IOSR Journal of Engineering (IOSRJEN)
e-ISSN: 2250-3021, p-ISSN: 2278-8719, www.iosrjen.org
Volume 2, Issue 11 (November2012), PP 08-17

Efficient Processing of Top-k Spatial Preference Queries

[1]K.Ankamma chowdary, [2]Y.Sowjanya Kumari, [3]Dr. P.Harini, Ph.D
[1]II year M.Tech(CSE) St.Ann'sCollege Of Engg &Technology Chirala
[2]Associate Professor (CSE) St.Ann'sCollege Of Engg &Technology Chirala
[3]HOD-CSE St.Ann'sCollege Of Engg &Technology Chirala

Abstract: *Top-k spatial preference queries arrival a ranked set of the k best data objects based on the scores of mark objects in their spatial neighborhood. Despite the wide assortment of location-based applications that rely on spatial predilection queries, existing algorithms incur non-negligible processing cost resulting in high retort time. The reason is that computing the score of an information object requires examining its spatial locality to find the feature object with the highest score. In this paper, we suggest a novel technique to speed up the performance of top-k spatial predilection queries. To this end, we propose a mapping of pairs of information and feature objects to a distance-score space, which in rotate allows us to identify and materialize the minimal subset of pairs that is adequate to answer any spatial preference query. Furthermore, we present a novel algorithm that improves uncertainty processing performance by avoiding examining the spatial neighborhood of the information objects during query execution. In addition, we recommend an efficient algorithm for materialization and we express useful properties that reduce the cost of maintenance. We show through wide experiments that our approach significantly reduces the number of I/Os and completing time compared to the state-of-the-art algorithms for dissimilar setups.*

Keywords *— spatial information, spatial location*

I. Introduction

With the popularization of geotagging in sequence, there has been an increasing number of Web information systems dedicated to providing interesting results through location-based queries. However, most of the accessible systems are limited to plain spatial queried that arrival the objects present in a given region of the space. In this paper, we cram a more sophisticated query that returns the best spatial objects based on the features (facilities) in their spatial neighborhood [16,17]. Given a set of information objects of interest, a top-k spatial preference query returns a ranked set of the k best information objects. The score of a data object is defined based on the non-spatial gain (quality) of feature objects in its spatial neighborhood. On the other hand, the score of an attribute object does not depend on its spatial location, but on the quality of the attributes object. Such quality values can be obtained by a rating provider (e.g. www.zagat.com).

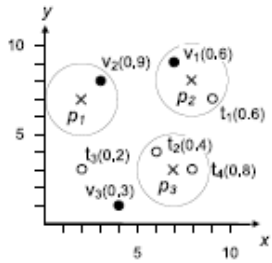


Figure 1: Spatial area containing data and feature objects.

For example, Figure 1 presents a spatial area containing information objects p (hotels) together with feature objects t (restaurants) and v (cafes) with their particular scores (e.g. rating). Consider a tourist concerned in hotels with good restaurants and cafes in their spatial neighborhood. The tourist specifies a spatial constraint (in the figure depicted as an assortment around each hotel) to restrict the distance of the eligible feature objects for each hotel. Thus, if the tourist wants to rank the hotels based on the gain of restaurants, the top-1 hotel is p3(0.8) whose score 0.8 is determined by t4. However, if the tourist wants to rank the hotels based on cafes, the top-1 hotel is pi (0.9) firm by v2. Finally, if the tourist is interested in equally restaurants and cafes (e.g. summing the scores), the top-1 hotel is P2(1.2).

Top-k spatial preference queries are intuitive and include a useful tool for novel location-based applications. Unfortunately, processing top-k spatial predilection queries is complex, because it may need to search the spatial neighborhood of all data objects previously to reporting the top-k. Due to this complexity, existing solutions are costly in terms of both I/Os and effecting time [16,17].

In this paper, we offer a novel approach for processing spatial preference queries efficiently. The main dissimilarity compared to traditional top-k queries is that the score of each information object is defined by the feature objects that assure a spatial constraint (for example range constraint). Therefore, pairs of information and feature objects need to be examined to decide the score of an object. Our approach relies on mapping of pairs of information and feature objects to a distance-score space, which in turn allows us to identify the negligible subset of pairs that is sufficient to answer all spatial predilection queries. Capitalizing on the materialization of this subset of pairs, we here an efficient algorithm that improves query processing concert by avoiding examining the spatial neighborhood of information objects during query execution. In addition, we suggest an efficient algorithm for materialization and describe useful properties that diminish the cost of maintaining the materialized information. In précis, the main contributions of this paper are:

We define a mapping of pairs of information and feature objects in the distance-score space that enables pruning of attribute objects that do not affect the score of any information object.

- We prove that there exists a negligible subset of pairs that is sufficient to answer all top-k spatial predilection queries.
- We suggest an efficient algorithm for processing top-fc spatial predilection queries that exploit the materialized separation of points.
- In addition, we suggest an effective algorithm for materialization, and we recognize useful properties for cost-efficient continuation of the materialized information.
- We show through a wider experimental evaluation that our algorithm outperforms the state-of-the-art algorithms in terms of equality I/Os and execution time.

The rest of this manuscript is organized as follows: In Section 2, we near an overview of the related work. In Section 3, we offer the necessary preliminaries and definitions. In Section 4, we illustrate the distance-score space and define the minimal set of relevant information and feature objects. Our algorithm for processing spatial predilection queries is presented in Section 5. In Section 6, we illustrate the process of materialization and discuss how continuance is performed. Finally, in Section 7, we present the experimental assessment and we conclude in Section 8.

II. Related Work

Several approaches have been projected for ranking spatial data objects. The reverse adjoining neighbor (RNN) query was first projected by Korn and Muthukrishnan [8]. Then, Xia et al. studied the difficulty of retrieving the top-fc most influential spatial objects [15], where the score of each spatial information object p is defined as the sum of the scores of all characteristic objects that have p as their adjacent neighbor. Yang et al. Studied the problem of finding an optimal location [4]. The main dissimilarity compared to [15] is that the optimal position can be any point in the data space and not necessarily a purpose of the dataset, while the score is computed in a similar way to [15].

The aforementioned approaches describe the score of a spatial data object p based on the scores of feature objects that have p as their adjacent neighbor and are limited to a single feature set. Differently, Yiu et al. First measured computing the score of a data object p based on feature objects in its spatial neighborhood from multiple feature sets [16,17]. To this end, three dissimilar spatial scores were defined: range, nearest neighbor, and influence score; and dissimilar algorithms were developed to compute top-fc spatial preference queries for these scores.

The algorithms developed by Yiu et al. Assume that the information objects are stored in an R-tree [6] based on spatial attributes, while the characteristic objects of each feature set are stored in a part aggregate R-tree (air-tree) [11]. The proposed algorithms can be divided into three categories. The first category is calm by probing algorithms, namely Simple (SP) and Group (GP) probing. These algorithms require to compute the score of all data objects before reporting the top-fc result set. The second category is composed by Branch and Bound (BB) and Branch and Bound Star (BB) algorithms. These algorithms avoid computing the achievements of some information objects. The ideas is computing an upper bound for each entry of the R-tree of the information objects, and prune the entries whose upper bound is smaller or equal to the score of the fc-th information object already found. The third category comprises the feature join (FJ) algorithm. FJ performs a multi-way spatial join on the feature sets to gain combinations of feature objects of high scores. Then, the

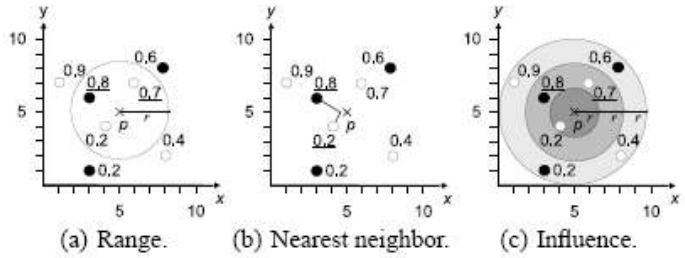


Figure 2: Examples of partial scores and spatial constraints.

The aim is to find information objects having the corresponding feature grouping with high score in their spatial neighborhood. A more exhaustive description of the algorithms can be found in Appendix A.
In this paper, we study the problem that was originally projected in [16]. Differently than [16,17], we suggest a materialization technique that leads to significant savings in together computational and I/O cost during query processing.

III. Preliminaries

Given an object dataset O and a set of c feature datasets {F; | i G [1, c]}, the top-k spatial predilection query [16,17] returns the k data objects {pi,... , pk } from O with the highest score. The score of an information object p G O is defined by the scores of feature objects t G F; in its spatial neighborhood. Each attribute object t is associated with a non-spatial score w(t) that indicates the goodness (quality) of t and its domain of values is the range [0,1].

The score t(p) of a data object p is determined by aggregating the partial scores tf (p) with deference to neighborhood condition 0 and the i-th feature dataset F;: t(p) = agg{Tf (p) | i G [1,c]}. The aggregate purpose agg can be any monotone function (such as sum, max, min), but we use sum in the following discussion for ease of appearance. The partial score Tf (p) of p is resolute by feature objects that belong to the i-th feature dataset Fi only, and in calculation satisfy the user-specified spatial limitation 0. More specifically, the partial score rf (p) is defined by the non-spatial score w(t*) of a single feature object t* G Fi. This feature purpose t* is the feature object with the highest score that satisfies the distressed condition 0. The following list provides intuitive definitions of partial score for dissimilar neighborhood conditions 0 (where d () denotes the distance function):
EXAMPLE 1. Figure 2 depicts an example of a set of spatial information objects. The points of feature datasets F1 and F2 are symbols with white and black dots respectively, while the information object p G O is represented with a cross mark. We assume that d () is the Euclidean detachment without loss of generality, i.e., Any other detachment function can be applied. In Figure 2 (a), for each F;, the range score of p is the maximum non-spatial score w (t) of the feature objects within distance r from p. Thus, T [U9 (p) = 0.7, T2,"9 (p) = 0.8, and the score of p is T (p) = ^2=1 t™9 (p) = 1.5. In Figure 2(b), for a given dataset F;, the adjacent neighbor score of p is the non-spatial score of the nearest characteristic object t G Fi to p. Thus, Tin (p) = 0.2, r2nn (p) = 0.8, and r (p) = 1.0. In Figure 2 (c), the manipulate score of p is the maximum influence score of all feature objects in Fi. The influence score is computed taking into explaining the non-spatial score oft that is reduced depending on the distance between t and p. The radius r controls how rapidly the score decreases with detachment and in our example we set r = 1.7.

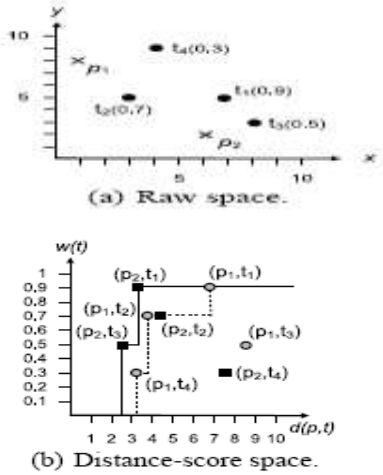


Figure 3: Mapping to the distance-score space M.

EXAMPLE 2. Figure 3 (a) depicts the spatial neighborhood of information objects p1 and p2, as well as the feature Fi, (black dots), while Figure 3 (b) depicts the mapping to the distance-score space M. In fastidious, the set M? 1 consists of pairs (p1, t1) ... (p1, t4) and is depicted with circles, whereas the set M? 2 consists of pairs (p2, t1) ... (p2, t4) and is depicted with black squares. Notice that smaller values are preferable for the reserve d(p, t), while higher values are preferable for the non-spatial score w(t). Therefore, the skyline sets of p1 andp2 with admiration to Fi, are S?1 ={(p1, t1), (p1 ,t2), (p1,t4)} and S?2 ={(p2, 11), (p2,t3)} respectively. Also notice that pairs that belong to dissimilar objects, i.e., p1, p2, are incomparable.

Algorithm 1 NextObject (Max Heap H)
1: INPUT: Max-heap H with entries in sliding order of non-spatial score and radius r.
2: OUTPUT: The next data object in H with maximum partial score.
3: Entry e ^ remove entry from top of the H
4: while e is not a data point do
5: for each entry e' that is child of e do
6: if d(e') < r then
7: insert e' into H
8: end if
9: end for
1000: e ^ remove entry from top of the H
11: end while
12: return e

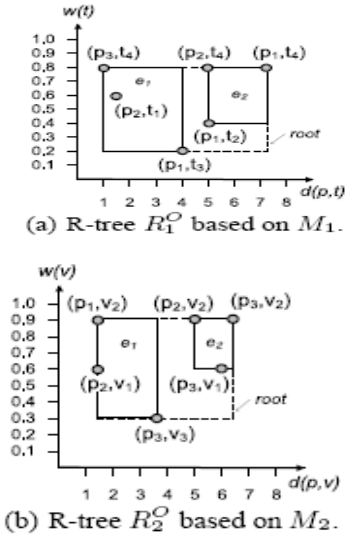


Figure 4: Example showing the contents of RiO.

Notice that the range constraint is posed only to the detachment independently of the score of the feature objects. The number of pairs *E* is an upper clear of the objects that are accessed during a spatial predilection range query, assuming uniform distribution of the data and the feature objects. In perform, our approach indexes only the set S?, Which is a subset of M:, and our algorithm takes into account the score of the characteristic objects to reduce the number of accessed pairs still further. The details on efficient materialization and maintenance of S? are presented in Section 6, while in the sequel we near the proposed top-*k* spatial preference uncertainty processing algorithm.

IV. Query Processing

In this section, we here the *Skyline Feature Algorithm (SFA)* for processing top-*k* spatial predilection queries. First, we present an algorithm that exploits the distance-score space and returns the information objects in descending order of their partial scores. Then, we here the algorithmic details of SFA, which produces the consequence of the top-*k* spatial preference query by coordinating access to the partial scores of information objects. For ease of presentation, in the following, we refer to a pair (p, t), where p G O and t G F:, as a *data point* indexed by R?.

Access to Partial Scores. During query processing, the information points in R? are retrieved sorted in descending order of their fractional scores. Furthermore, only node entries of the R-tree R? that satisfy the spatial limitation are processed. First, we present in details our algorithm for retrieving

information points sorted based on the range score (Algorithm 1). Then, we describe the necessary modifications for supporting the manipulate and nearest neighbor scores.

Next Object takes as input the radius *r* that defines the assortment constraint and a heap *H* that contains node entries and information points in descending order of partial score $T_{;}^{f}$ (). Initially, the heap H contains the root of R?. Each time, the ingress e at the top of the heap H, i.e., with maximum partial score, is retrieved (lines 3,10). As long as *e* is not a information point (line 4), Next Object inserts in the heap H (line 7) the children entries of e whose detachment is smaller or equal to the radius r (line 6).

NextObject can be adapted with minor modifications for the influence and nearest neighbor scores. For influence score, the radius is only used to calculate the score, therefore even feature objects whose detachment from a data object is larger than *r* may supply to the result set. Thus, line 6 of the algorithm has to be removed for manipulating score. Notice that *H* maintains the node entries in descending organize of partial score T^{f} (), which in this case is defined by the influence score. For adjacent neighbor score, NextObject has to be modified to prune pairs *(p, t)* such that *t* is not the nearest neighbor of p. For this principle, during the construction of *R*?, such data points are flagged to indicate if *t* is the adjacent neighbor of *p* in $F_{;}$ (by a bit 1-if nearest neighbor, 0-otherwise). Similarly, an entry *e* of $R_{i}^{?}$ is flagged, if at least one of its children entries is flagged. This enables proficient processing, as entries that do not contain an adjacent neighbor are immediately pruned. Then, lines 6-8 of Algorithm 1 are modified to first check if the child entry e' is a nearest neighbor entry, and only then e' is inserted in *H.* After these modifications, NextObject is readily employed for a range, adjacent neighbor and influence score.

The SFA Algorithm. SFA (Algorithm 2) computes the top-k spatial preference information objects progressively, by aggregating the incomplete scores of the information objects retrieved from each R-tree R? using NextObject algorithm. We use *sum* as the aggregate function in the following explanation and in the pseudocode.

Each time NextObject is invoked, the information object *p* with highest partial score T^{f} *(p)* is retrieved from R?, thus any unseen information object *p'* in R? has a smaller partial score than *p* (T^{f} *(p')* < t^{f} (p)). Therefore, we can calculate an upper bound on the score *t (p)* of any information object *p* based on the highest partial scores T^{f} *(p)* of seen information objects in each $R_{i}^{?}$.

SFA employs an upper bound $U_{;}$ on the score of some unseen object in each heap $H_{;}$. Also, for each $H_{;}$, a list $L_{;}$ of seen objects is maintained. Moreover, each time an object *p* is retrieved from $H_{;}$ for the first time, p's lower bound to score (p^{-}) can be updated using the aggregate purpose (in this case *sum*). In addition, SFA maintains a list *C* of candidate information objects that may eventually become top-k results. *C* is sorted based on descending lower bound to score.

In each iteration (line 6), SFA selects one heap *H*i (line 7) to retrieve the next information object *p* (line 8). The upper bound *U*i on the score of $H_{;}$ is set (line 9) based on p's partial score T^{f} *(p).* Then, if *p* has not been seen before in $H_{;}$, its lower bound p^{-} is updated based on the partial score T^{f} *(p)* and *p* is added to $L_{;}$ (lines 10-13). Notice that although *p* may be retrieved over from $H_{;}$,

Algorithm 2 *SFA(MaxHeap H*1,..., *Hc)*
1: INPUT: Heaps H; containing the root of R?.
2: OUTPUT: Top-k spatial preference objects.
3: C — *0* // *List of seen objects p sorted by lower bound on score* p^{-}
4: *L;* — 0 *//List of seen objects p from heap H;*
5: *U;* — <^ // *Upper bound on score for each heap H;*
6: while *3H;* such that H; = 0 do
7: i — index of the next input
8: *(p,t) — NextObject(H;) //Next unseen object of H;*
9: U; — r^{f} (p)
10: ifp ^ L; then
11:p^{-} — p^{-} + T^{f} (p)
12: L; —— L; U p
13:end if
14:if p ^ C then 15: C — C U p
16:end if
17:*q — C.peek() // Object with the best lower bound*
18:*max — max*$_{Vpe}$*o,p=q(*p^{-} + E *Vj-p^Lj* U;) *//Upper bound*
19: while q^{-} > *max* do
20:q — C.pop()
21:report *q* as next top-k, halt if *k* objects have been reported
22:q — *C.peek()* // *Object with the best lower bound*

23:max — *maxvpeo,p=q(p*$^{-}$ + $E_{Vj;p}/_{Lj}$ U;)
24: end while 25: end while
26: while fewer than *k* objects have been reported do 27: q — *C.pop()*
28: report *q* as next top-k
29: end while

the maximum T^{f} (p) is encountered at the first time, because $H_{;}$ is admission in descending order of score. In addition, *p* is added to the list *C* of candidate objects (lines 14-16). Then, the upper bound (denoted as *max)* on the score of any purpose is computed in line 18.
We can safely report as next top-k consequence, any object *q* in the top of the list *C* whose lower bound q^{-} is greater than or equal to *max* (lines 19-24). SFA continues in the same fashion, until k objects have been reported, or until all heaps are exhausted. In the latter case, if fewer than *k* objects have been reported, the objects in *C* are returned based on the sorting of *C* (because the lower bound now equals to the real score), until we have *k* objects (lines 26-29).

The problem of combining incomplete scores for top-*k* spatial preference queries is comparable to the problem of aggregating ranked inputs [5,9]. For ease of appearance, we omitted from Algorithm 2 implementation details that consequence in reducing the number of data objects in the list *C* and, therefore, also the essential number of comparisons (see [9]).

V. Materializationand Maintenance

SFA processes top-*k* spatial preference queries efficiently, when each set of points S? is stored in an R-tree *R*?. The remaining
challenge is to computefficiently and materialize the set $S_{;}^{?}$ in a preprocessing phase and to maintain $S_{;}^{?}$ when updates occur. The proofs of the theorems and the lemmas of this section can be found in Appendix B.

Materialization: The straightforward loom for computing the set $S_{;}^{?}$ is to combine each information object *p* G *O* with each feature object *t G* $F_{;}$ to construct pairs *(p,t),* and then execute a skyline algorithm to compute the set S?. This approach is correspondent to first computing the entire set $M_{;}^{p}$ and then computing its skyline, which is prohibitively exclusive for large datasets. An alternative approach is for each information object *p* G *O* and *F*; to execute a dynamic skyline query [12] on the dynamic coordinates *d(p, t)* and *w(t),* in regulate to compute Sp. For each data object, some feature objects can be pruned, but one dynamic skyline query is still obligatory for each information object. Hence, this approach also has a high I/O cost, especially when the cardinality |O| of the object dataset is high.

Nevertheless, data objects that are close in space, i.e., their distance is small, have comparable distances to any feature object. Therefore, the skyline sets of such objects are also comparable with high probability. In order to condense the number of required dynamic skyline queries (and, in consequence, the I/O cost induced by accessing $F_{;}$), the information objects are partitioned into groups, so that the distances of information objects that are in the same group are small. Then, for each group of information points, a *dynamic region skyline query* is posed (that will be defined in the following) and we will show that the result set is a superset of all skyline sets of information points that belong to the group.

Let us assume a grouping of points and let the region A be the minimum bounding rectangle that encloses all information points of the group. We denote as *maxDist(A, t)* and *minDist(A, t)* the maximum and the minim distance among *t* and any data object enclosed in *A* respectively. For the case that *t* is enclosed in *A,* the minimum reserve is zero.

Maintenance: In the following, we discuss the issue of index preservation in the presence of insertions, deletions and updates of information or feature objects. Insertions and deletions of a data object p G O are relatively straight forward and cost-efficient. When p is inserted in O, each index should be updated by inserting the skyline points S? of the mapping M? of p based on F_i. If p is deleted, any occurrence of p in an index must also be deleted. Updates of the spatial location of p are handled as a removal followed by an insertion.

The most frequent maintenance process is update of the score of a feature object. Usually, the score of feature objects (e.g. user ratings) adjust dynamically, in comparison to the spatial location of a feature object which is additional static. In practice, such updates of score are expected to occur more often than updates of the geographic location. The confront of handling updates of the score of a feature object t G Fi is that such an update can potentially influence all materialized skyline sets S^{O}. However, we show that we can utilize useful properties of the mapping to distance-score space to drastically decrease the cost of updates.

For ease of presentation, we first suppose that all feature objects t G F_i have distinct score values. We will drop this restraint later. We define a total ordering T of the feature objects t G F_i based on their scores w(t), such that t head t' if w(t) > w(t'). The following lemmas decide when an update of a feature object's score causes an update to the materialized skyline sets and, hence, to the indicator.

VI. Experimental Study

In this section, we evaluate our proposed algorithm *(SFA)* and we compare SFA against the algorithms developed by Yiu *et al.* [16, 17], denoted as *GP*, *BB*, *BB**, and *FJ*. All algorithms were implemented in Java and executed on a PC with 3GHz Dual Core AMD Processor with 2GB RAM. The datasets were indexed by an R-tree (aR-tree for [16,17]) with block size of 4KB. We used an LRU memory buffer with a fixed size of 0.2% of the size of the total number of objects stored in O and F_i. We report the average values of 20 experiments, and in each experiment we recreate all datasets and indexes to factor out the effects of randomization. In all experiments, we measured the total execution time (referred to as response time) and number of I/Os. All charts are plotted using a logarithmic scale on the y-axis.

6.1 Experimental Settings

We conduct experiments using both synthetic and real datasets. First, we perform experiments using uniform distribution (UN) for the spatial locations of data and feature objects and for the score of the feature objects (within the range [0,1]). We also generate a synthetic dataset (CN) that resembles the real world: (1) there exist multiple city centers (centroids) with higher occurrences of data objects, (2) there exists a higher probability of finding feature objects nearby the city centers (centroids). Appendix C.1 provides a detailed description of CN including a plot of a generated dataset. We use the synthetic dataset (CN) as our default dataset. By default, the non-spatial score of the feature objects is a uniformly generated value within the range [0,1]. In addition, we evaluate also score values that follow the exponential distribution (Appendix C.3). In Appendix C, we provide a table that contains the parameters and values used in the experimental evaluation, the description and more experimental results of the real dataset, and we evaluate the cost of materialization.

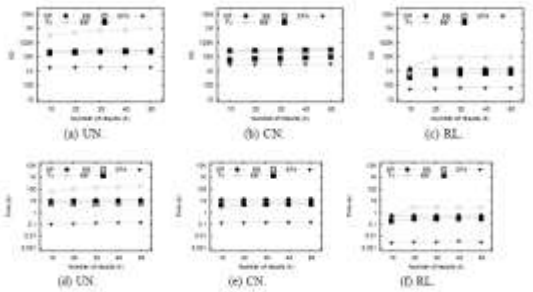


Figure 5: Effect of different data distributions {UN,CN,RL} on I/O and response time (range score).

7.2 Query Processing Performance

Range Score. In Figure 5, we use our default setup and study the number of I/Os and the response time for all datasets, while varying k. Figure 5(a) presents the I/O cost using the UN dataset. The performance of GP is stable because it always computes the score of all data objects. FJ requires a much higher number of I/Os, as it needs to access many leaf entries of the feature R-trees in order to report the correct top-k result set. The branch-and-bound algorithms (BB and BB*) perform slightly better than GP for this setup. However, SFA results in one order of magnitude fewer I/Os than the best of its competitors. In Figure 5(b), we plot the number of I/Os for the CN dataset. BB* performs better than GP, BB, and FJ due to the employed pruning. However, SFA reduces even further the number of required I/Os compared to BB* and scales better than BB* for increasing value of k. In Figure 5(c), the I/O cost for the real dataset (RL) is presented. Again, SFA outperforms all other algorithms (in terms of I/Os) by at least one order of magnitude. This experiment indicates that SFA performs efficiently for a wide range of different datasets. Figures 5(d), 5(e) and 5(f) depict the response time for the same experimental setups respectively. In general, we observe that the gain of SFA compared to the other algorithms in terms of response time is even higher than the gain in I/Os (between one and two orders of magnitude). The fast response time of SFA indicates that SFA is suitable for applications involving Web information systems, where the main challenge is to minimize the response time for the user.

In the next experiment, we vary the number of features c and evaluate the performance of our algorithm. SFA outperforms all other algorithms both in terms of I/Os (Figure 6(a)) and response time (Figure 6(b)). For a single feature dataset, SFA requires only few I/Os in order to retrieve the top-10 objects. Also, notice that SFA results in very small response time (under one second) even in the case of c=5 feature datasets. On the other hand, FJ does not scale with increasing values of c and has the worst performance of all algorithms for higher values of c. In the following, due to space limitations, we report only the response time, however we observed that the relative trends in I/Os are similar.

In Figure 7, we vary different parameters and evaluate the response time of queries with range score. Figure 7(a) depicts the response time with varying radius r. SFA is always faster than all

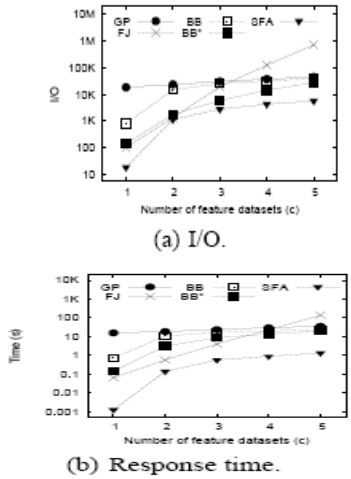


Figure 6: Effect of c on I/O and response time (range score).

other algorithms, irrespective of the value of radius. Notice that FJ and BB* perform worse for small radius and improve their performance with increasing radius until a certain point, because for very small radius many objects (or combinations of feature objects) have to be examined in order to identify an object with non-zero score, which can then be used for pruning. This is because most objects have zero score as there exist no feature objects in their neighborhood. Next, we study the scalability of SFA by varying the cardinality of the feature datasets |F;| (Figure 7(b)) and the cardinality of the object dataset |O| (Figure 7(c)). In Figure 7(b), we notice that increasing |F;| affects the performance of all algorithms, but not SFA. The main reason is that increasing the size of |F;| has a small impact on the cardinality of skyline sets S?. Since SFA materializes pairs that are not dominated, the number of such pairs is not affected significantly by increasing |Fi|. In Figure 7(c), SFA outperforms all algorithms, even though FJ is more stable with increasing |O| than SFA. This is mainly because FJ is sensitive to the cardinality of Fi and not to the size of O.

Influence Score. In the following, we evaluate the performance of SFA for processing queries with influence score (Figure 8). We compare our approach against BB and BB*, which support queries with influence score. In Figure 8(a), we vary the number of features c. Notice that computing queries with influence score is very costly for BB and BB*. The main reason is that the influence score limits the pruning capabilities of BB and BB*, therefore they have to search a large area of the space for computing the score of the data objects. SFA, on the contrary, accesses the data objects in decreasing order of influence score, without any significant additional cost compared to the range score. Thus, SFA is more than two orders of

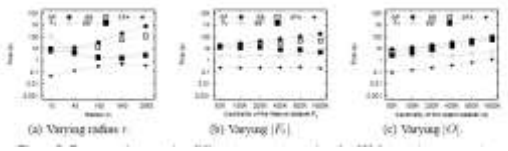


Figure 7: Response time varying different parameters using the CN dataset (range score).

magnitude faster than its competitors. In Figure 8(b), we evaluate the effect of varying k. All algorithms show stable performance for varying k and SFA always performs best.

Nearest Neighbor Score. In Figure 9, we compare our approach against GP and BB algorithms for processing nearest neighbor queries. There is no implementation of BB* for nearest neighbor queries, and it is not trivial to adapt BB* for these queries. In Figure 9(a), we evaluate the effect of increasing the number of features c. Similar to other experiments where we evaluated the impact of varying c, SFA performs more efficiently. In Figure 9(b), we evaluate the effect of increasing values of k. Again, SFA is two orders of magnitude better than BB and GP regardless of the exact value of k.

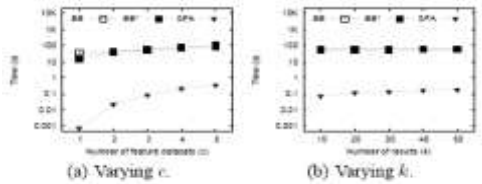
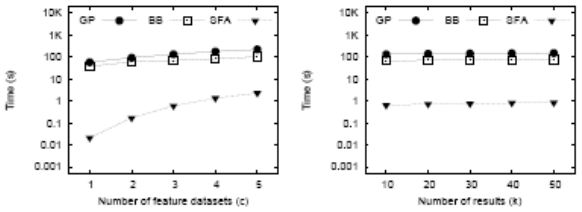


Figure 8: Effect of *c* and *k* on response time (influence score).



(a) Varying c. (b) Varying k.

Figure 9: Effect of c and k on response time (NN score).

VII. Conclusions

In this paper, we here a novel approach for boosting the presentation of top-k spatial preference query processing. At the heart of our framework lies a mapping of pairs of information and feature objects to a distance-score space, which enables us to recognize the minimal subset of pairs necessary to answer any ranked spatial predilection query. By materializing this subset of pairs, we here efficient algorithms for query processing that result in better performance. Furthermore, we describe an capable algorithm for materialization and elaborate on useful properties that condense the cost of maintenance. Our experimental assessment demonstrates that our approach reduces I/Os and response time by more than one order of extent compared to the state-of-the-art algorithms in most of the setups.

References

[1] C. Bohm, B. C. Ooi, C. Plant, and Y. Yan. Efficiently processing continuous k-nn queries on data streams. In Proc. ofInt. Conf. on Data Engineering (ICDE), pages 156-165, 2007.

[2] S. Borzsonyi, D. Kossmann, and K. Stocker. The skyline operator. In Proc. of Int. Conf on Data Engineering (ICDE), page 421430, 2001.

[3] S. Chaudhuri, N. N. Dalvi, and R. Kaushik. Robust cardinality and cost estimation for skyline operator. In Proc. ofInt. Conf. on Data Engineering (ICDE), page 64, 2006.

[4] Y. Du, D. Zhang, and T. Xia. The Optimal-Location query. In Proc. of the Int. Symposium on Spatial and Temporal Databases (SSTD), pages 163-180, 2005.

[5] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. Journal ofComputer and System Sciences, 66(4):614-656, 2003.

[6] A. Guttman. R-trees: a dynamic index structure for spatial searching. In Proc. of the Int. Conf on Management of Data (SIGMOD), pages 47-57, Boston, Massachusetts, 1984.

[7] J. Han, M. Kamber, and A. K. H. Tung. Spatial clustering methods in data mining: A survey. Geographic Data Mining and Knowledge Discovery, pages 1-29, 2001.

[8] F. Korn and S. Muthukrishnan. Influence sets based on reverse nearest neighbor queries. In Proc. of the Int. Conf on Management of Data (SIGMOD), pages 201-212, 2000.

[9] N. Mamoulis, M. L. Yiu, K. H. Cheng, and D. W. Cheung. Efficient top-k aggregation of ranked inputs. ACM Transactions on Database Systems (TODS), 32(3):19, 2007.

[10] K. Mouratidis, S. Bakiras, and D. Papadias. Continuous monitoring of top-k queries over sliding windows. In Proc. of the Int. Conf. on Management of Data (SIGMOD), pages 635-646, 2006.

[11] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. Efficient OLAP operations in spatial data warehouses. In Proc. ofthe Int. Symposium on Advances in Spatial and Temporal Databases (SSTD), pages 443-459. Springer-Verlag, 2001.

[12] D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. ACM Transactions on Database Systems (TODS), 30(1):41-82, 2005.

[13] E. Pekalska and R. P. W. Duin. Classifiers for dissimilarity-based pattern recognition. In Proc. ofInt. Conf. on Pattern Recognition (ICPR), pages 2012-2016, 2000.

[14] H. Samet. The quadtree and related hierarchical data structures. ACM Comput. Surv., 16(2):187-260, 1984.

[15] T. Xia, D. Zhang, E. Kanoulas, and Y. Du. On computing top-t most influential spatial sites. In Proc. of the Int. Conf. on Very Large Data Bases (VLDB), pages 946-957, 2005.

[16] M. L. Yiu, X. Dai, N. Mamoulis, and M. Vaitis. Top-k spatial preference queries. In Proc. ofInt. Conf. on Data Engineering (ICDE), pages 1076-1085, 2007.

[17] M. L. Yiu, H. Lu, N. Mamoulis, and M. Vaitis. Ranking spatial data by quality preferences. Transactions on Knowledge and Data Engineering (TKDE), to appear.

[18] Z. Zhang, Y. Yang, R. Cai, D. Papadias, and A. K. H. Tung. Kernel-based skyline cardinality estimation. In Proc. ofthe Int. Conf. on Management of Data (SIGMOD), pages 509-522, 2009.

ABOUT THE AUTHORS

Mr. k.Ankamma chowdary
has completed B.Tech. from RVR&JC College Engg & tech, and currently studying M.Tech in CSE at St.Ann's col of Engg & Tech.

Mrs. Y.Sowjanya kumari:
Currently she working as an Associate Professor in St.Anns's college Engg & Tech. College, Chirala

Dr.P.Harini received B.E. degree in Electronic and Communications Engineering from University of Madras, Chennai, in 1993, received M.Tech. degree in Remote Sensing from JNTU, Hyderabad, in 1997, received M.Tech. degree in Computer Science and Engineering from JNTU, Hyderabad, in 2003 and received Ph.D. in Computer Science and Engineering from JNTU, Anantapur, in 2011. She has 16 Years of Experience in which 1 year of Industrial, 1 year of Research & over 14 years of rich Teaching Experience in reputed Engineering Colleges & She is currently working as Professor & HOD in Computer Science & Engineering department in St.Ann's College of Engineering & Conferences. Guided many UG & PG students for projects & Life time Member of ISTE & CSI. Conducted successfully many Workshops, Seminars, conferences, FDPs and many National Level Technical Symposiums.