

Spatial Neighbourhood Features for Ranking Objects

¹K. Maneendra Varma, ²D. Srinivas

^{1,2}Dept. of CSE, Kakinada Institute of Engineering and Technology, Korangi, Kakinada, AP, India

Abstract

In this paper, we formally define spatial preference queries and propose appropriate indexing techniques and search algorithms for them. Extensive evaluation of our methods on both real and synthetic data reveals that an optimized branch-and-bound solution is efficient and robust with respect to different parameters. A spatial preference query ranks objects based on the qualities of features in their spatial neighbourhood. For example, using a real estate agency database of flats for lease, a customer may want to rank the flats with respect to the appropriateness of their location, defined after aggregating the qualities of other features (e.g., restaurants, cafes, hospital, market, etc.) within their spatial neighbourhood. Such a neighbourhood concept can be specified by the user via different functions. It can be an explicit circular region within a given distance from the flat. Another intuitive definition is to assign higher weights to the features based on their proximity to the flat.

Keywords

Query Processing, Spatial Databases

I. Introduction

There is a growing interest in spatial data mining in many different areas such as transportation, ecology, epidemiology, etc. Many algorithms to extract knowledge from such data have been proposed in the literature. One of the main problems is that only a few of these algorithms are implemented in toolkits which support the whole discovery process. In spatial/geographic databases data are stored in different relations (e.g. cities, rivers, roads), which need to be spatially joined in order to find novel and useful patterns in data mining. This step is normally performed in data preprocessing, and basically manually, since there is no available free and open source toolkit which automates this step. The result is that effort and time is required from the data mining user, which is usually neither an expert in geographic databases nor in data mining. It is stated that between 60 and 80 percent of the time and effort in the whole discovery process is required for data preparation [Addrians and Zantinge 1996]. Another problem is that data mining algorithms produce large amounts of outputs, mainly the association rule mining technique. In spatial data mining this problem increases significantly. Besides the large number of patterns, many are well known a priori, because of the natural geographic dependences intrinsic to geographic data [Bogorny et al 2006a, 2006b]. Many solutions for spatial data mining have been proposed in the literature, but only a few consider data preparation aspects. Most approaches extend query languages with new functions and operations for data mining. Han et al (1997) proposed a groining query language (GMQL) implemented in the Geo Miner software prototype. algebra et al (2000) proposed an object-oriented data mining query language named SDMOQL, implemented in the INGENS software prototype. In these approaches it is expected that the GDBMS will implement the proposed languages and operations. However, most GDBMS follow the Structured Query Language (SQL), which became the standard language to manipulate databases, and do not implement data mining functionalities to either automate or semi-automate geographic data preprocessing. Indeed, most geographic data mining software

prototypes are no longer available outside academic institutions. Aiming to make a contribution for the spatial data mining field and to facilitate the practice of knowledge discovery in geographic databases, we developed We ka- GDPM+, which is an extension of We ka [Witten and Frank 2005] [We ka 2007] to support spatial data mining from geographic data stored under Post GIS [Post GIS 2007]. We ka is a free and open source classical data mining toolkit which provides friendly graphical user interfaces to perform the whole discovery process. It implements a variety of data mining algorithms and has been widely used for mining non-spatial databases. A version of the extended We ka is available for download at [Bogorny 2007]. The main objective of this work is to automate geographic data preprocessing steps into the Weka data mining toolkit. Additionally, in this extension an interface is provided such that the user may define well known dependences which are used in data preprocessing to reduce the number of spatial joins and non-interesting patterns.

The main contributions include:

- Interoperability with any GDBMS constructed under Open GIS Simple Features Implementation Specification [OGC 1999a];
- Generate new datasets with different spatial relationships and attributes automatically;
- Allow the use of different classical data mining algorithms in the data mining step;
- Allow the user to choose the relevant spatial features, the reference feature type, different spatial relations, as well as different granularity levels [Han 1995];
- Geographic dependences can be specified by the user. These dependences are eliminated into We between the target feature type and any relevant feature type.

This results in more efficient spatial joins and the generation of less well known geographic domain patterns, as explained in [Bogorny et al 2006a]. haps X is also a customer of some other company, and that company provided the data to the target. Or perhaps X can be reconstructed from various publicly available sources on the web. Our goal is to estimate the likelihood that the leaked data came from the agents as opposed to other sources. We say an agent U_i is guilty and if it contributes one or more objects to the target. We denote the event that agent U_i is guilty by G_i and the event that agent U_i is guilty for a given leaked set S by $G_{ij}S$. Our next step is to estimate $PrfG_{ij}Sg$, i.e., the probability that agent U_i is guilty given evidence S .

II. Existing System

Object ranking is a popular retrieval task in various applications. In relational databases, we rank tuples using an aggregate score function on their attribute values [2]. For example, a real estate agency maintains a database that contains information of flats available for rent. A potential customer wishes to view the top-10 flats with the largest sizes and lowest prices. In this case, the score of each flat is expressed by the sum of two qualities: size and price, after normalization to the domain $[0,1]$ (e.g., 1 means the largest size and the lowest price). In spatial databases, ranking is often associated to nearest neighbour (NN) retrieval. Given a query location, we are interested in retrieving the set of nearest objects to it that qualify a condition (e.g., restaurants). Assuming that the

set of interesting objects is indexed by an R-tree [3], we can apply distance bounds and traverse the index in a branch-and-bound fashion to obtain the answer [4]. Nevertheless, it is not always possible to use multidimensional indexes for top-k retrieval. First, such indexes break-down in high dimensional spaces [5], [6]. Second, top-k queries may involve an arbitrary set of user-specified attributes (e.g., size and price) from possible ones (e.g., size, price, distance to the beach, number of bedrooms, floor, etc.) and indexes may not be available for all possible attribute combinations (i.e., they are too expensive to create and maintain). Third, information for different rankings to be combined (i.e., for different attributes) could appear in different databases (in a distributed database scenario) and unified indexes may not exist for them. Solutions for top-k queries, [2, 7-9] focus on the efficient merging of object rankings that may arrive from different (distributed) sources.

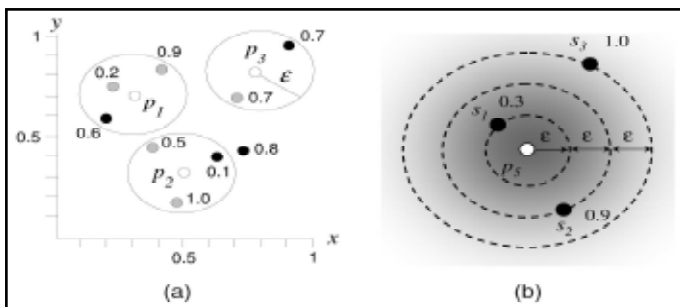


Fig. 1. Examples of Top-k Spatial Preference Queries. (a) Range Score, $\frac{1}{4}$ 0:2 km. (b) Influence Score, $\frac{1}{4}$ 0:2 km.

Their motivation is to minimize the number of accesses to the input rankings until the objects with the top-k aggregate scores have been identified. To achieve this, upper and lower bounds for the objects seen so far are maintained while scanning the sorted lists. In the following subsections, we first review the R tree, which is the most popular spatial access method and the NN search algorithm of [4]. The most popular spatial access method is the R-tree [3], which indexes minimum bounding rectangles (MBRs) of objects. Figure 2 shows a set $D = \{p_1, \dots, p_8\}$ of spatial objects (e.g., points) and an R-tree that indexes them. R-trees can efficiently process main spatial query types, including spatial range queries, nearest neighbour queries, and spatial joins. Given a spatial region W , a spatial range query retrieves from D the objects that intersect W . For instance, consider a range query that asks for all objects within the shaded area in Figure 2. Starting from the root of the tree, the query is processed by recursively following entries, having MBRs that intersect the query region. For instance, e_1 does not intersect the query region, thus the sub tree pointed by e_1 cannot contain any query result. In contrast, e_2 is followed by the algorithm and the points in the corresponding node are examined recursively to find the query result p_7 . A nearest neighbour (NN) query takes as input a query object q and returns the closest object in D to q . For instance, the nearest neighbour of q in Figure 2 is p_7 . Its generalization is the k-NN query, which returns the k closest objects to q , given a positive integer k . NN (and k-NN) queries can be efficiently processed using the best first (BF) algorithm of [4], provided that D is indexed by an R-tree. A min-heap H which organizes R-tree entries based on the (minimum) distance of their MBRs to q is initialized with the root entries. In order to find the NN of q in Figure 2, BF first inserts to H entries e_1 , e_2 , e_3 , and their distances to q . Then the nearest entry e_2 is retrieved from H and objects p_1 ; p_7 ; p_8 are inserted to H . The next nearest entry in H is p_7 , which is the nearest neighbour of q . In terms of I/O, the BF algorithm is shown to be no worse than any NN

algorithm on the same R-tree [4]. The aggregate R-tree (a R-tree) [10] is a variant of the R tree where each non-leaf entry augments an aggregate measure for some attribute value (measure e) of all points in its sub tree. As an example, the tree shown in fig. 2 can be upgraded to a MAX aR-tree over the point set, if entries e_1 ; e_2 ; e_3 contain the maximum measure values of sets $\{p_2, p_3\}$; $\{p_1, p_8\}$; $\{p_7, p_4, p_5, p_6\}$, respectively. Assume that the measure values of p_4, p_5, p_6 are 0.2; 0.1; 0.4, respectively. In this case, the aggregate measure augmented in e_3 would be $\max\{0.2; 0.1; 0.4\} = 0.4$. In this paper, we employ MAX aR-trees for indexing the feature datasets (e.g., restaurants), in order to accelerate the processing of top-k spatial preference queries. Given a feature dataset F and a multi-dimensional region R , the range top-k query selects the tuples (from F) within the region R and returns only those with the k highest qualities. Hong et al. [11] indexed the dataset by a MAX aR-tree and developed an efficient tree traversal algorithm to answer the query. Instead of finding the best k qualities from F in a specified region, our (range score) query considers multiple spatial regions based on the points from the object dataset D , and attempts to find out the best k regions (based on scores derived from multiple feature datasets F_c).

II. Proposed Work

In this section we discuss the IR-Tree. The IR-tree [2], which we use as a baseline, is essentially an R-tree [5] extended with inverted files [32]. The IR-tree's leaf nodes contain entries of the form $(p, p.\lambda, p.di)$, where p refers to an object in dataset D , $p.\lambda$ is the bounding rectangle of p , and $p.di$ is the identifier of the description of p . Each leaf node also contains a pointer to an inverted file with the text descriptions of the objects stored in the node. An inverted file index has two main components. • A vocabulary of all distinct words appearing in the description of an object. • A posting list for each word t that is a sequence of identifiers of the objects whose descriptions contain t . Each non-leaf node R in the IR-tree contains a number of entries of the form $(cp, rect, cp.di)$ where cp is the address of a child node of R , $rect$ is the MBR of all rectangles in entries of the child node, and $cp.di$ is the identifier of a pseudo text description that is the union of all text descriptions in the entries of the child node. As an example, Figure 2a contains 8 spatial objects $p_1, p_2, \dots, p_8$, and Figure 2b shows the words appearing in the description of each object. Figure 3a illustrates the corresponding IR-tree, and Figure 3b shows the contents of the inverted files associated with the nodes. The ITERATE algorithm (Algorithm 1) for computing the joint top-k spatial keyword query is adapted from an existing algorithm [2] that considers a single query. Recall that a joint top-k spatial keyword query Q consists of a set of sub queries q_i . The ITERATE algorithm computes the top-k results for each sub query separately. The arguments are a joint query Q , the root of an index root, and the number of results k for each sub query.

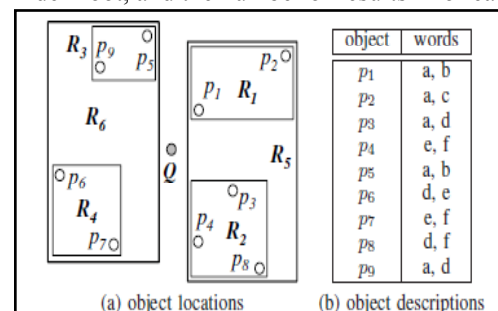


Fig. 2:

When processing a sub query $q_i \in Q$, the algorithm maintains a

priority queue U on the nodes to be visited. The key of an element $e \in U$ is the minimum distance $\text{mindist}(q_i, \lambda, e, \lambda)$ between the query q_i and the element e . The algorithm utilizes the keyword information to prune the search space. It only loads the posting lists of the words in q_i . A non-leaf entry is pruned if it does not match all the keywords of q_i . The algorithm returns k elements that have the smallest Euclidean distance to the query and contain the query keywords.

III. Algorithms

We propose a group evaluation technique that computes the scores of multiple points concurrently

A. Simple Probing Algorithm

Algorithm 1. Simple Probing Algorithm

```

algorithm SP(Node  $N$ )
1: for each entry  $e \in N$  do
2:   If  $N$  is nonleaf then
3:     read the child node  $N'$  pointed by  $e$ ;
4:     SP( $N'$ );
5:   else
6:     for  $c := 1$  to  $m$  do
7:       If  $\tau_+(e) > \gamma$  then  $\triangleright$  upper bound score
8:         compute  $\tau_c(e)$  using tree  $\mathcal{F}_c$ ; update  $\tau_+(e)$ ;
9:       If  $\tau(e) > \gamma$  then
10:        update  $W_k$  (and  $\gamma$ ) by  $e$ ;

```

B. Group Probing Algorithm

In GP, when a leaf node is visited, its points are first stored in a set V and then their component scores are computed concurrently at a single traversal of the $\mathcal{F}_c$ tree. We now introduce some distance notations for MBRs. Given a point p and an MBR e , the value $\text{mindist}(p, e)$ (max dist p, e) [4] denotes the minimum (maximum) possible distance between p and any point in e . Similarly, given two MBRs e_a and e_b , the value $\text{mindist}(e_a, e_b)$

b , R_1 is inserted into the queue, while R_2 is pruned. Now R_6 and R_1 are in the queue, and R_6 is dequeued. After loading the posting lists of words a and b in $\text{InvFile-}R_6$, R_3 is inserted into the queue, while R_4 is pruned. Now R_1 and R_3 are in the queue, and R_1 is dequeued.

Algorithm 2. Group Range Score Algorithm

```

algorithm Group_Range(Node  $N$ , Set  $V$ , Value  $c$ ,
Value  $\epsilon$ )
1: for each entry  $e \in N$  do
2:   If  $N$  is nonleaf then
3:     If  $\exists p \in V, \text{mindist}(p, e) \leq \epsilon$  then
4:       read the child node  $N'$  pointed by  $e$ ;
5:       Group_Range( $N', V, c, \epsilon$ );
6:   else
7:     for each  $p \in V$  such that  $\text{dist}(p, e) \leq \epsilon$  do
8:        $\tau_c(p) := \max\{\tau_c(p), \omega(e)\}$ ;

```

C. Branch-and-Bound Algorithm

GP is still expensive as it examines all objects in D and computes their component scores. We now propose an algorithm that can significantly reduce the number of objects to be examined. The key idea is to compute, for non leaf entries e in the object tree D , an upper bound T of the score for any point p in the sub tree of e . If then we need not access the sub tree of e , thus we can save numerous score computations.

Algorithm 3 is a pseudo code of our BB algorithm, based on this idea. BB is called with N being the root node of D . If N is a non leaf node, Lines 3-5 compute the scores Top for non leaf entries e concurrently. Recall that Top is an upper bound score for any point in the sub tree of e . The techniques for computing Top will be discussed shortly. Like (3), with the component scores Top known so far, we can derive Top an upper bound of Top then the sub tree of e cannot contain better results than those in W_k and it is removed from V . In order to obtain points with high scores early, we sort the entries in descending order of Top invoking the above procedure recursively on the child nodes pointed by the entries in V .

If N is a leaf node, we compute the scores for all points of N concurrently and then update the set W_k of the top- k results. Since both W_k and γ are global variables, their values are updated during recursive call of BB.

Algorithm 3. Branch-and-Bound Algorithm

```

 $W_k :=$  new min-heap of size  $k$  (initially empty);
 $\gamma := 0$ ;  $\triangleright$   $k$ th score in  $W_k$ 

algorithm BB(Node  $N$ )
1:  $V := \{e | e \in N\}$ ;
2: If  $N$  is nonleaf then
3:   for  $c := 1$  to  $m$  do
4:     compute  $\mathcal{T}_c(e)$  for all  $e \in V$  concurrently;
5:     remove entries  $e \in V$  such that  $\mathcal{T}_+(e) \leq \gamma$ ;
6:   sort entries  $e \in V$  in descending order of  $\mathcal{T}(e)$ ;
7:   for each entry  $e \in V$  such that  $\mathcal{T}(e) > \gamma$  do
8:     read the child node  $N'$  pointed by  $e$ ;
9:     BB( $N'$ );
10: else
11:   for  $c := 1$  to  $m$  do
12:     compute  $\tau_c(e)$  for all  $e \in V$  concurrently;
13:     remove entries  $e \in V$  such that  $\tau_+(e) \leq \gamma$ ;
14:   update  $W_k$  (and  $\gamma$ ) by entries in  $V$ ;

```

Algorithm 1 ITERATE (Joint query Q , Tree root $root$, Integer k)

```

1: for each subquery  $q_i$  do
2:    $V_i \leftarrow$  new max-priority queue;  $\triangleright$  maintain the top  $k$  objects
3:   Initialize  $V_i$  with  $k$  null objects with distance  $\infty$ ;
4:    $U \leftarrow$  new min-priority queue;
5:    $U.\text{Enqueue}(root, 0)$ ;
6:   while  $U$  is not empty do
7:      $e \leftarrow U.\text{Dequeue}()$ ;
8:     if  $e$  is an object then
9:       update  $V_i$  by  $(e, \text{dist}(q_i, \lambda, e, \lambda))$ ;
10:      if  $V_i$  has  $k$  non-null objects then
11:        break the while-loop;
12:     else  $\triangleright e$  points to a child node
13:       read the node  $CN$  of  $e$ ;
14:       read the posting lists of  $CN$  for keywords in  $q_i, \psi$ ;
15:       for each entry  $e'$  in the node  $CN$  do
16:         if  $q_i, \psi \subseteq e', \psi$  then
17:            $U.\text{Enqueue}(e', \text{mindist}(q_i, \lambda, e', \lambda))$ ;
18: return  $\{V_i\}$ ;  $\triangleright$  top- $k$  results of each subquery

```

We want to find the top-1 object. For each sub query q_i , ITERATE thus computes the top-1 result. Sub query $q_1 = _ \lambda, \{a, b\}$ first visits the root and loads the posting lists of words a and b in InvFile-root . Since entries R_5 and R_6 both contain a and b , both entries are inserted into the priority queue with their distances to q_1 . The next dequeued entry is R_5 , and the posting lists of words a and b in $\text{InvFile-}R_5$ are loaded. Since only R_1 contains a and

D. Spatial Query Evaluation on R-Trees

The most popular spatial access method is the R-tree [3], which indexes minimum bounding rectangles (MBRs) of objects. Fig. 2 shows a set $D = \{p_1; \dots; p_8\}$ of spatial objects (e.g., points) and an R-tree that indexes them. R-trees can efficiently process main spatial query types, including spatial range queries, nearest neighbour queries, and spatial joins. Given a spatial region W , a spatial range query retrieves from D the objects that intersect W . For instance, consider a range query that asks for all objects within the shaded area in Fig. 2. Starting from the root of the tree, the query is processed by recursively following entries, having MBRs that intersect the query region. For instance, e_1 does not intersect the query region, thus the sub tree pointed by e_1 . Given a feature data set F and a multidimensional region R , the range top- k query selects the tuples (from F) within the region R and returns only those with the k highest qualities. Hong et al. [11] indexed the data set by a MAX a R-tree and developed an efficient tree traversal algorithm to answer the query. Instead of finding the best k qualities from F in a specified region, our (range score) query considers multiple spatial regions based on the points from the object data set D , and attempts to find out the best k regions (based on scores derived from multiple feature data sets F_c).

IV. Results

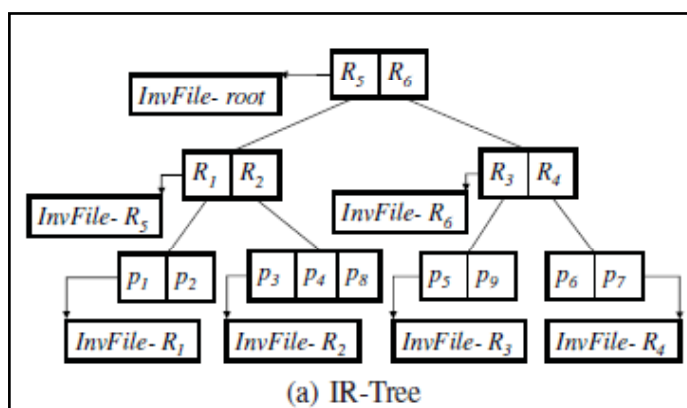


Fig. 3:

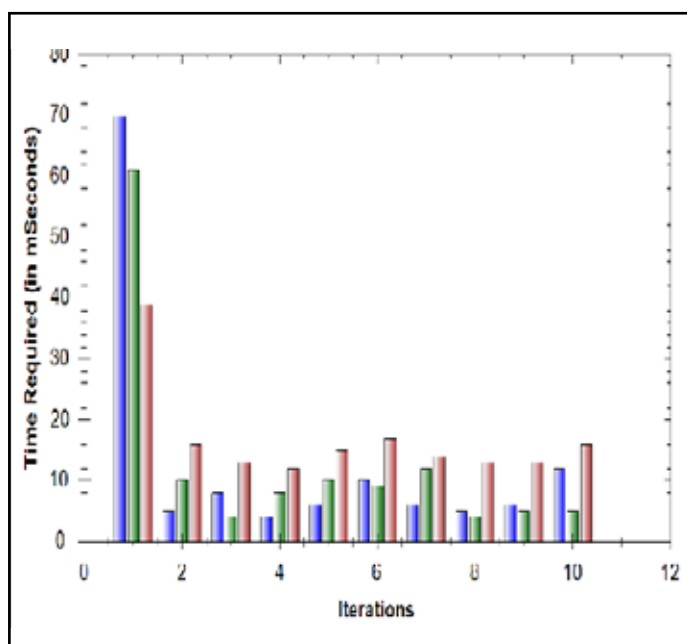


Fig. 4: Bar Chart on basis on Time Complexity

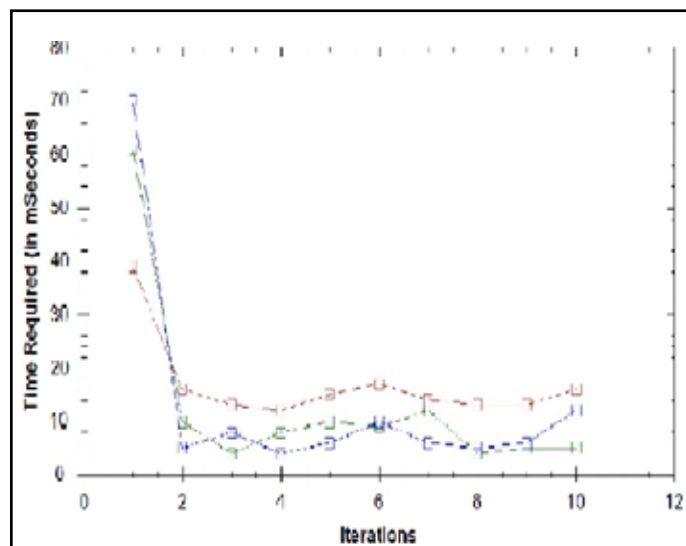


Fig. 5: Line Chart on basis on Time Complexity

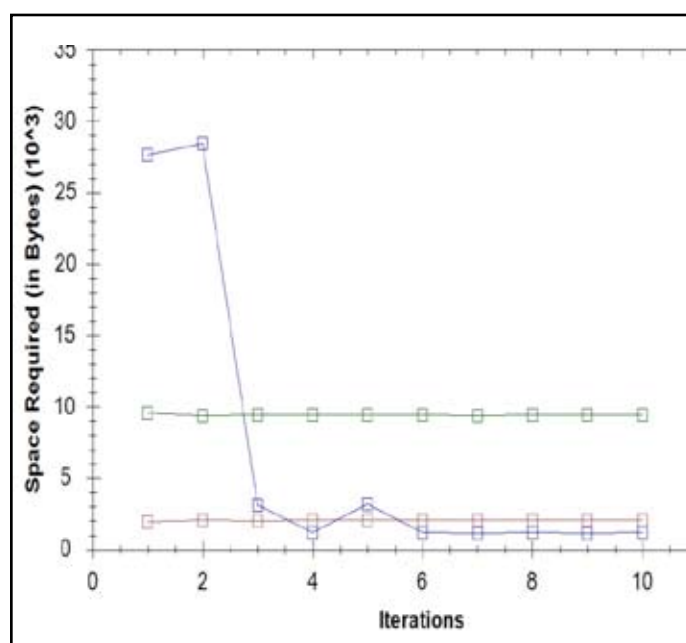


Fig. 6: Line Chart on basis on Space Complexity

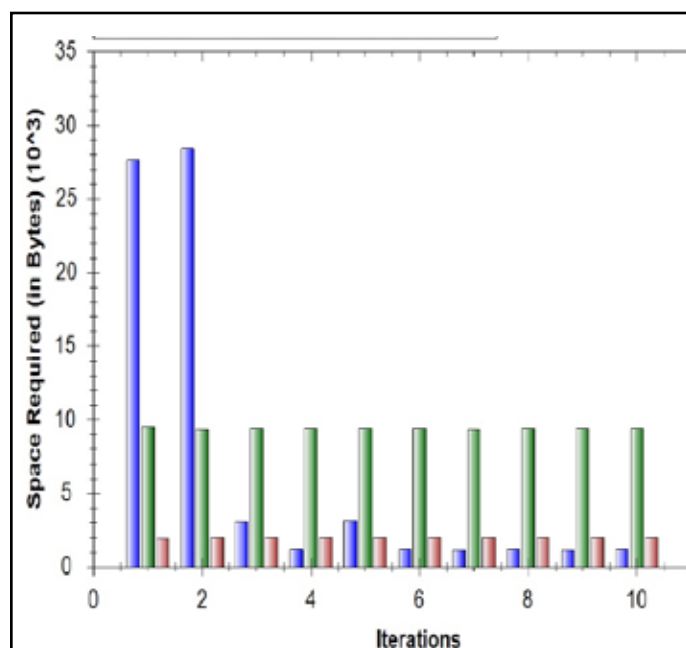


Fig. 7: Bar Chart on basis on Space Complexity

V. Conclusions

In this paper, we studied top-k spatial preference queries, which provides a novel type of ranking for spatial objects based on qualities of features in their neighbourhood. The neighbourhood of an object p is captured by the scoring function: (i) the range score restricts the neighbourhood to a crisp region centered at p , whereas (ii) the influence score relaxes the neighbourhood to the whole space and assigns higher weights to locations closer to p . We presented five algorithms for processing top-k spatial preference queries. The baseline algorithm SP computes the scores of every object by querying on feature datasets. The algorithm GP is a variant of SP that reduces I/O cost by computing scores of objects in the same leaf node concurrently. The algorithm BB derives upper bound scores for non-leaf entries in the object tree, and prunes those that cannot lead to better results. The algorithm BB* is a variant of BB that utilizes an optimized method for computing the scores of objects (and upper bound scores of non-leaf entries). The algorithm FJ performs a multi-way join on feature trees to obtain qualified combinations of feature points and then search for their relevant objects in the object tree. Based on our experimental findings, BB* is scalable to large datasets and it is the most robust algorithm with respect to various parameters.

However, FJ is the best algorithm in cases where the number m of feature datasets is low and each feature dataset is small. In the future, we will study the top-k spatial preference query on road network, in which the distance between two points is defined by their shortest path distance rather than their Euclidean distance. The challenge is to develop alternative methods for computing the upper bound scores for a group of points on a road network

References

- [1] S.S. Pradhan, J. Kusuma, K. Ramchandran, "Distributed Compression in a Dense Microsensor Network", IEEE Signal Processing Magazine, Vol. 19, No. 2, pp. 51-60, Mar. 2002.
- [2] A. Scaglione, S.D. Servetto, "On the Interdependence of Routing and Data Compression in Multi-Hop Sensor Networks", Proc. Eighth Ann. Int'l Conf. Mobile Computing and Networking, pp. 140-147, 2002.
- [3] N. Meratnia, R.A. de By, "A New Perspective on Trajectory Compression Techniques", Proc. ISPRS Commission II and IV, WG II/5, II/6, IV/1 and IV/2 Joint Workshop Spatial, Temporal and Multi- Dimensional Data Modelling and Analysis, Oct. 2003.
- [4] S. Baek, G. de Veciana, X. Su, "Minimizing Energy Consumption in Large-Scale Sensor Networks through Distributed Data Compression and Hierarchical Aggregation", IEEE J Selected Areas in Comm., Vol. 22, No. 6, pp. 1130-1140, Aug. 2004.
- [5] C.M. Sadler, M. Martonosi, "Data Compression Algorithms for Energy-Constrained Devices in Delay Tolerant Networks", Proc. ACM Conf. Embedded Networked Sensor Systems, Nov. 2006.
- [6] Y. Xu, W.-C. Lee, "Compressing Moving Object Trajectory in Wireless Sensor Networks", Int'l J. Distributed Sensor Networks, Vol. 3, No. 2, pp. 151-174, Apr. 2007.
- [7] G. Shannon, B. Page, K. Duffy, R. Slotow, "African Elephant Home Range and Habitat Selection in Pongola Game Reserve, South Africa", African Zoology, Vol. 41, No. 1, pp. 37-44, Apr. 2006.
- [8] C. Roux, R.T.F. Bernard, "Home Range Size, Spatial Distribution and Habitat Use of Elephants in Two Enclosed

Game Reserves in the Eastern Cape Province, South Africa", African J. Ecology, Vol. 47, No. 2, pp. 146-153, June 2009.

- [9] J. Yang, M. Hu, "Trajpattern: Mining Sequential Patterns from Imprecise Trajectories of Mobile Objects", Proc. 10th Int'l Conf. Extending Database Technology, pp. 664-681, Mar. 2006.
- [10] M. Morzy, "Mining Frequent Trajectories of Moving Objects for Location Prediction", Proc. Fifth Int'l Conf. Machine Learning and Data Mining in Pattern Recognition, pp. 667-680, July 2007.
- [11] F. Giannotti, M. Nanni, F. Pinelli, D. Pedreschi, "Trajectory Pattern Mining", Proc. ACM SIGKDD, pp. 330-339, 2007.
- [12] V.S. Tseng, K.W. Lin, "Energy Efficient Strategies for Object Tracking in Sensor Networks: A Data Mining Approach, J. Systems and Software, Vol. 80, No. 10, pp. 1678-1698, 2007.
- [13] L. Chen, M. Tamer Özsu, V. Oria, "Robust and Fast Similarity Search for Moving Object Trajectories", Proc. ACM SIGMOD, pp. 491-502, 2005.
- [14] Y. Wang, E.-P. Lim, S.-Y. Hwang, "Efficient Mining of Group Patterns from User Movement Data", Data Knowledge Eng., Vol. 57, No. 3, pp. 240-282, 2006.



Mr. K. Maneendra Varma is a student of Kakinada Institute of Engineering and Technology, Korangi. Presently he is pursuing her M.Tech (C.S) from this college and he received her Bachelor of Technology (B.Tech) from JNTUK, Kakinada in the year 2009. Her area of interest includes Computer Networks and Data Mining and Database Management Systems.



Mr. D. Srinivas, an efficient teacher, received M.Tech (CSE) from JNTU Kakinada is working as an Assistant Professor in Department of C.S.E, KIET Engineering College, Kakinada. He has 4 years of teaching experience. He has published many papers in both National & International Journals. His area of Interest includes Data Communications & Networks, Database Management Systems, C Programming and other advances in Computer Applications. He guided many projects for Engineering Students.