Volume 3, Issue 11, November 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Rank Assignment Algorithms for Spatial Data by Quality Preferences

I. Navakanth, T.Yashasree ,
Assistant Professor in Vignan
Institute of Technology and Science, India

Dr. C. Srinivasa Kumar
Professor, Head of the Department-CSE,
Vignan Institute of Technology and Science, India

Abstract- ***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. 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.***

Index Terms: Query Processing,Spatial Database, Traceability Matrix ,SQL, DSN, MBR, SP.

1. INTRODUCTION

Spatial Database:

Spatial database systems manage large collections of geographic entities, which apart from spatial attributes contain non spatial information (e.g., name, size, type, price etc.). We study an interesting type of preference queries, which select the best spatial location with respect to the quality of facilities in its spatial neighborhood. A database system that is optimized to store and query spatial objects: Point: a hotel, a car
Line: a road segment Polygon: landmarks, layout of VLSI
Provide structures for storage and analysis of spatial data
Data is comprised of objects in multi-dimensional space Storing spatial data in a standard database would require excessive amounts of space Queries to retrieve and analyze spatial data from a standard database would be long and Cumbersome leaving a lot of room for error. Provide much more efficient storage, retrieval, and analysis of spatial data

Why Spatial Databases?

- Provide structures for storage and analysis of spatial data
- Data is comprised of objects in multi-dimensional space
- Storing spatial data in a standard database would require excessive amounts of space
- Queries to retrieve and analyze spatial data from a standard database would be long and cumbersome leaving a lot of room for error
- Provide much more efficient storage, retrieval, and analysis of spatial data

2. LITERATURE SURVEY

2.1 Evaluating Top-k Queries over Web-Accessible Databases
2.2 Optimal Aggregation Algorithms for Middleware.
2.3 Supporting Top-k join Queries in Relational Databases.
2.4 On Computing Top-t Most Influential Spatial sites.
2.5 Progressive Computation of The Min-Dist Optimal-Location Query
2.6 Efficient Evaluation of All-Nearest-Neighbour Queries

Experiment Results:

we present the results of our experimental evaluation. We compare our ANN methods with previous ANN algorithms. Of all the previously proposed ANN methods,
the recent batch NN (BNN) and GORDER methods are considered to be the most efficient. Consequently ,in our empirical evaluations, we only compare our
methods with these two algorithms. We note that BNN and GORDER haven't actually been compared to each other in previous work. A part of the contribution that we make via our experimental evaluation is to also evaluate the relative performance of these two methods.
Spatial ranking, which orders the objects according to their

3. EXISTING SYSTEM

In the existing system there are two basic ways for ranking objects:
Non-spatial ranking, which orders the objects by an aggregate function on their non-spatial values. distance from a reference point.

4. PROPOSED SYSTEM

Traditionally, there are two basic ways for ranking objects: spatial ranking, which orders the objects according to their distance from a reference point, and non-spatial ranking, which orders the objects by an aggregate function on their non-spatial values. In the proposed system the top- k spatial preference query integrates these two types of ranking in an intuitive way.

Process Involved

The maximum quality for each feature in the neighborhood region of p, and the aggregation of those qualities. A simple score instance, called the range score, binds the Neighborhood region to a circular region at p with radius (shown as a circle), and the aggregate function to SUM. For instance, the maximum quality of gray and black points , Within the circle of p1 are 0.9 and 0.6, respectively, so the score of p1 is 0.9+0.6=1.5.Similarly, we obtain score of p2=1.0+0.1=1.1 and score of p3=0.7+0.7=1.4 hence hotel p1 is returned as the top result.

The neighborhood region in the above spatial preference query can also be defined by other score functions. A meaningful score function is the influence score.As opposed to the crisp radius ε constraint in the range score, the influence score smoothens the effect of ε and assigns higher weights to cafes that are closer to the hotel. Above figure b shows a hotel p5 and three cafes s1; s2; s3 (with their quality values). The circles have their radii as multiples of ε. Now, the score of a cafe si is computed by multiplying its quality with the weight 2-j where j is the order of the smallest circle containing si. For example, the scores of s1, s2, and s3 are 0.3/21=0.15,0.9/22=0.225 and 1.0/23=0.125 respectively. The influence score of p5 is taken as the highest value (0.225).

non-spatial values. In the proposed system the top- k spatial preference query integrates these two types of ranking in an intuitive way.

Fig. Examples of top-k spatial preference queries. (a) Range score, ε=0.2 (b) Influence score, ε=0.2 km.

In fact, the semantics of the aggregate function is relevant to the user"s query. The SUM function attempts to balance the overall qualities of all features. For the MIN function, the top result becomes p3, with the score =min{0.7,0.7}=0.7. It ensures that the top result has reasonably high qualities in all features. For the MAX function, the top result is p2, with score of p2 is =max{1.0,0.1}=1.0. It is used to optimize the quality in a particular feature, but not necessarily all of them.

5. ALGORITHMS

5.1 Spatial queries on R-trees:

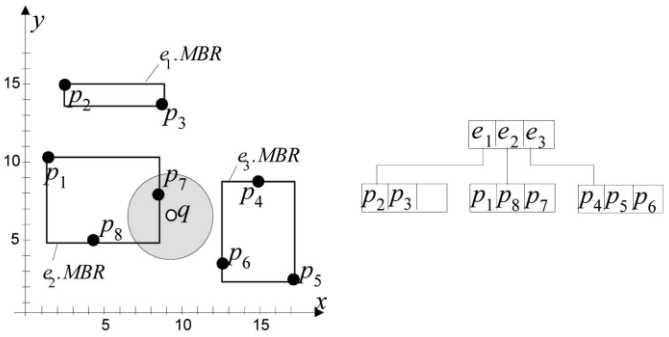
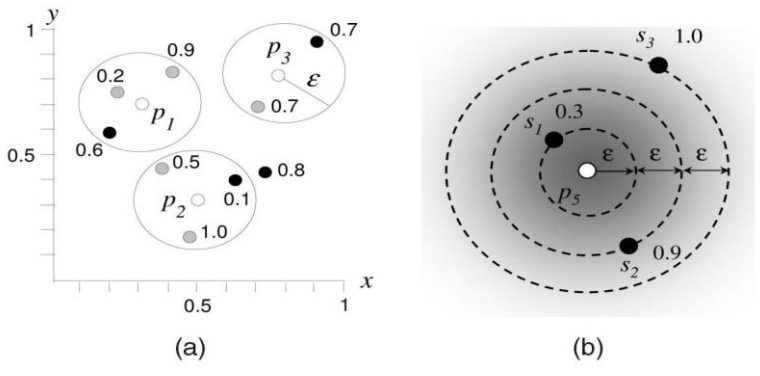


Fig 2: spatial queries on R-trees

The most popular spatial access method is the R-tree,which indexes minimum bounding rectangles (MBRs) of objects. of above Fig. shows a set D={p1,. . . p8} 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 neighbor 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 above Fig. Starting from the root of the tree, the query is processed by recursively following entries, having MBRs that intersect the query region. For instance, e1 does not intersect the query region, thus the sub tree pointed by e1 cannot contain any query result. In contrast, e2 is followed by the algorithm and the points in the corresponding node are examined recursively to find the query result p7



Algorithm 1:
Simple Probing Algorithm (SP)
Algorithm SP (Node N)
1: for each entry e ε N do
2: if N is non-leaf then
3: read the child node N1pointed by e;
4: SP(N1);
5: else
6: for c:=1 to m do
7: if Ţ + (e) > T then upper bound score
8: compute Ţ c(e) using tree Fc; update Ţ +(e);
9: if Ţ (e) > T then
10: update Wk (and T) by e;

Simple Probing Algorithm Procedure:
The above algorithm is pseudocode of the simple probing (SP) algorithm, which retrieves the query results by computing the score of every object point.
Step1: The alogorithm uses two global variables: Wk is a min-heap for managing the top-k results and T represents the top-k score so far (i.e., lowest score in Wk)
Step 2: Initially , the algorithm is invoked at the root node of the object tree (i.e.,
N= D.root)
Step 3: The procedure is recursively applied (at Line 4) on tree nodes until a leaf
Node is accessed.
Step 4: When a leaf node is reached, the component score Ţ(e) (at Line 8) is computed by executing a range search on the feature tree Fc for range score queries.
Step 5: Lines 6-8 describe an incremental computation technique, for reducing
Unnecessary component score computations.
Step 6: In particular, the point e is ignored as soon as its upper bound score Ţ+(e)
cannot be greater than the best-k score(T). The variables Wk and T are
updated when the actual score Ţ(e) is greater than T.

Algorithm 2:
Branch and Bound Algorithm (BB)
Wk: =new min-heap of size k (initially empty);T:=0; . k-th score in Wk.
Algorithm BB(Node N)
1: V :={ e/e ε N };
2: if N is non-leaf then
3: for c:=1 to m do
4: compute Ţ c (e) for all e ε V concurrently;
5: remove entries e in V such that Ţ + (e)<=T ;
6: sort entries e ε V in descending order of Ţ (e);
7: for each entry e ε V such that Ţ (e) >T do
8: read the child node N0 pointed by e;
9: BB(N0);
10: else
11: for c:=1 to m do
12: compute Ţ c (e) for all e ε V concurrently;
13: remove entries e in V such that Ţ +(e)<=T ;
14: update Wk (and T) by entries in V ;

Branch –and-Bound Algorithm :(BB)
Step 1: BB is called with N being the root node of D. if N is a nonleaf node, (Lines 3-5) compute the scores Ţ (e) for nonleaf entries concurrently.
Step 2: Recall that Ţ (e) is an upper bound score for any point in the sub tree of e.
Step 3: Based on Ţ (e) we can compute upper bound score(Ţ +(e)) of Ţ (e).
Step 4: In order to obtain points with high scores early, we sort the entries in descending order of Ţ (e) before invoking the above procedure recursively on the child nodes pointed by the entries in V.
Step 5: If N is a leaf node , we compute the scores for all points of N concurrently and then update the set Wk of the top-k results.
Since both Wk and T are global variables ,their values are updated during recursive call of BB.

Algorithm 3:
Optimized Branch and Bound Algorithm (BB)
algorithm Optimized_Group_Range(Trees
F1;F2; . . . ;Fm, Set V , Value _, Value _)
1: for c :¼ 1 to m do
2: Hc :¼ new max-heap (with quality score as key);

3: insert Fc:root into Hc;
4: _c :¼ 1;
5: for each entry p 2 V do
6: _cðpÞ :¼ 0;
7: _ :¼ 1; . ID of the current feature tree
8: while jV j > 0 and there exists a nonempty heap Hc do
9: deheap an entry e from H_;
10: __ :¼ !ðeÞ; .update threshold
11: if 8p 2 V ; mindistðp; eÞ > _ then
12: continue at Line 8;
13: for each p 2 V do . prune unqualified points
14: if ðPm
c¼1 maxf_c; _cðpÞgÞ _ _ then
15: remove p from V ;
16: read the child node CN pointed to by e;
17: for each entry e0 of CN do
18: if CN is a nonleaf node then
19: if 9p 2 V ; mindistðp; e0Þ _ _ then
20: insert e0 into H_;
21: else . update component scores
22: for each p 2 V such that distðp; e0Þ _ _ do
23: __ðpÞ :¼ maxf__ðpÞ; !ðe0Þg;
24: _ :¼ next (round-robin) value where H_ is not
empty;
25: for each entry p 2 V do
26: _ðpÞ :¼ Pm
c¼1 _cðpÞ;

At Line 7, the variable _ keeps track of the ID of the current feature tree being processed. The loop at Line 8 is used to compute the scores for the points in the set V. then deheap an entry e from the current heap H_. The property of the max-heap guarantees that the quality value of any future entry deheaped from H_ is at most !ðeÞ. Thus, the bound _c is updated to !ðeÞ. At Lines 11-12, we prune the entry e if its distance from each object point p 2 V is larger than _. In case e is not pruned, we compute the tight upper bound score _?ðpÞ for each p 2 V (by (4)); the object p is removed from V if _?ðpÞ _ _ (Lines 13-15).

Next, we access the child node pointed to by e, and examine each entry e0 in the node (Lines 16-17). A nonleaf entry e0 is inserted into the heap H_ if its minimum distance from some p 2 V is within _ (Lines 18-20); whereas a leaf entry e0 is used to update the component score __ðpÞ for any p 2 V within distance _ from e0 (Lines 22-23). At Line 24, we apply the round-robin strategy to find the next _ value such that the heap H_ is not empty. The loop at Line 8 repeats while V is not empty and there exists a nonempty heap Hc. At the end, the algorithm derives the exact scores for the remaining points of V .

Algorithm 4:

Feature Join Algorithm

Wk :¼ new min-heap of size k (initially empty);
_ :¼ 0; .kth score in Wk
algorithm FJ(Tree D,Trees F1;F2; . . . ;Fm)
1: H :¼ new max-heap (combination score as the key);
2: insert hF1:root;F2:root; . . . ;Fm:rooti into H;
3: while H is not empty do
4: deheap hf1; f2; . . . ; fmi from H;
5: if 8 c 2 ½1;m_; fc points to a leaf node
6: for c :¼ 1 to m do
7: read the child node Lc pointed by fc;
8: Find_Result(D:root, L1; . . . ; Lm);
9: else
10: fc :¼ highest level entry among f1; f2; . . . ; fm;
11: read the child node Nc pointed by fc;
12: for each entry ec 2 Nc do
13: insert hf1; f2; . . . ; ec; . . . ; fmi into H if its score is
greater than _ and it qualifies the query;
algorithm Find_Result(Node N, Nodes L1; . . . ; Lm)
1: for each entry e 2 N do
2: if N is nonleaf then
3: compute T ðeÞ by entries in L1; . . . ; Lm;
4: if T ðeÞ > _ then

5: read the child node N0 pointed by e;
6: Find_Result(N0, L1; . . . ; Lm);
7: else
8: compute _ðeÞ by entries in L1; . . . ; Lm;
9: update Wk (and _) by e (when necessary);
In case not all entries of the deheaped combination point to leaf nodes (Line 9 of FJ), we select the one at the highest level, access its child node Nc and then form new combinations with the entries in Nc. A new combination is inserted into H for further processing if its score is higher than _ and it qualifies the query. The loop (at Line 3) continues until H becomes empty.

6. IMPLEMENTATION & RESULTS

Ranking Approach to Spatial data:

Given a set D of interesting objects (e.g., candidate locations), a top-k spatial preference query retrieves the k objects in D with the highest scores. The score of an object is defined by the quality of features (e.g., facilities or services) in its spatial neighborhood. As a motivating example, consider a real estate agency office that holds a database with available flats for lease. Here "feature" refers to a class of objects in a spatial map such as specific facilities or services. A customer may want to rank the contents of this database with respect to the quality of their locations, quantified by aggregating nonspatial characteristics of other features (e.g., restaurants, cafes, hospital, market, etc.) in the spatial neighborhood of the flat (defined by a spatial range around it). Quality may be subjective and query-parametric. For example, a user may define quality with respect to Nonspatial attributes of restaurants around it (e.g., whether they serve seafood, price range, etc.).

We studied top-k spatial preference queries, which provide a novel type of ranking for spatial objects based on qualities of features in their neighborhood. The neighborhood of an object p is captured by the scoring function: 1) the range score restricts the neighborhood to a crisp region centered at p, whereas 2) the influence score relaxes the neighborhood to the whole space and assigns higher weights to locations closer to p.

7. METHOD OF IMPLEMENTATION

Fig 3: Influential sites and optimal location queries.(a) Top-k influential (b) Max-influence (c) Min-distance

Fig. shows a set of sites (white points), a set of features (black points with weights), such that each line links a feature point to its nearest site. The influence of a site pi is defined by the sum of weights of feature points having pi as their closest site. For instance, the score of p1 is 0.9 + 0.5 =1.4. Similarly, the scores of p2 and p3 are 1.5 and 1.2, respectively. Hence, p2 is returned as the top-1 influential site.

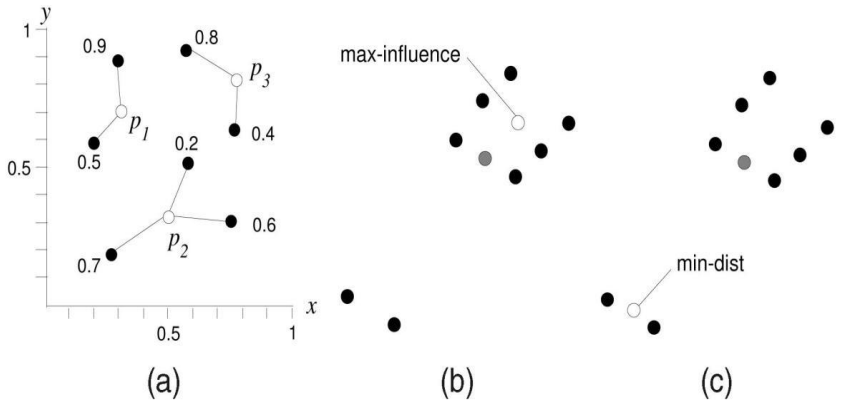


Fig 3: Influential sites and optimal location queries.(a) Top-k influential (b) Max-influence (c) Min-distance

The goal is to find the location in space (not chosen from a specific set of sites) that Minimizes an objective function. In Figs. 3b and 3c, feature points and existing sites are shown as black and gray points, respectively. Assume that all feature points have the Same quality. The maximum influence optimal location query finds the location (to insert to the existing set of sites) with the maximum influence, whereas the minimum distance optimal location query ,searches for the location that minimizes the average distance from each feature point to its nearest site. The optimal locations for both queries are marked as white points in Figs. b and c, respectively.

8. CONCLUSION & FUTURE ENHANCEMENT

We studied top-k spatial preference queries, which provide a novel type of ranking for spatial objects based on qualities of features in their neighborhood. The neighborhood of an object p is captured by the scoring function: (i) the range score restricts the neighborhood to a crisp region centered at p, whereas (ii) the influence score relaxes the neighborhood 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] M.L. Yiu, X. Dai, N. Mamoulis, and M. Vaitis, "Top-k Spatial Preference Queries," Proc. IEEE Int‟l Conf. Data Eng. (ICDE), 2007.

[2] N. Bruno, L. Gravano, and A. Marian, "Evaluating Top-k Queries over Web-Accessible Databases," Proc. IEEE Int‟l Conf. Data Eng. (ICDE), 2002

[3] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," Proc. ACM SIGMOD, 1984.

[4] G.R. Hjaltason and H. Samet, "Distance Browsing in Spatial Databases," ACM Trans. Database Systems, vol. 24, no. 2, pp. 265-318, 1999.

[5] R. Weber, H.-J. Schek, and S. Blott, "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces," Proc. Int‟l Conf. Very Large Data Bases (VLDB), 1998.

[6] K.S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is „Nearest Neighbor‟ Meaningful?" Proc. Seventh Int‟l Conf. DatabaseTheory (ICDT), 1999.

[7] R. Fagin, A. Lotem, and M. Naor, "Optimal Aggregation Algorithms for Middleware," Proc. Int‟l Symp. Principles of Database Systems (PODS), 2001.

[8] I.F. Ilyas, W.G. Aref, and A. Elmagarmid, "Supporting Top-k Join Queries in Relational Databases," Proc. 29th Int‟l Conf. Very Large Data Bases (VLDB), 2003.

[9] N. Mamoulis, M.L. Yiu, K.H. Cheng, and D.W. Cheung, "Efficient Top-k Aggregation of Ranked Inputs," ACM Trans. Database Systems, vol. 32, no.3, p. 19, 2007.

[10] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao, "Efficient OLAP Operations in Spatial Data Warehouses," Proc. Int‟l Symp. Spatial and Temporal Databases (SSTD), 2001.

[11] S. Hong, B. Moon, and S. Lee, "Efficient Execution of Range Top-k Queries in Aggregate R-Trees," IEICE Trans. Information and Systems, vol. 88-D, no. 11, pp. 2544-2554, 2005.

[12] T. Xia, D. Zhang, E. Kanoulas, and Y. Du, "On Computing Top-t Most Influential Spatial Sites," Proc. 31st Int‟l Conf. Very Large Data Bases (VLDB), 2005. 86

[13] Y. Du, D. Zhang, and T. Xia, "The Optimal-Location Query," Proc. Int‟l Symp. Spatial and Temporal Databases (SSTD), 2005.

[14] D. Zhang, Y. Du, T. Xia, and Y. Tao, "Progessive Computation of The Min-Dist Optimal-Location Query," Proc. 32nd Int‟l Conf. Very Large Data Bases (VLDB), 2006.

[15] Y. Chen and J.M. Patel, "Efficient Evaluation of All-Nearest- Neighbor Queries," Proc. IEEE Int‟l Conf. Data Eng. (ICDE), 2007.

[16] P.G.Y. Kumar and R. Janardan, "Efficient Algorithms for Reverse Proximity Query Problems," Proc. 16th ACM Int‟l Conf. Advances in Geographic Information Systems (GIS), 2008.

[17] M.L. Yiu, P. Karras, and N. Mamoulis, "Ring-Constrained Join:Deriving Fair Middleman Locations from Pointsets via a Geometric Constraint," Proc. 11th Int‟l Conf. Extending Database Technology (EDBT), 2008.

[18] http://www.computer.org/portal/web/csdl

[19] http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4384488

[20] http://www.reserarchgate.net

[21] M.L. Yiu, N. Mamoulis, and P. Karras, "Common Influence Join: A Natural Join Operation for Spatial Pointsets," Proc. IEEE Int'l Conf. Data Eng. (ICDE), 2008.

[22] Y.-Y. Chen, T. Suel, and A. Markowetz, "Efficient Query Processing in Geographic Web Search Engines," Proc. ACM SIGMOD, 2006.

[23] V.S. Sengar, T. Joshi, J. Joy, S. Prakash, and K. Toyama, "Robust Location Search from Text Queries," Proc. 15th Ann. ACM Int'l Symp. Advances in Geographic Information Systems (GIS), 2007.

[24] S. Berchtold, C. Boehm, D. Keim, and H. Kriegel, "A Cost Model for Nearest Neighbor Search in High-Dimensional Data Space," Proc. ACM Symp. Principles of Database Systems (PODS), 1997.

[25] E. Dellis, B. Seeger, and A. Vlachou, "Nearest Neighbor Search on Vertically Partitioned High-Dimensional Data," Proc. Seventh Int'l Conf. Data Warehousing and Knowledge Discovery (DaWaK), pp. 243- 253, 2005.

[26] N. Mamoulis and D. Papadias, "Multiway Spatial Joins," ACM Trans. Database Systems, vol. 26, no. 4, pp. 424-75, 2001.

[27] A. Hinneburg and D.A. Keim, "An Efficient Approach to Clustering in Large Multimedia Databases with Noise," Proc. Fourth Int'l Conf. Knowledge Discovery and Data Mining (KDD), 1998.

