exists and the algorithm discards a pair $(s_i, s_j) \subset S$. There are two possible cases:

1. *$s_i = p_d$ or $s_j = p_d$*
   This means that PG-findSetMaxDiv discarded a pair containing p_d. Since S is a non-dominated solution, $div(p_d, s_p) > minDiv$ for any $s_p \in S, s_p \neq p_r$. However, all pairs (p_d, s_p) such that $div(p_d, s_p) > minDiv$ are maintained by our algorithm, a contradiction to the assumption that (s_i, s_j) was discarded.

2. *$s_i \neq p_d$ and $s_j \neq p_d$*
   The pair (s_i, s_j) is only discarded by the algorithm if $div(p_d, s_i) \leq minDiv$ and/or $div(p_d, s_j) \leq minDiv$. Since $\{p_d, s_i, s_j\} \subseteq S$ and S is a non-dominated solution, $div(p_d, s_i) > minDiv$ and $div(p_d, s_j) > minDiv$, a contradiction.

Therefore no necessary pair is discarded.

Next We need to prove that PG-findSetMaxDiv is able to find S. We do so, again by contradiction, assuming that $S = (s_1, s_2, ..., s_i, s_{i+1}, ..., s_k)$ exists and it is not found. This means that S is not removed from the queue in the method. This happens if, during the expansion of a s_i, $2 \leq i < k$, the partial solution $S' = (s_1, s_2, ..., s_i, s_{i+1})$ is not inserted in the queue, which in turn occurs in the following situations:

1. *There is no edge $(s_i, s_{i+1}) \in E$.*
   This is a contradiction since as proven above, no pair $(s_i, s_j) \subseteq S$ is discarded by the method and consequently there is an edge $(s_i, s_{i+1}) \in E$.
2. *$ub(S') \leq minDiv$.*
   Therefore $div(S') \leq minDiv$ and/or $div(S', MN) \leq minDiv$. Since $S' \subseteq S$, $div(S') \geq div(S)$. As S is a non-dominated solution, $div(S) > minDiv$ and consequently $div(S') > minDiv$, a contradiction. If $div(S', MN) \leq minDiv$ then there is no non-dominated solution of size k that contains S'. A contradiction to the fact that S is non-dominated and $S' \subseteq S$. □

4.3 From Conventional Skylines to Linear Skylines

In order to extract the linear skyline from the conventional one we follow a strategy based on the one proposed in [15]. In that paper the authors find linearly non-dominated paths in a bi-criteria road network. However, their goal was to minimize two competing costs associated with each path, while we aim at maximizing both closeness and diversity of solutions. Moreover, in that paper the linear skyline is constructed during the network expansion, while we first find the conventional skyline and then filter the linear skyline from it.

Due to lack of space we omit the details of the extraction process but summarize it as follows. Let SK be the conventional skyline, a solution $D \in SK$ is linearly dominated by two solutions $R, T \in SK$ if the 2-dimensional cost vector d representing D lies below the line between r and t, or more formally: let n be the normal vector of the line between R and T, such that $n^T r = n^T t$, then $\{r, t\} \prec_{lin} d$ iff $n^T d < n^T r = n^T t$ [15].

The procedure for extracting the linear skyline from SK then consists of eliminating all the points of SK that lie below a line between any two other skyline points [15]. More specifically, the solutions from SK are inserted into a list LS, that represents the linear skyline, in increasing order of diversity. After, let us assume that LS contains i elements and a new solution S^{i+1} is inserted at position $i + 1$. We verify whether this insertion leads to the removal of other solutions already in LS by performing a left traversal in LS. We first check if the left neighbor of S^{i+1}, i.e., S^i, is linearly dominated by verifying whether $\{s^{i-1}, s^{i+1}\} \prec_{lin} s^i$ holds. If so, S^i is removed from LS and S^{i-1} becomes the left neighbor of S^{i+1}. We then check whether $\{s^{i-2}, s^{i+1}\} \prec_{lin} s^{i-1}$. This process will be terminated when the first element of LS has been examined or the current left neighbor of S^{i+1} is a linearly non-dominated solution.

5 Experiments

We performed experiments using synthetically generated data sets with different distributions and sizes. (Even though real datasets could be used, synthetic datasets allowed us to investigate our proposal with respect to a broad range of parameters.) We first investigate how our proposed solutions perform in comparison to the brute force solution (*BF*) mentioned in Sect. 4. Then we show how our *PG* and *RRF* compare to the previously proposed approximate approach *GNE* in terms of processing time and quality of the results. The algorithms were developed in Java and the experiments were conducted on a Linux-based virtual machine with 4 vCPU and 8 GB main memory.

Given the kDNN problem's NP-hardness [4], and similarly to what was done in previous works, e.g., [1,6,18], we assume that the data set given as input to our algorithms is a candidate set $CS \subseteq P$ with elements that are relevant to the user[1]. This filtering step discards elements that are far from the query point and would not add any value to the results.

In the experiments that follow we varied the following parameters: the number of candidate points $|CS|$ between 100 and 1000 (with 500 being the default), the number k of points in each subset between 3 and 10 (with the default being 5) and the data distribution as Uniform (default), Gaussian and Clustered. The clustered data set was created using the generator described in [8] based on multivariate normal distribution. The data set generated contains three clusters with roughly the same number of points. As usual, when varying one parameter the others were kept at their default values.

Finally, it is important to stress that (1) all the graphs shown next present the average results in logarithmic scale, and (2) if an algorithm is not able to find the skyline in less than 10 min the computation is aborted and (simplistically) assumed to take 10 min when computing the average.

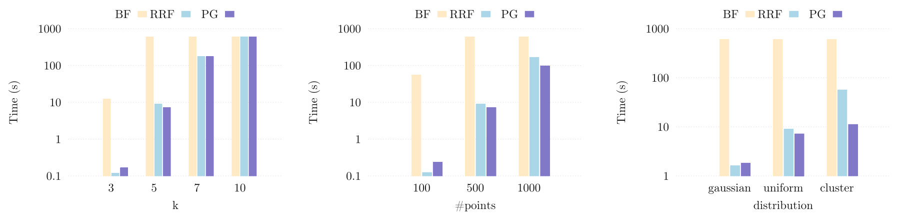
5.1 Spatial Diversity

We first evaluated how our proposed solutions behave when the spatial diversity is considered, i.e. when the dissimilarity between two points is given by their Euclidean distance. Figure 4 shows the results of this experiment.

Effect of k. As expected and shown in Fig. 4a, the running time of all solutions increases with k. However, the performance of *BF* degrades much faster since it generates all the possible subsets of size k in order to find the ones that belong to

[1] For simplicity we assume that CS is the set of $|CS|$-NN wrt the query point.

the skyline. For $k > 3$ *BF* is not able to find the skyline in less than 10 min. *RRF* showed to be around 30% faster than *PG* for $k = 3$ and slower for larger values. Most of the *RRF*'s execution time is spent in the filtering process. For each point added to a partial set, a range query centered at that point is performed in order to discard points that can not be combined with the current partial set in order to obtain a non-dominated solution of size k. Therefore, the larger the value of k, the greater the number of range filtering performed. Moreover, *RRF* finds all the possible sets of size k with diversity higher than $minDiv$. On the other hand, in the *PG* approach, partial subsets that offer a greater chance to be part of the solution with maximum diversity are expanded first. Once the first solution of size k is found, the algorithm stops. This pruning is more effective for larger values of k. As shown in Fig. 4a, for $k = 3$ the gains from such pruning are not outweighed by the extra cost of computing the upper bound to the diversity of partial subsets and the costs involved in the queue operations.



(a) Varying k (b) Varying $|CS|$ (c) Varying the data distribution

Fig. 4. Running time for spatial diversification.

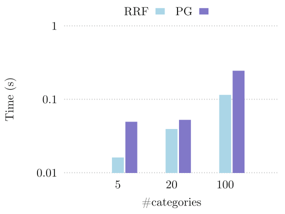
Effect of the number of candidate points. Figure 4b shows that the running time of all solutions also increases with the number of candidate points. This is simply because there is a larger number of subsets that could be part of the skyline, requiring the algorithms to examine more combinations. As can be seen, *BF* performs poorly even for a small number of points. Similar to the previous experiment, *RRF* performs better for a lower number of points, while *PG* is more efficient when more points are considered for the same reason explained above.

Effect of the data distribution. As shown in Fig. 4c, *BF* is not affected by the data distribution since it generates all combinations of size k regardless of how the data is distributed. On the other hand, *RRF* and *PG* perform better for a Gaussian distribution. Since there is a higher concentration of points around the mean of the distribution, a range around a point in a dense area will include more points, which in turn will be eliminated from the result set and consequently less combinations will need to be tested. In an uniform distribution the points are more scattered in space, which implies that a range will likely include less points, increasing the number of possible combinations that need to be checked. Finally, in the clustered distribution there is more than one concentration of points, which means that, although a range around a point will probably eliminate points in the same cluster, it will most likely not include points from other clusters. Therefore, since the diversity between points in different clusters will tend to be high, the number of subsets that can potentially be a non-dominated solution will also increase, requiring more combinations of points to be generated and checked.

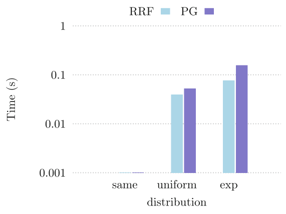
5.2 Categorical Diversity

Next, we evaluated how our proposed approaches behave for categorical diversity varying the number of categories and the distribution of the diversities between two categories. We have considered data sets with 5, 20 and 100 categories, where the default value is 20. Regarding the diversity distribution, we have generated diversity matrices where the diversity between two categories is either the same for any pair of categories, follows an uniform distribution or an exponential distribution. The uniform distribution is used as default. In the following experiments we do not report the results for *BF* since it is not affected by these parameters and was shown above to be a non-competitive approach.

Effect of the number of categories. As shown in Fig. 5a, the running time of both approaches increases with the number of categories. Since for categorical diversity only the closest points from each category are needed, the higher the number of categories, the higher the number of points that need to be examined in order to find the skyline set. Also note that for the same reason, the execution time is much shorter than when a spatial diversity is considered. Moreover, *RRF* outperforms *PG* for all examined values. This can be explained by the fact that *RRF* is more efficient for a smaller number of points, as shown in a previous experiment, and only up to 100 points, i.e., the number of categories, are needed in this experiment in order to find the skyline set.



(a) Varying #categories.



(b) Varying the distribution of diversity.

Fig. 5. Running time for categorical diversification.

Effect of the distribution of diversities. Figure 5b shows that both solutions present better performance when the diversity between any pair of categories is the same. This is simply because once a solution where no two points belong to the same category is found, it is not possible to find a posterior solution with higher diversity. Thus the range filtering performed in both algorithms will eliminate all remaining points. Compared to an uniform distribution, in an exponential distribution there are more pairs of categories with small diversity and fewer with high diversity. This means that it may take longer to find solutions with higher diversity, which in turn reduces the pruning power of both solutions since subsets are discarded based on the highest diversity found so far.

5.3 Effectiveness and Efficiency

As mentioned in Sect. 2, *GNE* provides an approximated result whereas we obtain the optimal one. Nonetheless, for the sake of completeness we evaluated how sub-optimal were *GNE* results as well as how much faster they were obtained. We fixed λ, the trade-off between similarity and diversity given as input to the *GNE* algorithm, at 0.5. We assumed that the similarity of a point $p_i \in P$ w.r.t. q is given by $sim(p_i, q) = 1 - \frac{d(p_i,q)}{d_m}$, where d_m is the maximum distance used for normalization. Since *GNE* requires both criteria to be on the same scale, we also normalized the diversity value. Due to limited space we only present a summary of our findings.

Regarding spatial diversity, the quality of the results produced by *GNE* decreases when k increases. For instance, for $k = 3$, *GNE* is around 20% worse than our optimal approaches, while for $k = 7$ it is about 54% worse. The same behavior was observed when we varied the number of candidate points, where *GNE's* results were up to 40% worse (for 1000 candidate points). Lastly, *GNE* presented better results for a Gaussian distribution, being around 21% worse than our optimal approaches, followed by the uniform distribution, where it was about 33% worse. For a clustered distribution, *GNE*'s results were around 46% worse than the optimal ones our proposed approaches provide.

With relation to categorical diversity, the quality of the results produced by *GNE* increases with the number of categories. For instance, for 5 categories, *GNE* is around 54% worse than our optimal approaches, while for 100 categories it is only 7% worse. Regarding the diversity of the categories, *GNE* presented better results for the uniform distribution, being about 20% worse than our optimal approaches, but it was around 90% worse for an exponential distribution.

GNE's sub-optimality is compensated by its efficiency. It is able to find an approximate solution in less than 1 s in all cases we evaluated. However, it is important to stress that while it is faster, it finds an *approximate* solution for *one* single linear combination of the optimized criteria whereas our solutions find all *optimal* solutions for *any* such linear combination.

6 Conclusion

We have addressed the kDNN query, i.e., a variation of k-NN queries. This problem consists of finding elements that are as close as possible to the query point, while, at the same time, being as diverse as possible. We have considered two different notions of diversity, namely spatial and categorical. Previously proposed solutions are approximate and require the user to determine a specific weight for each type of diversity. Our proposed approaches based on linear skylines find *all* results that are *optimal* under *any* given arbitrary linear combination of the two competing diversity criteria.

We proposed two solutions to kDNN queries using the notion of skylines. *Recursive Range Filtering (RRF)* strives to reduce the number of combinations generated by recursively combining a partial set with candidate points that are diverse enough to be part of a non-dominated solution. *Pair Graph (PG)* works by pruning subsets that contain pairs of points that have low diversity and could not be together in a non-dominated solution. Experiments varying a number of parameters showed that (1) both *RRF* and *PG* are orders of magnitude faster than a straightforward solution, and (2) *PG* outperforms *RRF* for higher values of k and a greater number of points and *RRF* is more efficient otherwise.

As future work we would like to investigate kDNN queries within road networks, the use of real datasets, as well as to explore further the notion of diversity in other domains, e.g., text retrieval.

References

1. Abbar, S., et al.: Diverse near neighbor problem. In: SOCG, pp. 207–214 (2013)
2. Börzsönyi, S., et al.: The skyline operator. In: ICDE, pp. 421–430 (2001)
3. Carbonell, J., Goldstein, J.: The use of MMR, diversity-based reranking for reordering documents and producing summaries. In: SIGIR, pp. 335–336 (1998)
4. Carterette, B.: An analysis of NP-completeness in novelty and diversity ranking. Inf. Retrieval **14**, 89–106 (2011)
5. Clarke, C.L., et al.: Novelty and diversity in information retrieval evaluation. In: SIGIR, pp. 659–666 (2008)
6. Gollapudi, S., Sharma, A.: An axiomatic approach for result diversification. In: WWW, pp. 381–390 (2009)
7. Gu, Y., et al.: The moving k diversified nearest neighbor query. In: TKDE, pp. 2778–2792 (2016)
8. Handl, J., Knowles, J.: Cluster generators for large high-dimensional data sets with large numbers of clusters (2005). http://dbkgroup.org/handl/generators
9. Huang, Z., et al.: A clustering based approach for skyline diversity. Expert Syst. Appl. **38**(7), 7984–7993 (2011)
10. Jain, A., Sarda, P., Haritsa, J.R.: Providing diversity in K-nearest neighbor query results. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 404–413. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24775-3_49
11. Kucuktunc, O., Ferhatosmanoglu, H.: λ-diverse nearest neighbors browsing for multidimensional data. In: TKDE, pp. 481–493 (2013)
12. Lee, K.C.K., Lee, W.-C., Leong, H.V.: Nearest surrounder queries. In: ICDE, p. 85 (2006)
13. Rafiei, D., Bharat, K., Shukla, A.: Diversifying web search results. In: WWW, pp. 781–790 (2010)
14. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest neighbor queries. ACM SIGMOD Rec. **24**, 71–79 (1995)

15. Shekelyan, M., Jossé, G., Schubert, M., Kriegel, H.-P.: Linear path skyline computation in bicriteria networks. In: Bhowmick, S.S., Dyreson, C.E., Jensen, C.S., Lee, M.L., Muliantara, A., Thalheim, B. (eds.) DASFAA 2014. LNCS, vol. 8421, pp. 173–187. Springer, Cham (2014). doi:10.1007/978-3-319-05810-8_12
16. Tao, Y.: Diversity in skylines. IEEE Data Eng. Bull. **32**(4), 65–72 (2009)
17. Valkanas, G., Papadopoulos, A.N., Gunopulos, D.: Skydiver: a framework for skyline diversification. In: EDBT, pp. 406–417 (2013)
18. Vieira, M.R., et al.: On query result diversification. In: ICDE, pp. 1163–1174 (2011)
19. Yu, C., Lakshmanan, L., and Amer-Yahia, S.: It takes variety to make a world: diversification in recommender systems. In: EDBT 2009, pp. 368–378 (2009)
20. Zhang, C., et al.: Diversified spatial keyword search on road networks. In: EDBT, pp. 367–378 (2014)