© 2016 IJSRSET | Volume 2 | Issue 1 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

Efficient Framework for Group Nearest Group Query

P. Damodharan*[1], Dr. C. S. Ravichandran[2]

*[1]Department of Computer Science and Engineering, Akshaya College of Engineering and Technology, Coimbatore, Tamil Nadu, India

[2]Department of Electrical and Electronics Engineering, Sri Ramakrishna Engineering College, Coimbatore, Tamil Nadu, India

ABSTRACT

The Efficient Nearest Neighbor group query (ENNGQ) uses group points to provide the optimal solution for the nearest group point in the dataset. The novel type of spatial keyword query called Group Nearest Group (GNG) query will be used to optimize the query. Given a data point set D, a query point set Q and an integer k, the Group Nearest Group query finds a subset of points from D, ω ($|\omega| \leq k$), such that the total distance from all points in Q to the nearest point in ω is no greater than any other subset of points in D. Each nearest point obtained matches at least one of the query keywords. For processing this query several algorithms are proposed. The processing of ENNGQ query consists of a Data Classifier with De-duplicator Block (DCDB), Most Frequent Search Cache Engine (MFSCE) and Exhaustive Subset Refinement algorithm (ESRA) . A ENNGQ query returns the location of a meeting place that minimizes the aggregate distance from a spread out group of users. For example, a group of users can look for a restaurant that serves Chinese dishes that minimizes the total travel distance from them. The duplicates in the dataset can be identified to improve the search query from the given data using the DCDB.. The data can also be grouped in such a manner that it can be used for efficient retrieval. The MFSCE is a Deterministic Cache engine which helps to store the most frequently accessed keyword and the location with respect to a reference point. Finally the ESRA is a local search heuristic with the current best value is refined by visiting the child nodes of the block using bounded Priority queues. In order to prune large portion of query objects reducing the number of node accesses, this extensive experiments on spatial databases is effective in reducing group query response time which exhibits good scalability with the query objects and the number of query.

Keywords: Spatial Data Mining, Boolean spatial keyword query, reverse k Boolean spatial keyword query, Nearest Neighbor group query

I. INTRODUCTION

Keyword based Search has received lots of attention from the database research community in the past decade, due to its importance in a wide spectrum of applications such as decision support, profile - based marketing, and resource allocation. Reverse top-k Boolean spatial keyword (RkBSK) search is one of the research in Information Technology where several algorithms and theoretical performance bounds have been devised for exact and approximate processing in main memory. Due to the popularity of search services on the Internet, users are allowed to provide a list of keywords besides the spatial information of objects, which reduces scalability and an increase of query response time. Therefore there is a need for improved training methods, and virtual reality technology for processing this query, which is implemented by means of Efficient Nearest Neighbor group Query (ENNGQ) search.

The RkBSK query assumes objects on the road network, and returns the objects having a specified query point q as one of the answer objects for the top-k Boolean spatial keyword (TkBSK) query. The significance of RkBSK queries is that it retrieves both spatial and textual information to prune the search space significantly is utilized. It can tackle exact RkBSK retrieval with an arbitrary k, without any pre-computation. Initially, the set of Data points, containing the keyword information of query object and the query

keyword should be given by the User. By ENNGQ query, each nearest point matches at least one of the query keywords of the User.

1.1 About Spatial Data Mining

Spatial data mining is the process of discovering interesting and previously un- known, but potentially useful patterns from large spatial datasets. Extracting interesting and useful patterns from spatial datasets is more difficult than extracting the corresponding patterns from traditional numeric and categorical data due to the complexity of spatial data types, spatial relationships, and spatial autocorrelation. Standard data mining algorithms can be modified for spatial data mining, with a substantial part of preprocessing to take into account of spatial information. The main difference between data mining in relationaldata base system and in spatial database system is that attributes of the Neighbors of some object of interest mayhave an influence on the object and therefore have to be considered as well.

The explicit location and extension of spatial objects define implicit relations of spatial Neighborhood (such as topological, distance and direction relations) which are used by spatial data mining algorithms. Therefore, new techniques are required for effective data mining. Spatial data is stored in spatial databases. Multidimensional trees are used, in order to build indices for these data (e.g. quad trees, k-d trees, R-trees, R*-trees). Often attributes of spatial objects are still one dimensional, so that this non-spatial part can be stored inrelational databases with references to the spatial data. Spatial operations like spatial join and map overlay are the most expensive. Spatial analysis or spatial statistics includes any of the formal techniques which study entities using their topological, geometric, or geographic properties.

Complex issues arise in spatial analysis, many of which are neither clearly defined nor completely resolved, but form the basis for current research. The most fundamental of these is the problem of defining the spatial location of the entities being studied. Other issues in spatial analysis include the limitations of mathematical knowledge, the assumptions required by existing statistical techniques, and problems in computer based calculations.

1.2 About Nearest Neighbor Search Using Rkbsk Query – Baseline Method

Nearest Neighbor search (NNS), also known as proximity search, similarity search or closest point search, is an optimization problem for finding closest points in metric spaces. The problem is: given a set S of points in a metric space M and a query point $q \in M$, find the closest point in S to q. In metric space, there is a valid concept of distance between points. The distance between two features is calculated by calculating the points distance between them. The lesser the distance between them, the more similar they are in appearance.
Various solutions to the Nearest Keyword Search problem have been proposed. The quality and usefulness of the algorithms are determined by the time complexity of queries as well as the space complexity of any search data structures that must be maintained. The informal observation usually referred to as the curse of dimensionality states that there is no general-purpose exact solution for NNS in high-dimensional Euclidean space using polynomial preprocessing and poly logarithmic search time.

An important subclass are the local search methods, that view the elements of the search space as the vertices of a graph, with edges defined by a set of heuristics applicable to the case; and scan the space by moving from item to item along the edges. Another class includes various tree search algorithms, which view the elements as vertices of a tree and traverse the tree in some special order. Examples include the exhaustive methods such as depth-first search and breadth-first search, as well as various heuristic-based search tree pruning methods such as backtracking and branch and bound.
An increasing number of applications require the efficient execution of nearest Neighbor (NN) queries constrained by the properties of the spatial objects. Due to the popularity of keyword search, particularly on the Internet, many of these applications allow the user to provide a list of keywords that the spatial objects (henceforth referred to simply as objects) should contain, in their description or other attribute. For example, online yellow pages allow users to specify their address and a set of keywords, and return businesses whose description contains these keywords, ordered by their distance to the specified address location. As another example, real estate web sites allow users to search for properties with specific keywords in their description

and rank them according to their distance from a specified location.

RkBSK retrieval, which assumes objects are on the road network and considers both spatial and textual information. Given a data set P on a road network and a query point q with a set of keywords, an RkBSK query retrieves the points in P that have q as one of answer points for their top-k Boolean spatial keyword queries. The RkBSK query is enhanced by using a proposed filter-and-refinement framework based algorithms for answering RkBSK search with arbitrary k and no any pre-computation. To accelerate the query process, several novel pruning heuristics that utilize both spatial and textual information are employed to shrink the search space efficiently. The Enhanced RkBSK algorithm actually shrinks the expanded network area and meanwhile reduces the size of candidate set.

Reverse k nearest neighbor (RkNN) queries have a broad application base such as decision support, profile-based marketing, and resource allocation. RkBSK queries constitute a suite of interesting and practical problems from not only the research point of view but also the application point of view. It takes into account both the reverse spatial proximity between objects on road networks and the textual constraint.

1.3 Objective – Problem Definition

The objective of "Efficient Nearest Neighbor Group Query Optimization is to provide the finest solution for the Nearest keyword search query on group point in the dataset. In nearest Neighbor queries, an optimization problem is evaluated for finding the closest points in metric spaces. Given a data point set D, a query point set Q and an integer k, the Group Nearest Group query finds a subset of points from D, ω ($|\omega| \leq k$), such that the total distance from all points in Q to the nearest point in ω is no greater than any other subset of points in D. The processing focus of our approaches is to minimize the access and evaluation of subsets of cardinality k in D since the number of such subsets is exponentially greater than the dataset.

The data to be searched is classified using DCDB such that all related entities forming a sub-group are mapped to a parent group. The deduplication on the data-group ensures atomicity and uniqueness of data. The most frequent searches performed using MFSCE are stored in a Deterministic cache memory for efficient retrieval of repeatedly occuring queries. This can improve the performance of frequently issued keyword search queries.

The final solution is obtained by following the guided search direction at low level so as to prune irrelevant subsets and grouping them to form a Subset Refinement Tree (SRT). The ESRA searches the SRT based on keyword until the best match is found. Bounded Priority Queues are used to store the nodes and data points that are scanned.

II. LITERATURE SURVEY

The range nearest-Neighbor (RNN) query retrieves the nearest Neighbor (NN) for every point in a range.. These techniques are generalized for kRNN queries, which return the k nearest Neighbors for every point in the range [1]. In general, processing an NN query on a spatial index involves two interleaving phases:

- secondary memory pruning of distant index nodes.
- In-memory computation of the nearest Neighbors.
- Determining the nearest data points from a query point satisfying the keyword search criteria.

2.1 Efficient Retrieval Of The Top-K Mostrelevent Spatial Web Objects

Location-based instant search that combines location based keyword search with instant search is formulated. Nearest Neighbor (NN) queries on a spatial database is a classical problem. The k-NN algorithm for R-trees traverses an R-tree while maintaining a list of k potential nearest Neighbors in a priority queue in a Depth-First (DF) manner. The closest pair queries (CPQ) are a combination of spatial join and nearest Neighbor queries, which find the pair with the minimum distance among all pairs from two data sets. The difference between nearest Neighbor queries and closest pair queries is that the algorithms of the latter access two index structures (one for each data set) and utilize the distance function of the two intermediate nodes to prune the pairs.NNK specifies only one query location specifies a set of query locations.

2.2 Keyword Search On Spatial Databases

The spatial data search on k nearest Neighbor queries is based on the Revived R*-tree index structure. The

incremental methods for search have the following drawbacks:

- They cannot support objects in multidimensional space
- Their methods are very inefficient for incremental query.

To solve such search problem efficiently, the novel incremental search on spatial data is applied to multidimensional spatial databases. The counter for every entry of RR*-tree index structure, which marks the number of nearest Neighbor and thus offers the information about the influences of a query point.

III. MODULES AND ITS DESCRIPTION – PROPOSED METHOD

The "Efficient Nearest Neighbor Group Query Optimization is to provide the finest solution for the Nearest keyword search query on group point in the dataset. This is achieved using the following modules.

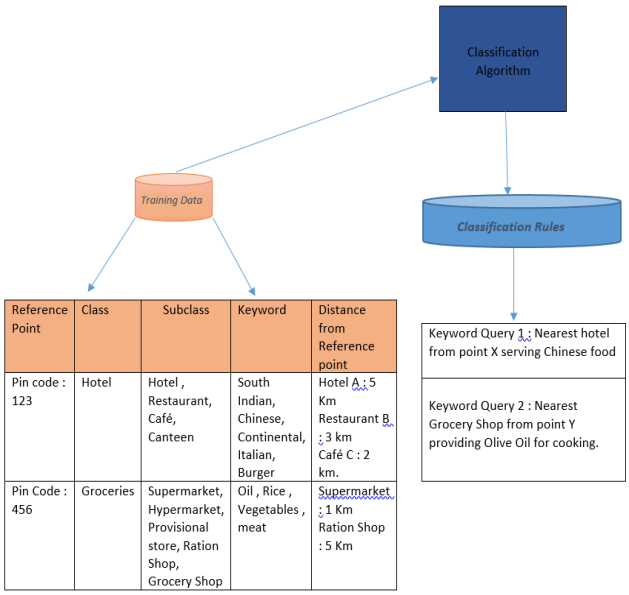
3.1 Data Classifier And De-Duplication Block

Data classification is the process of organizing data into categories for its most effective and efficient use. It is performed by sorting and categorizing data into various types, forms or any other distinct class. Data classification enables the separation and segmentation of data according to data set requirements for various business objectives. In the case of clustering, the segmentation is done using similarities between the feature variables, with no prior understanding of the structure of the groups. In the case of classification, the segmentation is done on the basis of a training data set, which encodes knowledge about the structure of the groups in the form of a target variable.

Deduplication is ideal for highly redundant operations like backup, which requires repeatedly copying and storing the same data set multiple times for recovery purposes. As a result, enterprises of all sizes rely on backup and recovery with deduplication for fast, reliable, and cost-effective backup and recovery. Deduplication segments an incoming data stream, uniquely identifies data segments, and then compares the segments to previously stored data. If the segment is unique, it's stored on disk. However, if an incoming data segment is a duplicate of what has already been stored, a reference is created to it and the segment isn't stored again.

The Data classifier unit is built from the training set made up of database tuples and their associated class labels. Each tuple that constitutes the training set is referred to as a category or class. These tuples can also be referred to as sample, object or data points. Figure 1. illustrates how a data Classifier module is designed and how a Classification algorithm can be used to fetch the data from the Data Classifier. The training data module contains all the data points grouped in different classes and which are uniquely mapped to the sub-class. The Classification Algorithm gathers the required data from the training set which satisfies the Classification rule for a particular Keyword Query.

The Data De-duplicator block shall ensure that there are no duplicate entries stored and retrieved from the training data sets. The De-duplication operation further ensures that each keyword query is associated with a unique sub-class and which is mapped in turn to a unique class of data. This I responsible for the atomicity of data retrieved from the Classifier unit.



Reference Point	Class	Subclass	Keyword	Distance from Reference point
Pin code : 123	Hotel	Hotel , Restaurant, Café, Canteen	South Indian, Chinese, Continental, Italian, Burger	Hotel A : 5 Km Restaurant B : 3 km Café C : 2 km.
Pin Code : 456	Groceries	Supermarket, Hypermarket, Provisional store, Ration Shop, Grocery Shop	Oil , Rice , Vegetables , meat	Supermarket : 1 Km Ration Shop : 5 Km

Figure 1 : A Data Classifier Module

3.2 Most Frequent Search Cache Engine

With may search Algorithms, we have to make tradeoffs between the amount of data we have and the speed at which we can access it. The more the data, the lesser the speed to access it. The Most Frequent Search Cache Engine is a deterministic Online algorithms for caching which helps to store the most frequently accessed

keyword and the location with respect toa reference point (x, y).

The algorithm works as follows:

- Initially, all cache pages are unmarked;
- Whenever a spatial query is issued:
- If the query is in the cache, mark the page which contains the query;

Otherwise:

If there is at least one unmarked page in the cache, evict an arbitrary unmarked page, bring the requested query.
As an example of how the MFSCE increases performance of a NN query, consider 4 friends planning to dine at a common restaurant which serves south Indian dish. Assume that these 4 friends are coming from 4 different places located nearby. Each friend will issue the same nearest Neighborhood query to trace the restaurant. The Most Frequent Search Cache Engine will store the search results identified by first friend in its cache memory. The remaining 3 friends can get the query results from the cache which results in faster retrieval of the keyword query.

3.3 Exhaustive subset refinement algorithm

The ESRA consist of a Subset Refinement Tree (SRT) which is very similar to a k-d-tree. The internal nodes have a split dimension and a split point. The internal nodes partition the data space into disjoint regions based on some splitting rule. Every internal node has two children and leaf nodes contain the data points. Unlike the standard k-d-tree where a node holds a single data point, the ESRA has leaves that contain subset arrays of fixed size. Each subset array can be filled with data items until the array is full. When a new point is inserted into a full array, the subset array will overflow. New subset arrays are then created. The overflowing subset array is replaced by a new internal node and two another subset arrays are attached to this node. Data points in the old subset arrays are distributed over the new subset array. To determine the distribution, the split dimension and split point have to be determined first. This can be done solely based on the data space of the subset array, either data dependent or distribution dependent.

The ESRA uses two bounded priority queues to determine which nodes should be visited: one queue for nodes (NQ) and one queue for data points (DQ). The priority is determined by the distance from the node or data point to the query point. Every time we descend in the tree its sibling node is stored in the node priority queue. Points are added to the queue for data points when a subset array is processed. The trees are searched as long as less than the maximum number of leaves have been visited and the best item in NQ is better than the worst item in DQ.

The input consists of the roots of all trees in the forest (trees), the number of required nearest Neighbors (nn), two empty queues DQ (Data Object Priority Queue) and NQ (Node Priority Queue) and our query point q. NQ holds nodes as where DQ holds data points. The priority is determined by the distance to q. The entire forest has just two queues so we search in the tree that contains the node with highest priority (smallest distance) to q. The output is result set R that will contain the nn approximate nearest Neighbors to q.

In the main loop the trees are searched as long as the best item in NQ is better than the worst item in DQ. Every iteration the node closest (based on the distance between its bounding box and the query point) to the query point is removed from NQ and used to traverse the tree. As long as this node has a left child (and thus a right child) the algorithm has to determine if the left or right subtree should be taken. This is determined on the value of the query in the split dimension of the node.

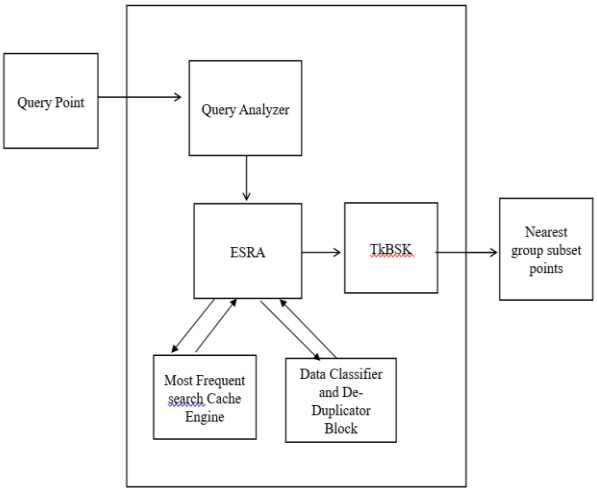


Figure 2 : Architecture Diagram of ENNGQ

ALGORITHM : ESRA

INPUT : Empty priority queues NQ and DQ, number of nearest Neighbors nn, roots of all treesin trees and query point q.

OUTPUT: Result set R filled with nn approximate nearestNeighbors.

1. for 0 to nn do // Fill queue to prevent checks
2. DQ.add(NULL,∞);
3. foreach w ϵ trees do // Push all roots in queue NQ.add(w, 0);
4. while dist(NQ_{min}, q) <dist(DQ_{max}, q) and NQ.size() > 0 do
5. w = NQ_{min};
6. NQ.deleteMin();
7. while wl do // Node is not a subset Array
8. s_{dim} = split dimension stored in w;
9. s_{point} = split point stored in w;
10. if q[s_{dim}] ≤ spoint then
11. NQ.add(w_r, dist(w_r, q));
12. w = w_l ; // Take left branch
13. else
14. NQ.add(w_l , dist(w_l , q));
15. w = w_r; // Take right branch

// We reached a subset array

16. foreach o ϵ subset do
17. if not isProcessed(o) then
18. if dist(o, q) <dist(DQ_{max}, q) then
19. DQ.deleteMax();
20. DQ.add(o, dist(o, q));
21. for 0 to nn do
22. R.add (D_{min});
23. DQ.deleteMin();

At the start of our algorithm we put some dummy data in DQ such that it always holds nn. Otherwise every time an item is added to DQ the size of the queue should be checked to determine if the worst item from DQ should be removed. All roots are stored in NQ.We use a distance of 0 since the distance from the root to the query point will be very small (in most cases 0): The root contains the entire search space, of which its boundary is defined by the bounding box of all points, and it is very likely our query point falls within this bounding box. By choosing 0 we limit the amount of distance calculations.

IV. EXPERIMENT SETUP AND RESULTS

The real and synthetic data sets were deployed emulating real road networks as the data sets. For these datasets, POIs with real keyword sets are randomly generated in a. The first data set is to study the impact of the keyword set size per data point on the search performance. The average distinct number of keywords per data point in each dataset is roughly 2, 4, 6, 8, and 10, respectively. The second data set is to explore the impact of data point density on the search performance. The experiments investigate the performance of the proposed algorithms under a variety of parameters:

(i)The response time
(ii) The number of node accessed by various algorithms during the search; and
(iii)The number of data points pruned by each pruning

Heuristic and located accurately.

The comparison among the three common query mechanism ie. ENNGQ, RkBSK and Enhanced RkBSK is shown in Figure 3, 4 and 5. As seen the ENNGQ has marginal improvement amongst other queries in terms of number of POI scanned, CPU response time as well as the Accuracy of the search. The performance of the ENGGQ query is highest when the number of query keyword is small in which case the nearest Neighbors are retrieved from the cache engine. The search accuracy of the ENGG query is also the highest when there are few keywords which is usually the case most of the time. The number of POI identified in case of ENGG also has a considerable improvement in comparison with the other search queries.

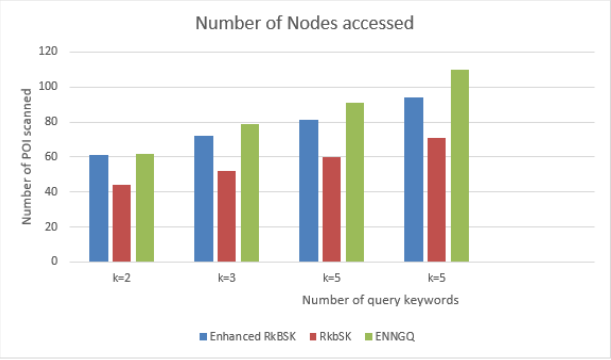


Figure 3 : Number of POI searched vs Number of Query Keywords

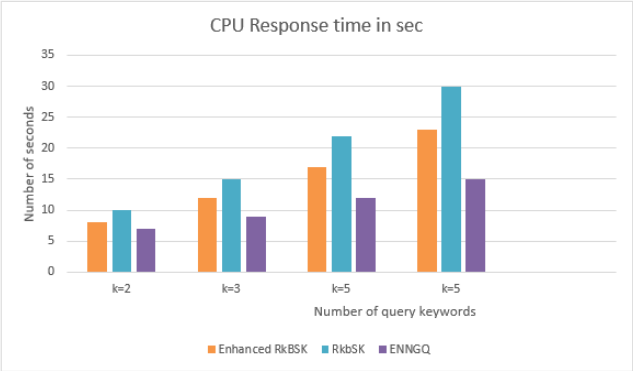


Figure 4 : Response Time vs Number of Query Keywords

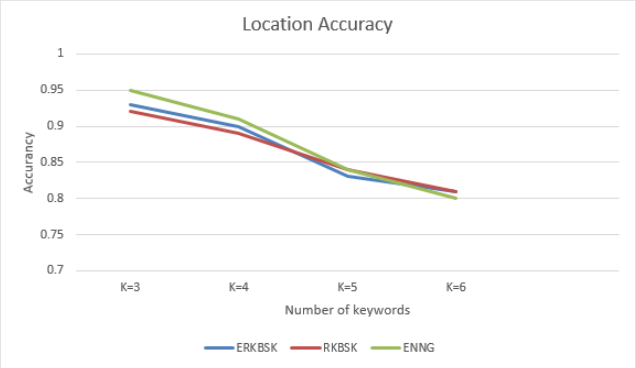


Figure 5 : Accuracy vs Number of query Keywords

V. CONCLUSION

The Efficient Nearest Neighbor group query retrieves number of objects from Query keyword Q with minimum sum of distances to its nearest Data points with integration . Reverse top-k Boolean spatial keyword and Filter and Refinement framework algorithm, prunes the query objects and finally the minimized summed distance is calculated. The data classifier with de-duplicator block helps to eliminate the duplicate nodes and also ensure that all possible related keywords are scanned. On an average the ENGGQ returns twice as number of Search results for each query as done by RkBSK or any other Nearest Neighborhood query. The MFSCE ensures that performance and response time is improved for queries which have more than one occurrence and which is usually the case with most Keyword search queries. The Exhaustive Subset refinement Algorithm exhibits good scalability and accuracy with the query objects and the number of query keywords.

ESRA is also effective for Score based spatial keyword query which aims to retrieve the k objects with the highest ranking scores, measured as a combination of their distances to the query location (a point) and the relevance of their textual descriptions to the query keywords In future work the time constraint will be introduced to improve the accuracy level, response time and further the data sets based on textual data also can be integrated on their geographical location with its graphical map points of the scale (x, yaxis) will be updated to prune the search time of the algorithm. Second, the multi-source NN query can be explored for multiple query points, whichcan be regarded as the group version of RkBSK search. Finally, in addition to road networks, how to use NN queries on social networks is also worth considering.

VI. REFERENCES

[1]. YunjunGao,Baihua Zheng and Gang Chen "Efficient Reverse Top-k Boolean Spatial Keyword Queries on Road Networks". Ieee transactions on knowledge and data engineering, VOL. 27, NO. 5, MAY 2015

[2]. AlexandrAndoni and Piotr Indyk. “Near-optimal hashing algorithms for approximate nearest Neighbor in high dimensions”. In: Foundations of Computer Science, 2006. FOCS’06. 47th Annual IEEE Symposium on. IEEE. 2006, pp. 459-468.

[3]. Sunil Arya et al. “An optimal algorithm for approximate nearest Neighbor searching fixed dimensions”. Journal of the ACM (JACM) 45.6 (1998), pp. 891-923.

[4]. Wei Dong, Charikar Moses, and Kai Li. “Efficient k-nearest Neighbor graph construction for generic similarity measures”. In: Proceedings of the 20th international conference on World wide web. ACM. 2011,pp. 577-586.

[5]. X. Cao, L. Chen, G. Cong, C. S. Jensen, Q. Qu, A. Skovsgaard, D. Wu, and M. L. Yiu, “Spatial keyword querying,” in Proc. Int.Conf. Conceptual Model., 2012, pp. 16-29.

[6]. A. Cary, O. Wolfson, and N. Rishe, “Efficient and scalable method for processing top-k spatial Boolean queries,” in Proc. Int. Conf. Sci. Statistical Database Manage., 2010, pp. 87-95.

[7]. Y. Gao, B. Zheng, G. Chen, W.-C. Lee, K. C. Lee, and Q. Li, “Visible reverse k-nearest Neighbor query processing in partial databases,”

IEEE Trans. Knowl. Data Eng., vol. 21, no. 9, pp. 1314-1327, Sep. 2009.

[8]. F. Korn and S. Muthukrishnan, "Influence sets based on reverse nearest Neighbor queries," in Proc. ACM SIGMOD Int. Conf.Manage. Data, 2000, pp. 201-212.

[9]. M. A. Cheema, W. Zhang, X. Lin, Y. Zhang, and X. Li, "Continuous reverse k nearest Neighbors queries in Euclidean space and in spatial networks," VLDB J., vol. 21, no. 1, pp. 69-95, Feb. 2012.

[10]. G. Li, Y. Li, J. Li, L. Shu, and F. Yang, "Continuous reverse k nearest Neighbor monitoring on moving objects in road networks," Inf. Syst., vol. 35, no. 8, pp. 860-883, Dec. 2010.

[11]. Kiana Hajebi et al. "Fast approximate nearestNeighbor search with k-nearest Neighbor graph". In: IJCAI Proceedings- International Joint Conference on Artificial Intelligence. Vol. 22. 1. 2011, p. 1312.

[12]. KeinosukeFukunaga and Patrenahalli M Narendra. "A branch and bound algorithm for computing knearestNeighbors". Computers, IEEE Transactions on 100.7 (1975), pp. 750-753.

[13]. D. Zhang, K. L. Tan, and A. K. H. Tung, "Scalable top-k spatial keyword search," in Proc. Int. Conf. Extending Database Technol., 2013, pp. 359-370.

[14]. J. Lu, Y. Lu, and G. Cong, "Reverse spatial and textual k nearest Neighbor search," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2011, pp. 349-360.

[15]. J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Norvag, "Efficient processing of top-k spatial keyword queries," in Proc. Int. Symp. Advances Spatial Temporal Databases, 2011, pp. 205-222.