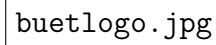


B.Sc. Engineering Thesis

Spatial Alarms in Obstructed Space

A square box containing the text "buetlogo.jpg", indicating a missing image file.

buetlogo.jpg

By

Sezana Fahmida

Student No.: 1005057

Md Touhid Zaman

Student No.: 1005018

Submitted to

Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka-1000, Bangladesh

July, 2014

Declaration

Supervisor's Declaration

I confirm that, to the best of my knowledge, the research was carried out and the thesis was prepared under my direct supervision. It represents the original research work of the candidates. The contribution made to the research by me and by other members of staff of the University was consistent with normal supervisory practice. I have read this work and in my opinion this work is adequate in terms of scope and quality for the purpose of awarding the degree of Bachelor of Science in Computer Science and Engineering.

Tanzima Hashem, PhD

Assistant Professor

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology (BUET)

Dhaka-1000.

Candidates' Declaration

We confirm that, the work presented in this thesis entitled “**Spatial Alarms in Obstructed Space**” is the outcome of the investigations carried out by us under the supervision of Assistant Professor Dr. Tanzima Hashem in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka. We also declare that, neither this thesis nor any part thereof has been submitted or is being currently submitted anywhere else for the award of any degree or diploma.

Samiha Samrose

Student No.: 0805066

Mohammad Hafiz Uddin

Student No.: 0805004

Iftexhar Mahmud

Student No.: 0805102

Contents

| | |
|------------------------------------------------------------------------------------------------------------|------------|
| Declaration | i |
| Acknowledgments | vi |
| Abstract | vii |
| 1 Introduction | 1 |
| 1.1 Problem Setup | 2 |
| 1.2 Solution Overview | 3 |
| 1.3 Contributions | 3 |
| 1.4 Thesis Organization | 4 |
| 2 Related Works | 5 |
| 2.1 Road Alarm | 6 |
| 2.2 Energy Efficient Middle Ware Architecture for processing spatial alarms on Mobile Clients | 8 |
| 2.3 Group Nearest Neighbor Queries in Obstructed Space | 9 |
| 2.4 Group Trip Planning Queries | 11 |
| 2.5 R-Tree: An efficient indexing Method | 12 |
| 3 Our Approach | 13 |
| 3.1 Selection of Initial POIs | 14 |
| 3.2 Refinement of Search Region | 15 |
| 3.3 Optimal Answer Computation | 17 |
| 3.4 Algorithms | 17 |
| 3.5 Handling Flexible k GTP Queries | 19 |

| | | |
|----------|--------------------------------------------------|-----------|
| 4 | Naive Solution | 22 |
| 4.1 | Overview | 22 |
| 4.2 | Algorithm | 22 |
| 4.2.1 | $r_k p$ Computation | 23 |
| 4.2.2 | $r_k o$ computation | 23 |
| 4.2.3 | Visibility graph generation | 24 |
| 4.2.4 | Recomputation on every location change | 24 |
| 5 | Experimental Study | 25 |
| 5.1 | Experiments | 25 |
| 5.1.1 | Aggregate function SUM | 26 |
| 5.1.2 | Aggregate function MAX | 29 |
| 5.1.3 | Comparison between SUM and MAX | 33 |
| 6 | Conclusion | 34 |
| 6.1 | Future Directions | 34 |
| | References | 35 |

List of Figures

| | | |
|-----|------------------------------------------------------------------------------------------------|----|
| 2.1 | MBR corresponding to a R-tree | 12 |
| 3.1 | Centroid method | 14 |
| 3.2 | Refined search region | 16 |
| 3.3 | Optimal answer selection | 18 |
| 4.1 | Naive approach | 23 |
| 5.1 | Effect of group size n for California data (a) I/Os and (b) query processing time | 27 |
| 5.2 | Effect of answer set k for California data (a) I/Os and (b) Query processing time | 28 |
| 5.3 | Effect of query area M for California data (a) I/Os and (b) query processing time | 28 |
| 5.4 | Effect of varying dataset size (a) I/Os and (b) query processing time | 29 |
| 5.5 | Effect of group size n for California data (a) I/Os and (b) Query processing time) | 30 |
| 5.6 | Effect of answer set k for California data (a) I/Os and (b) Query processing time | 31 |
| 5.7 | Effect of Query Area M for California data (a) I/Os and (b) Query processing time | 31 |
| 5.8 | Effect of synthetic data set (a) I/Os and (b) Query processing time | 32 |

List of Tables

| | | |
|-----|---------------------------------------------------------------------------------------------------------|----|
| 5.1 | Values of different query parameters used in our experiments . . | 26 |
| 5.2 | Parameters of synthetic datasets | 26 |
| 5.3 | Average I/Os for $kGTPQ_{sum}$ and $kGTPQ_{max}$ for different pa- rameters | 32 |
| 5.4 | Average query processing time for $kGTPQ_{sum}$ and $kGTPQ_{max}$ for different parameters | 32 |

Acknowledgments

First of all, we would like to express our deepest gratitude to our supervisor Dr. Tanzima Hashem for introducing us to the fascinating world of spatial databases, guiding us through the path of conducting successful research and above all for always being there as our mentor. She shared her wisdom with us in analyzing subject matters and at the same time valued our thinking approach to synthesize those topics. Her suggestions drove us towards better ways of thinking, her reviews enriched us in solving research problems, and her support gave us strength at the time of our disappointment. We shall forever cherish the memories of working with her.

We would like to sincerely thank Sukarna Barua for his guidance and encouragement in developing our thesis work. Our very special thanks remain for Dr. Mohammed Eunus Ali for all his valuable reviews and remarks towards the dissertation. During the thesis work evaluation process, the teachers of our department made valuable comments which helped us a lot in refining our work. We express our gratitude to them all.

We are indebted to Md. Shahin Pramanik, a member of the Samsung Innovation Laboratory, who provided us with a congenial research environment in the lab while we were conducting our work there.

Last but not the least, we deeply thank our friends and families for always believing in us even at the moment when we were losing our believes in ourselves. We are forever grateful to them for their selfless support and understanding.

Abstract

The proliferation of location-based social networks allows people to access location-based services as a group. In this thesis, we introduce a novel approach to process Group Trip Planning (GTP) queries in road networks, which enable a group to plan a trip with a minimum aggregate trip distance. The trip starts from the source locations of the group members, goes via different point of interests (POIs) such as a restaurant, shopping center and movie theater, and ends at the destination locations of the group members. The aggregate trip distance can be the total or the maximum trip distance of the group members. We exploit elliptical properties to refine the search region for the POIs and develop efficient algorithms to evaluate GTP queries. Our experimental results show that our approach outperforms a naive approach significantly both in terms of query processing time and I/Os. Our approach is on average 17 and 40 times faster than the naive approach and requires 51 and 58 times less I/Os than the naive approach for minimizing total and maximum trip distances, respectively.

Chapter 1

Introduction

The high availability of smart phones has led to the proliferation of location based services. According to many, the next step in location based services is Spatial Alarms. Many believe this particular feature is going to dominate the future mobile-phone computing systems. Spatial alarms are an extension of time-based alarms. It is, however, triggered by a specific location irrespective of time. "Remind me if I'm within 100 meters of a pharmacy" is a possible example of a spatial alarm. It is a personalized location based service which can vary from user to user. Existing research has categorized spatial alarms into three types: public, shared and private. Public alarms are alarms which are active for every user within the system, such as an alarm must be sent to everyone within 100 meters of a building on fire. Private alarms are user defined alarms which are only viewable by the user herself, such as a user might set an alarm to alert her if she is within 100 meters of her favourite coffee shop. Shared alarms are shared between specific groups of people. In the previous example if a user chooses to share the alarm for the coffee shop with some of her friends it becomes a shared spatial alarm.

It is noteworthy that spatial alarms are quite dissimilar to spatial range query. Spatial alarms are based on a fixed location thus applying the techniques that are used in answering spatial range queries is both inefficient and wasteful for the two dominating reasons, Firstly, in spatial range query as the user is the main point of interest, continuous re-evaluation of her location is needed in case of a mobile user. In contrast, spatial alarms need only be re-

evaluated when the user is approaching a specific location. Secondly, in spatial alarm, the main point of interest is a static location. So the user’s location is not relevant at all times. It is quite clear that applying spatial range query techniques in evaluating spatial alarms is going to result in wastage of resources. If we start to evaluate spatial alarms as soon as the user is on the move even if the user is far away from her desired location our efforts will be futile. Existing work in this area has focused mainly on Euclidean distance and road network models, to the best of our knowledge; no work is yet done on spatial alarms in obstructed space. Spatial alarm evaluation in obstructed space is different than road network or Euclidean space as it considers the obstacles in the path to the location of alarm. It is better approximated by a pedestrian scenario while road networks are approximated by vehicle scenarios. A vehicle can only go to a specific location following a predefined road, but a pedestrian’s path is not limited by roads. However, a pedestrian is obstructed by various obstacles such as buildings or trees. So while calculating the distance from spatial alarms, we have to consider the obstructed distance.

Spatial alarms are location-based, user-defined triggers which will possibly shape the future mobile application computations. They are distinct from spatial range query and do not need immediate evaluation after the user has activated them. The spatial alarm evaluation strategies are judged based on two features, High accuracy and High system scalability. High accuracy refers to the quality that guarantees no alarms are missed. And High scalability is the feature that ensures that the system can adapt to a large number of spatial alarms. In this paper, We propose a novel approach to evaluate spatial alarms in obstructed space which ensures both high accuracy and high scalability.

1.1 Problem Setup

Obstructed Space Route Problem denotes the problem of finding the shortest route between two query-points in Obstructed Space where non-intersecting 2D polygons represent *obstacles* and where the route does not traverse through any obstacles. The length of the Obstructed route between points a and b is called the *Obstructed Distance* between a and b, denoted by $D_{\text{obs,a,b}}$.

A **Spatial Alarm Query in Obstructed Space** is formally defined as follows:

Definition 1.1.1. Given the user's current location p , and an alarming distance U_d for an alarm, spatial alarm query returns the set of alarms A , where for each $a \in A, D_{obs,p,a} \leq U_d$.

We define three different type of regions: *Known Region*, *Reliable Region* and *Safe Region*

Definition 1.1.2. Known Region: We define two different known region for the POIs and the obstacles. The region containing at least 1 POI is the known region for POI.

The region circulating the POIs known region containing none or single colliding obstacles is the known region for the obstacles. The set of obstacles and POIs within this region is known to the client. We will denote the radius of the POIs known region as r_{kp} and that of the obstacles known region as r_{ko}

Definition 1.1.3. Reliable Region: We will denote the radius of the reliable region as $R_{reliable}$. Given the user's previous location P_1 and current location P_2 if, $P_1 - P_2 \leq R_{reliable}$, then no further queries to the server has to be done to compute a consistent set of answers.

Definition 1.1.4. Safe Region: The region located inside reliable region within which the answer set of POIs remains unchanged for a moving client. We will denote the radius of the safe region as R_{safe} . Given the user's previous location P_1 and current location P_2 if, $P_1 - P_2 \leq R_{safe}$, then no recalculation is needed to compute the answer.

1.2 Solution Overview

Single user trip planning query has been extensively studied

The key idea of our approach is to ...

1.3 Contributions

In summary, the contributions of the thesis are as follows:

1.4 Thesis Organization

The next chapters are organized as follows. In Chapter 2, related works are discussed. In Chapter 3 and 4, we present a naive approach and our approach, respectively, to evaluate Spatial Alarm in Obstructed Space queries. Chapter 5 shows the experimental results for our proposed algorithm. Finally, in Chapter 6, we conclude with future research directions.

Chapter 2

Related Works

In this chapter, we give a description of previous works related to this thesis. Spatial alarm processing technique in Road Network and Euclidean Space has been extensively studied in recent years. In this chapter, we first show the related works on Spatial Alarm queries in Road Networks in Section 2.1, Group nearest neighbor (GNN) queries in Obstructed Space in Section 2.2 Efficient Spatial Alarm finding techniques in Section 2.3, and finally a discussion on R-tree as an efficient data structure for indexing spatial alarms is given in Section 2.5.

Extensive research has been performed and various effective algorithms have been proposed [LYL⁺13],[ML10],[BLIY09] to process spatial alarms in Euclidean space and road network in recent years. Euclidean space considers the straight line distance between two points irrespective of obstacles on the other hand in road networks navigation is limited along predefined roads.

Obstructed space considers the shortest distance between two points in the presence of obstacles. Various spatial range query algorithms have been presented in recent times [ZPMZ04],[GYC⁺11],[SHK14] such as nearest neighbour and group nearest neighbour in obstructed space.

Again, comprehensive research [DLNV10] has been conducted to make spatial alarm evaluation energy-efficient and effective in road networks. However, to the best of our knowledge no research work has yet been published on the topic of spatial alarms in obstructed space.

2.1 Road Alarm

NN queries and k nearest neighbor (k NN) queries constitute a very important category of queries in database studies. They have many kinds of applications including but not limited to geographic information systems GIS, CAD/CAM, multimedia [CFM94], knowledge discovery [MES99] and data mining. The first problem of answering k NN queries using the R-tree was first introduced in [NRV95]. That algorithm searches the R-tree in a depth first manner, using two different metrics to help pruning intermediate nodes and leaf nodes. One metric is optimistic mindist, which corresponds to the shortest distance between the query point and the MBR (Minimum Bounding Rectangle) of a tree entry. The other is pessimistic minmaxdist, which measures the longest distance from the query point to a tree entry MBR that ensures the existence of some data point (s). For a given query point q , mindist and minmaxdist are used to order and prune R-tree node entries according to three heuristics:

- Every MBR M with mindist (q, M) greater than the actual distance from q to a given object o is discarded;
- For two MBRs M and M' , if mindist (q, M) is greater than minmaxdist (q, M'), M is pruned;
- If the distance from q to a given object o is greater than minmaxdist (q, M) for an MBR M , o is discarded.

The depth-first k NN algorithm accesses more R-tree nodes than necessary. To enable optimal index node access, a best-first algorithm was proposed in [HS99, PM97]. A memory heap is used to hold the R-tree entries to be searched, which gives priority to smaller mindist between an entry and the query point. Entries for the memory heap are selected according to the mindist between an entry and the query point. Only those entries with a small enough mindist are pushed into the heap and searched later with its sub-entries checked and pushed back if necessary. With the optimization done with the metric mindist, the best-first algorithm only accesses those nodes containing k nearest neighbors, thus achieving optimal node access.

Yu et al. [CYJ01] proposed an efficient method, called iDistance, for k nearest neighbor k NN search in a high-dimensional space. iDistance partitions the

data and selects a reference point for each partition. The data in each cluster are transformed into a single dimensional space based on their similarity with respect to a reference point. This allows the points to be indexed using a B+-tree structure and k NN search be performed using one dimensional range search. The choice of partition and reference point provides the iDistance technique with degrees of freedom most other techniques do not have. They describe how appropriate choices here can effectively adapt the index structure to the data distribution.

The majority of the research before [CSS01] were based on Euclidean distance between the points. The first contribution of this paper is that it demonstrates Euclidean metric is not good distance approximation for road network. In that case they worked with real-world dataset. And they conclude that about 40% false hits are counted when Euclidean distance is used.

Shekhar et al. [SY03] introduced In-route nearest neighbor query, called IRNN. They discussed four different techniques namely Simple Graph-Based (SGB), Recursive Spatial Range query-based (RSR), Spatial Distance Join-based (SDJ) and Precomputed Zone-based (PCZ) solution. Precomputed zone-based method always outperforms the other methods when there are no updates on the road map. However, the response time of the precomputed zone-based method dramatically increases with increases in the update ratio I/Os. It also needs more storage space for storing the Precomputed results. In the other methods, the overall response time decreases with increases in the facility density and increases with increases in the route length and the size of the road map. The spatial distance join-based method outperforms the recursive methods with fewer numbers of path computations. The experiment shows that the strategy to reduce the number of path computations to minimize the response time is reasonable. However, in the dense road map area, the performance of the recursive spatial range query-based method declines due to the increase of the false hit ratio of the Euclidean candidates and the simple graph-based method shows better response time than the spatial range query-based method.

Conventional NN search (i.e., point queries) and its variations in low and high dimensional spaces have received considerable attention (e.g. [KF96, SK98]) due to their applicability in domains such as content based retrieval and similarity search. Because of the proliferation of location based e-commerce and mobile

computing, continuous nearest neighbor CNN search promises to gain similar importance in the research applications communities. Algorithm of CNN query first introduced in [SZ01] in which the proposed solution has the false misses and the high processing cost. In [SK98] the study of the problem is done extensively and propose algorithms that avoid the pitfalls of previous ones, namely, the false misses and the high processing cost. They also propose theoretical bounds for the performance of CNN algorithms and experimentally verify that their proposed methods are nearly optimal in terms of node accesses.

However Kolahdouzan and Shahabi [KS04] introduced Voronoi-Based k Nearest Neighbor Search for Spatial Network Databases.

2.2 Energy Efficient Middle Ware Architecture for processing spatial alarms on Mobile Clients

Papadias et al. [Pap05] have proposed and solved query processing in the context of large road network by extending k NN queries. They proposed three algorithms that solve ANN queries by effectively minimizing network traversal. The first one utilizes Euclidean lower bounds and an incremental Euclidean ANN method. The other two algorithms are motivated by aggregate top- k query processing techniques. The incremental Euclidean restriction IER incrementally retrieves Euclidean aggregate nearest neighbors and computes their network distances by shortest path queries until the result cannot be improved. TA and CE explore the network around the query points until the aggregate nearest neighbors are discovered. Their techniques can be applied for various aggregate distance functions (sum and max). In addition, they can be combined with spatial access methods and shortest path materialization techniques. A thorough experimental study suggests that their relative performance depends on the problem characteristics. IER is the best algorithm when the edge weights are proportional to their lengths since, in that case, Euclidean distance becomes a quite tight lower bound of the actual network distance. Nevertheless, the performance of IER degrades fast as the weights are less reflected by the edge lengths. For such cases, threshold algorithm TA is the most appropriate method for sum queries, whereas CE concurrent expansion is the best approach for max queries. In addition, TA and CE are the only choices when the interesting

points are not indexed by R-trees or when the Euclidean distance bounds may not be used (e.g., in non spatial networks).

2.3 Group Nearest Neighbor Queries in Obstructed Space

A group nearest neighbor (GNN) query [DPM04, TH10, XL08] is slightly different form of NN query, finds the nearest data point with respect to all user locations of a particular group. A GNN query minimizes the aggregate distance for the group, where an aggregate distance is computed as the total, minimum or maximum distance of the data point from the group. Papadias et al. [DPM04] have proposed three techniques: multiple query method MQM, single point method SPM, and minimum bounding method MBM, to evaluate GNN queries.

- The multiple query method MQM is a threshold algorithm. It executes incremental NN search for each point q_i in Q , and combines their results. The distance to q_i 's current NN is kept as a threshold t_i for each q_i . The sum of all thresholds is used as the total threshold T . best dist is $\text{dist}(NN_{cur}, Q)$, where NN_{cur} is the candidate nearest neighbor found so far. Initially, best dist is set to 1 and T is set to 0. The algorithm computes the nearest neighbor for each query point incrementally, updating the thresholds and best dist until threshold T is larger than best dist.
- The single point method SPM processes a GNN query in a single traversal of the R-tree. SPM first decides the centroid q of Q , which is a point in the data space with a small or minimum value of $\text{dist}(q, Q)$. Then a depth-first k NN as the query point. During the search, some heuristics based on triangular inequality is used to prune intermediate nodes and determine the real nearest neighbors to Q .
- The minimum bounding method MBM regards Q as a whole and uses its MBR M to prune the search space in a single query, in either a depth-first or best-first manner. Two pruning heuristics involving the distance from an intermediate node N to M or query points are proposed and can be used in either manner.

If we compare among the above three techniques MQM retrieves the NN for every point in query set Q , it sometimes accesses the same tree nodes more than once for different query points, so it causes the overhead while high query set cardinality is adopted. In contrast with MQM, both SPM and MBM perform a single query with some pruning heuristics. SPM is a modified single depth-first NN search with the centroid of Q being the query point while MBM considers the MBR of Q with best-first manner. The distribution shape of query points in Q has an impact on the performance of SPM and MBM because the former approximates q_i 's centroid and uses it in the single query, and the latter takes into account the extent of Q when pruning nodes. According to the comparison conducted in [DPM04] among these three techniques, MBM performs the best as it traverses the R-tree once and takes the area covering the user's location into consideration.

Li et al. [FL11] have showed the MBM method only inspects the triangle inequality and there all other important pruning and optimization opportunities are failed to achieve. That is why they introduced a heuristic approach having query cost $O(N + M)$ according to the assumed facts. They first addressed a new exact method which has much lower query cost in compare to MBM method. They also explore high-quality approximation method that is bounded to logarithmic query cost and it has approximation ratio $\sqrt{2}$ in the worst case for a fixed dimension.

Huang et al. [HL05] have evolved two pruning strategies namely distance pruning method and MBR pruning method for GNN queries over spatial datasets indexed by the R-tree. By taking into account the distribution of all query points, we use an ellipse in both methods to approximate the query extent. Then a distance or MBR derived from the ellipse is used to prune intermediate index nodes during search. The proposed pruning strategies can be used in both the depth-first and best-first traversal paradigms. The experimental results demonstrate that their proposed methods outperform the existing ones significantly and consistently with real geographical datasets, in both page access number and CPU time.

2.4 Group Trip Planning Queries

Location-based social networks such as Facebook [fac], Google+ [goo], and Loopt [loo] have enabled a group of friends to remain connected virtually from anywhere at any time via location aware mobile devices. In this regard, GTP query problem [THK13] has been introduced for multiple sources and destination locations in Euclidean distance. In that purpose they presented two algorithms, among them one is iterative method as baseline algorithm and the other is an efficient hierarchical method. The iterative algorithm does not evaluate the trip planning query individually for every user but still have to access the same database more than once that causes more queries processing overhead. And on the other hand they evolved hierarchical algorithm that evaluates k -GTP queries with a single visit to the database and renders less processing overhead. They proposed hierarchical algorithm based on two heuristics, GTP-HA1 and GTP-HA2. Heuristic 1 and heuristic 2 are yielded to compute smaller upper bound of total distance. In terms of I/Os if the three algorithms are compared, iterative algorithm requires two orders of magnitude more I/Os than that of hierarchical algorithm. And GTP-HA1 takes on average 20% more I/Os and 27% more processing time than GTP-HA2.

In general when the users plan a trip that includes different types of data objects the number of different data points is typically restricted to 2 or 3. For example a group would like first go for shopping, taking dinner afterwards and possibly go to watch cinema later on. When a group will be planned for a large number of places and considered for flexible k -GTP queries then the query processing time may cause higher complexity. Let consider a group specify the order of visiting types $\{2,1,3\}$ then in case of flexible group they also have to compute $\{3,1,2\}, \{1,2,3\}, \{3,2,1\}, \{1,3,2\}, \{2,3,1\}$. So $m > 1$ we have to consider $m!$ visiting types for computing k -GTP queries. Thus they set the number of data types (m) to 2 and 3. They also provide the reason to consider ordered k -GTP queries because of a group wants to visit different places they know beforehand in which order they will visit the different places.

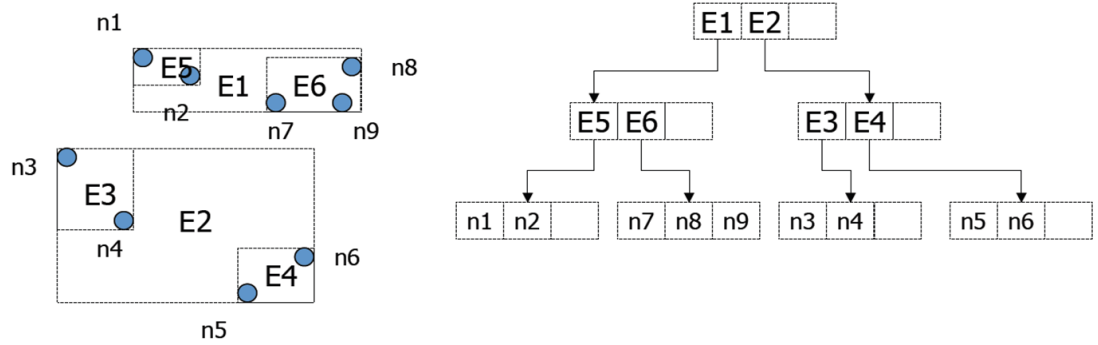


Figure 2.1: MBR corresponding to a R-tree

2.5 R-Tree: An efficient indexing Method

B-tree [PLL81] is a data structure for storing data with amortized run times for insertion and deletion in logarithmic time. It is often used for data storing on long latency I/Os (such as file systems and Data bases) because child nodes can be accessed together since they are in order. B-tree can not store new type of data (i.e. geometrical data, multi-dimensional data). So Guttman [Gut84] provided R-Tree to do that. R-tree [Sel87, Bec90] has some variants and the following properties in common-

- R-Trees can organize any-dimensional data by representing the data by a minimum bounding rectangle MBR.
- Each node bounds it's children. A node can have many objects in it.
- The leaves point to the actual objects which are stored on the disk.
- The height is always $\log n$ (It is height balanced).

Chapter 3

Our Approach

In this chapter, we propose a novel approach to evaluate k GTP queries. In a k GTP query, the group provides the source and destination locations of the group members, k , and the required categories of POIs and the query returns k sets of POIs as answer. Road networks are enormous in size with numerous POIs and the operation on all POIs brings huge processing overhead. The underlying idea of our approach is to refine the search space to reduce the number of POIs to be considered. The steps of our solution are as summarized as follows:

Step 1: We develop a heuristic technique to retrieve an initial candidate answer set that includes one POI from each category. Then we compute the aggregate distance of the retrieved POIs with respect to the source-destination pairs of the group members.

Step 2: Based on the retrieved candidate answer set in the previous step, we refine the search region using elliptical properties. Any data point outside the refined search region cannot minimize the aggregate travel distance.

Step 3: This step executes a range query to find all POIs included in the refined region. From the retrieved POIs, our approach finds the actual POIs that provide the minimum aggregate travel distance.

In Sections 3.1, 3.2, and 3.3, we discuss detail of these three steps. Section 3.4 shows our algorithms and presents detail discussion. In Section 3.5, we propose our algorithm in a way that can solve flexible k GTP queries.

3.1 Selection of Initial POIs

In this section, we present a heuristic to retrieve an initial answer set that includes one POI from every requested category. Our heuristic is based on the intuition that the selection of a path for one user affects all others in the group. We first show how our heuristic works for a k GTP query, when the visiting order of POIs is fixed.

Our developed heuristic first computes geometric centroid, c_s and c_d , with respect to the set of source and destination locations of the group members, respectively. From c_s to c_d , we apply Nearest Neighbor (NN) search greedily for the next POI from the ordered categories. This initial answer set P contains a POI from each required category. Without loss of generality, Figure 3.1 shows an example, where a GTP query needs to be evaluated from 3 categories of POIs C_1 , C_2 , and C_3 , $k = 1$ and for source and destination sets, $\{s_1, s_2, s_3\}$ and $\{d_1, d_2, d_3\}$, respectively. Our heuristic first finds nearest neighbor p_1 of c_s from C_1 , then the nearest neighbor p_2 of p_1 from C_2 , and finally, the nearest neighbor p_3 of p_2 from C_3 .

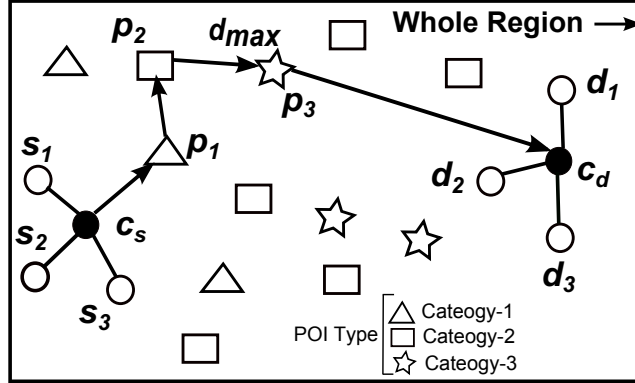


Figure 3.1: Centroid method

For each user u_i ($i = 1, 2, \dots, n$), our approach computes the trip distance T_i from s_i to d_i via p_1, p_2, \dots, p_m . Then we apply the aggregate function f to determine the aggregate trip distance as d_{best} for the initial answer set, P_{dum} . To remind the reader, for SUM, the aggregate distance is $\sum_{i=1}^n T_i$ and for MAX, the aggregate distance is $\max_{i=1}^n T_i$.

For $k > 1$, our heuristic retrieves k nearest neighbors from C_m with respect

to p_{m-1}^1 as $p_m^1, p_m^2, \dots, p_m^k$. Our approach computes aggregate travel distances for k sets of POIs $\{p_1, p_2, \dots, p_m^1\}, \{p_1, p_2, \dots, p_m^2\}, \dots, \{p_1, p_2, \dots, p_m^k\}$ and initializes d_{best}^k with the maximum of these aggregate distances.

In a flexible $kGTP$ query, the order of visiting POI category is not fixed. The group only mentions the required POI categories and is happy to visit the POIs in any order that minimizes the aggregate travel distance. In a flexible $kGTP$ query, our approach randomly selects an order of visiting POI categories and then apply the same heuristic for the ordered $kGTP$ query discussed above to retrieve the initial answer set P_{dum} for that order. After retrieving the initial POIs, our approach computes aggregate trip distances for all POI sets considering all possible orders and initializes d_{best}^k with the k^{th} minimum of these aggregate distances. For example, if the $P = \{p_1, p_2, p_3\}$ then the possible sets are $\{p_1, p_2, p_3\}, \{p_1, p_3, p_2\}, \{p_2, p_1, p_3\}, \{p_2, p_3, p_1\}, \{p_3, p_1, p_2\}$, and $\{p_3, p_2, p_1\}$.

3.2 Refinement of Search Region

We use the minimum distance d_{best}^k from step 1 to reduce the search space from the *whole search region* to a *refined search region* for both ordered and flexible $kGTP$ queries. The whole road networks with numerous data points of various categories is the *whole search region*. The search for the solution in this whole region is infeasible. We refine the search space using d_{best}^k as follows:

We consider n ellipses E_1, E_2, \dots, E_n for each user u_1, u_2, \dots, u_n . Ellipse E_i has foci at s_i and d_i , and length of the major axis equal to d_{best}^k . Let, E_{int} be the intersection region of the ellipses, i.e., $E_{int} = \cap_{i=1}^n E_i$. We confine the search only within E_{int} .

The following lemma and theorem show that the optimal answer resides within E_{int} .

LEMMA 1: $E_{int} \neq \emptyset$

Proof. Let, p be a data point in the k^{th} optimal solution found so far. For any arbitrary user u_l (where $1 \leq l \leq n$), $d(s_l, p) + d(p, d_l) < T_l$ where T_l is the trip distance of u_l in the k^{th} optimal solution. In case of sum aggregate function, $T_l < \sum_{i=1}^n T_i \leq d_{best}^k$, since d_{best}^k is the value of current best solution for SUM aggregate function. So, we find that $d(s_l, p) + d(p, d_l) < d_{best}^k$. This implies

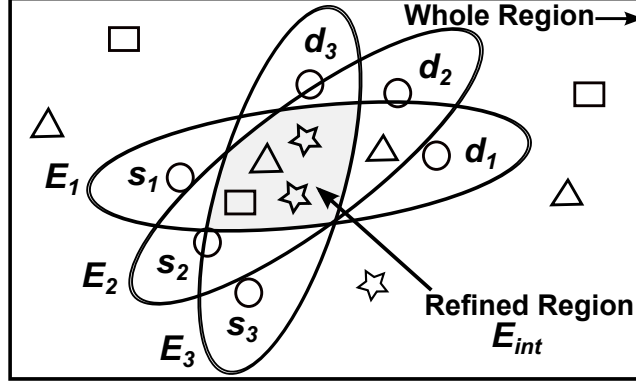


Figure 3.2: Refined search region

that point p resides inside the ellipse E_l (For any point inside an ellipse, the summation of distances from the point to the two foci is less than the major axis length). Since, u_l was chosen arbitrarily, point p must reside inside all ellipses E_1, E_2, \dots, E_n . This means the intersection region E_{int} is non-empty. $\mathcal{Q.E.D.}$

Theorem 3.2.1. *The intersection of the regions E_{int} bounded by ellipses E_1, E_2, \dots, E_n contains all the data points that constitute optimal solutions.*

Proof. (By contradiction). Let, p be a data point outside the region E_{int} and is a part of j^{th} optimal solution where $1 \leq j \leq k$. Since, p is outside E_{int} , it is outside of at least one of the ellipses E_1, E_2, \dots, E_n . Let, E_l be such an ellipse that do not contain the point p where $1 \leq l \leq n$.

Since, user u_l passes through the data point p , the trip distance T_l in the j^{th} optimal solution cannot be less than $d(s_l, p) + d(p, d_l)$, i.e., $T_l \geq d(s_l, p) + d(p, d_l)$. According to the properties of ellipses, for any point outside of an ellipse, the summation of the distances from the point to the two foci is greater than its major axis length. Hence, for ellipse E_l , $d(s_l, p) + d(p, d_l) > d_{best}^k$. But, $T_l \geq d(s_l, p) + d(p, d_l)$. So, $T_l > d_{best}^k$.

Now, consider SUM aggregate function first. Then, total group distance for j^{th} solution will be greater than d_{best}^k as shown by $\sum_{i=1}^n T_i > T_l > d_{best}^k$.

But, d_{best}^k is the current best value for k^{th} optimal solution. So, j^{th} solution value cannot be greater than d_{best}^k . (Contradiction)

In case of MAX aggregate function, the maximum distance traveled by any user will also be greater than d_{best}^k : $\max_{i=1}^n T_i > T_l > d_{best}^k$, however, d_{best}^k is the current best value of k^{th} solution. (Contradiction). $\mathcal{Q.E.D.}$

3.3 Optimal Answer Computation

In this step, our approach executes a range query on the POIs in the candidate answer set CA which resides inside the intersection of ellipses, E_{int} . The application of brute force technique to extract the solution increases the complexity with the increase of categories and users. We use d_{best}^k as current threshold for the aggregate trip distance and further prune the POIs, which makes the partial aggregate distance greater than d_{best}^k .

Specifically, we traverse starting from the source set S , pass through all categorized POIs in order and react at destination set D . If at any stage the cumulative distance greater than d_{best}^k , the traversal stops along the path. Each time a smaller cumulative distance appears by the traversal from source s_i to corresponding destination d_i through the ordered categorized POIs, we update d_{best}^k . Note that for flexible k GTP queries, we have to repeat the traversal of candidate POIs in all possible orders.

In Figure 3.3, let the predefined value for d_{best}^k is 19. Two categories C_1 and C_2 exist such that POIs $\{p'_1, p''_1, p'''_1\} \in C_1$ and $\{p'_2, p''_2, p'''_2\} \in C_2$. The traversal along path $s_1 - p'_1 - p'_2 - d_1$ is completely forsaken after visiting node p'_2 . Because at that node, the cumulative distance along that path is 20 which exceeds the range value 19. For another case, along the path $s_1 - p''_1 - p''_2 - d_1$, the cumulative distance becomes 15 at the destination point. This value is smaller than the predefined range d_{best}^k . So 15 becomes the new range value of d_{best}^k .

3.4 Algorithms

Algorithm 1 shows the details steps for processing an ordered k GTP queries. The POIs are indexed using an R -tree in the database. The source location set S , the destination set D , the category set C , the aggregate function to be minimized f and k are the inputs. The output A is k sets of POIs that have k smallest trip distances.

Algorithm 1 starts with initializing the variables A , $dist$ and d_{best}^k . Then

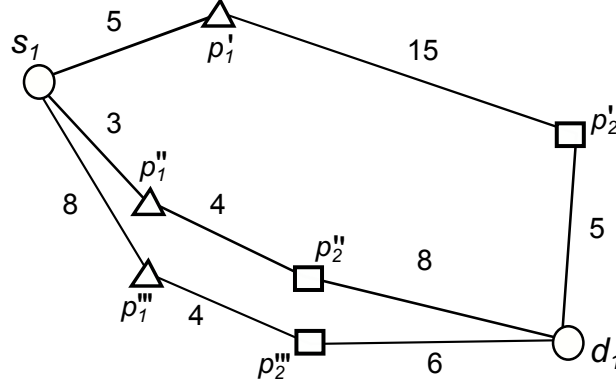


Figure 3.3: Optimal answer selection

the algorithm applies our heuristic to retrieve the initial POIs from required categories (Lines 4–11). The algorithm uses function *FindCentroid* to compute geometric centroid, c_s and c_d , with respect to the source and destination set, respectively. From c_s , the algorithm finds the greedy nearest neighbor y from category C_1 by using the function *FindNN*. The input to *FindNN* are c_s (from which the NN will be computed), C_1 (from which category the NN will be computed) and 1 (the number of requested NN). Any existing nearest neighbor algorithm in road networks can be used for this purpose. P stores the POIs computed using the proposed heuristic.

The function *ElementNum* returns the number of categories in set C . Starting from the first category, *FindNN* function returns the nearest neighbor of the next category with respect to the POI of the previous category, until the next category is C_{m-2} . All these neighbors are stored in P . Then the algorithm find k NNs from category C_m with respect to the identified POI p_{m-1} .

The algorithm uses the function *CalcBestD* to compute k^{th} aggregate distance with respect to f and assigns it to the variable d_{best}^k . *CalcBestD* takes S, D, P, k, f as inputs and computes k possible sets from POIs stored in P : $\{p_1, p_2, \dots, p_m^1\}$, $\{p_1, p_2, \dots, p_m^2\}$, \dots , $\{p_1, p_2, \dots, p_m^k\}$. *CalcBestD* computes the aggregate trip distance for each set and returns the maximum one as k^{th} aggregate distance.

CalcBestD function is described in Algorithm 2. It takes in the parameters S, D, P, k, f and produces the value of d_{best}^k as output. To compute trip distances, we apply network distance function d for each user for each values of k . For each user, the cases considered are as follows: source to POI of category-

1, POI of category-1 to ordered POIs in categories up to category- $(m-2)$, POI of category- $(m-1)$ to j^{th} POI of category- m , j^{th} POI of category- m to destination, where $j = 1, 2, \dots k$. These are all added to produce total trip distance T_i for each user in Line 11. Then, if the case is for SUM, adding the maximum trip distances for each user gives the greedy best distance, shown in Line 15. Otherwise for MAX, MAX function in Line 17 gives the maximum value of trip distances for all users and the value is assigned to d_{best}^k .

In the second step, the algorithm executes *CandidateSet*, which takes three parameters S, D, d_{best}^k as inputs and returns all POIs in the refined search region as CA . More specifically, *CandidateSet* first computes the individual ellipses, compute the intersection of ellipses and then execute a range query to retrieve the POIs within the intersection of ellipses from the database.

After getting the candidate set CA , the algorithm searches for set of POIs k optimal trips have been found. The function *RangeQf* executes a search for k trips based on d_{best}^k . *RangeQf* maintains the following characteristics: the search towards a path terminates if the cumulative distance found so far exceeds d_{best}^k value. Each time an aggregate trip distance $dist$ is smaller than d_{best}^k , d_{best}^k and A are updated.

3.5 Handling Flexible k GTP Queries

For flexible k GTP queries, as we have already mentioned in Sections 3.1 and 3.3, the algorithms need to determine paths by taking all possible orders of POI categories into consideration while computing distances d_{best}^k and $dist$. For other steps, the algorithm for flexible k GTP queries works in the similar way of ordered k GTP queries.

Algorithm 1 *kGTPQ*

Input: S, D, C, f, k Output: A

```
1:  $A \leftarrow \text{null}, P \leftarrow \text{null}, \text{dist} \leftarrow 0, d_{best}^k \leftarrow 0, i \leftarrow 1, m \leftarrow 0$ 
2:  $c_s \leftarrow \text{FindCentroid}(S)$ 
3:  $c_d \leftarrow \text{FindCentroid}(D)$ 
4:  $y \leftarrow \text{FindNN}(c_s, C_1, 1)$ 
5:  $P \leftarrow P \cup \{y\}$ 
6:  $m \leftarrow \text{ElementNum}(C)$ 
7: while  $i \leq (m-2)$  do
8:    $x \leftarrow y$ 
9:    $y \leftarrow \text{FindNN}(x, C_{i+1}, 1)$ 
10:   $P \leftarrow P \cup \{y\}$ 
11:   $i \leftarrow i+1$ 
12: end while
13:  $P \leftarrow P \cup \text{FindNN}(p_{m-1}, C_m, k)$ 
14:  $d_{best}^k \leftarrow \text{CalcBestD}(S, D, P, k, f)$ 
15:  $CA \leftarrow \text{CandidateSet}(S, D, d_{best}^k)$ 
16:  $i \leftarrow 1$ 
17: repeat
18:   for every  $p_1^i$  in  $C_1 \in CA$  do
19:      $\text{dist} \leftarrow \text{RangeQf}(S, D, CA, f, p_1^i, d_{best}^k)$ 
20:     if  $\text{dist} < d_{best}^k$  then
21:        $d_{best}^k \leftarrow \text{dist}$ 
22:        $A \leftarrow A \cup \text{RangeSet}(S, D, p_1^i, d_{best}^k)$ 
23:     end if
24:   end for
25: until All  $k$  sets are found =0
```

Algorithm 2 *CalcBestD*

Input: S, D, c_s, c_d, P, k, f Output: d_{best}^k

```
1:  $d_{best}^k \leftarrow 0, T \leftarrow null, i \leftarrow 1, j \leftarrow 1, m \leftarrow 0, n \leftarrow 0, r \leftarrow 1$ 
2:  $m \leftarrow ElementNum(P)$ 
3:  $n \leftarrow ElementNum(S)$ 
4: for  $i \leq n$  do
5:   if  $(m = 1)$  then
6:      $p_{m-1} \leftarrow s_i$ 
7:      $p_1 \leftarrow s_i$ 
8:   end if
9:    $j \leftarrow 1$ 
10:  for  $j \leq k$  do
11:     $T_i^j \leftarrow d(s_i, p_1) + \sum_{r=1}^{m-2} d(p_r, p_{r+1}) + d(p_{m-1}, p_m^j) + d(p_m^j, d_i)$ 
12:  end for
13: end for
14: if  $f$  denotes SUM then
15:   $d_{best}^k = \sum_{i=1}^n (max_{j=1}^k (T_i^j))$ 
16: else if  $f$  denotes MAX then
17:   $d_{best}^k = max_{i=1, j=1}^{i \leq n, j \leq k} (T_i^j)$ 
18: end if
    =0
```

Chapter 4

Naive Solution

To compare the efficiency of our approach, we present a straightforward solution for processing Spatial Alarms in Obstructed Space. The naive approach is a straight forward approach with no safe region computation. This approach searches for a new alarm in the known region as soon as the client changes its position.

In Section 4.1, we give an overview of the naive solution. Section 4.2 presents the algorithm of the solution. In Subsections 4.2.1, 4.2.2, 4.2.3, and 4.2.4, we discuss the steps of the algorithm in detail. Finally Section ?? proposes the improvements of this algorithm to save computational cost and client-server communication overhead.

4.1 Overview

The naive algorithm has three steps, first it computes the known region for POIs, then it computes the known region for obstacles and then it sends the visibility graph for all the POIs to the client. lastly the client computes the answer on every location change of the user.

4.2 Algorithm

Algorithm ?? shows the pseudocode for the naive approach for processing ordered k GTP queries. The input to the algorithms are the source location set S ,

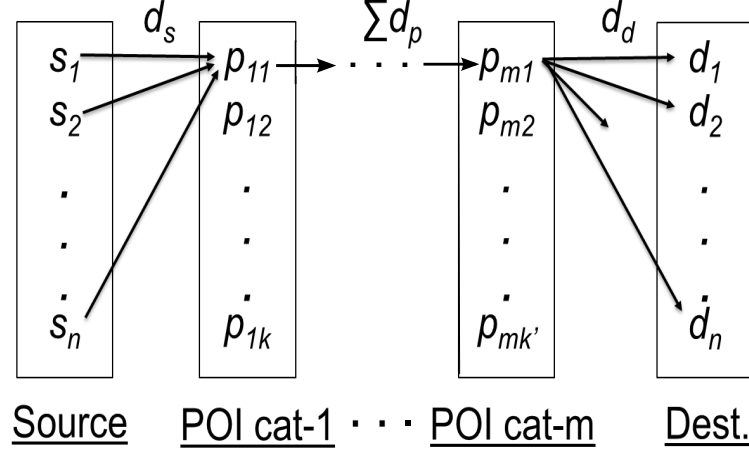


Figure 4.1: Naive approach

the destination set D , the category set C , the aggregate function to be minimized f and k . The output A is k sets of POIs that have k smallest trip distances.

We discuss the algorithm in the following four subsections.

4.2.1 r_{kp} Computation

The naive approach has two known regions, Known region for the POI and known region for the Obstacles. At first the known region for the POIs is computed. The radius of the known region is set to the alarming zone of the user upon initialization. The server, upon request of the client application, searches for POIs surrounding the user's current location. It incrementally increases the known regions radius, r_{kp} until it finds atleast one POI.

4.2.2 r_{ko} computation

After computing r_{kp} the server retrieves all the obstacles within this radius and sets the radius of the known region for obstacles, r_{ko} as r_{kp} . Then it looks for any unreachable POIs within this region. The server incrementally increases r_{ko} until all POIs are reachable.

Algorithm 3 Initialization

0: **procedure** INITIALIZATION

1: incrementally increase r until it finds atleast 1 POI

2: Make

3: Retrieve all the obstacles inside the circular region of radius $r_k p$

4: $r_k o = r_k p$

5: Construct the visibility graph V_G for the POIs

5: **while** there is any unreachable POI in the V_G and there are more than 1 collision of obstacle perimeter of the circle of radius $r_k o$ **do**

5: expand $r_k o$ until the unreachable POI gets reachable or there is less than 2 collisions

5: retrieve new obstacles;

5: If any new POI is found, then let $r_k p = r_k o$ and go to the 3rd step

5: **end while**

 Make $O =$ set of all obstacles inside the radius $r_k o$

 Return the client a response with $r_k p$, $r_k o$, P , O

5: **end procedure**=0

4.2.3 Visibility graph generation

4.2.4 Recomputation on every location change

Upon location change of the user the client application computes the obstructed distance from all the POI's in the known region and if any obstructed distance is less than the alarming distance, an alarm is invoked.

Chapter 5

Experimental Study

5.1 Experiments

In this chapter, we evaluate the performance of our proposed range based $kGTP$ query processing algorithm, $kGTPQ$, and compare it with the naive approach, $kGTPQ - NA$. We run all experiments using a computer with Intel Core i5 2.30 GHz CPU and 4GB RAM.

We have used both real and synthetic datasets to evaluate our solution. For real dataset, we have used road networks and point of interests (POIs) of California [Cal]. The road network has 21048 nodes and 21693 edges, and it contains 87635 POIs of 63 types of California. For synthetic data sets, we vary the size of the road networks and number of POIs (as shown in Table 5.2).

We have run our experiments for ordered $kGTP$ queries, which is more common in real life. We have also set the default number of category as 3, as usually a group do not plan for more than 3 categories of POIs. Note that our algorithm is applicable for any number of categories.

In our experiments, we vary the following query parameters: (i) the group size n , (ii) the number of answer set of data points k , (iii) the query area i.e., the minimum bounding rectangle covering the source and destination locations M , and (iv) the size of data sets (synthetic data set). Table 5.1 summarizes the parameter values used in our experiments. In all experiments, we estimate I/Os and the query processing time to measure the efficiency of our algorithms. In each set of experiments, we run the experiment for 100 queries and present the average result.

| Parameter | Values | Default |
|-------------------------|-------------------|---------|
| Group size | 2, 4, 8, 16 | 4 |
| Query area M | 2%, 4%, 8%, 16% | 2% |
| k | 2, 4, 8, 16 | 8 |
| Synthetic data set size | 5K, 10K, 15K, 20K | - |

Table 5.1: Values of different query parameters used in our experiments

| Nodes | Edges | POIs | Types |
|-------|--------|--------|-------|
| 5000 | 38354 | 57280 | 4 |
| 1000 | 65707 | 98424 | 4 |
| 15000 | 86567 | 129688 | 4 |
| 20000 | 107090 | 160732 | 4 |

Table 5.2: Parameters of synthetic datasets

We first present our experiment results for processing k GTP queries for aggregate function SUM in Section 5.1.1, then we present experimental results for aggregate function MAX (Section 5.1.2). Finally, in Section 5.1.3, we compare the results of aggregate functions SUM and MAX.

5.1.1 Aggregate function SUM

In this section, we present the experimental results for processing k GTP queries for aggregate function SUM. For SUM, we name our proposed approach and the naive approach as $kGTPQ_{sum}$ and $kGTPQ - NA_{sum}$, respectively. In the following sets of experiments, we vary group size, the answer set size, query area, and dataset size. When we vary one parameter, we set other parameters to default values as stated in Table 5.1.

Effect of group size n : Figure 5.1(a) and 5.1(b) show I/Os and processing time, respectively, for processing k GTP queries using $kGTPQ_{sum}$ and $kGTPQ - NA_{sum}$. We observe that for both algorithms $kGTPQ_{sum}$ and $kGTPQ - NA_{sum}$, I/Os and processing time increase with the increase of the group size n . We also observe that in terms of I/Os, our approach $kGTPQ_{sum}$ outperforms the naive approach $kGTPQ - NA_{sum}$ by 2 orders of magnitude.

Moreover, the query processing time of $kGTPQ - NA_{sum}$ is on average one and half order of magnitude higher than that of $kGTPQ_{sum}$.

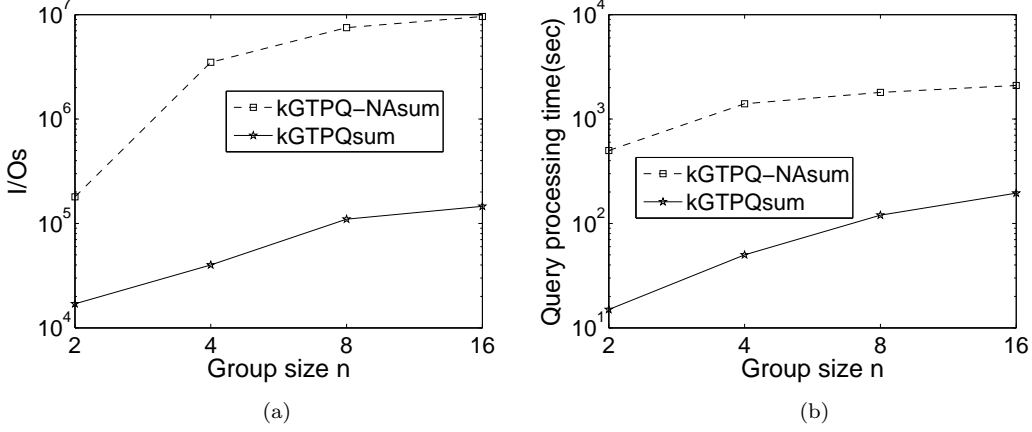


Figure 5.1: Effect of group size n for California data (a) I/Os and (b) query processing time

Effect of answer set k : In this set of experiments, we vary k as 2, 4, 8, and 16, and compare the experimental results $kGTPQ_{sum}$ and $kGTPQ - NA_{sum}$.

From Figure 5.2(a) and Figure 5.2(b), we observe that I/Os and processing time increase with the increase of k for both $kGTPQ_{sum}$ and $kGTPQ - NA_{sum}$. Figures show that $kGTPQ - NA_{sum}$ takes almost one and half orders of magnitude more I/Os compared to $kGTPQ_{sum}$. We also see from Figure 5.2(b) that the query processing time of $kGTPQ - NA_{sum}$ is on average one and half orders of magnitude higher than that of $kGTPQ_{sum}$. The superiority of our approach over the naive approach comes from the ellipse based pruning strategies that reduces the search space significantly.

Effect of Query Area M : In this case, we vary the query area M as 2%, 4%, 8% and 16% of the data space. Figure 5.3(a) and 5.3(b) show I/Os and query processing time, required by $kGTPQ_{sum}$ and $kGTPQ - NA_{sum}$, respectively. Since we need to access more data points from R -trees of different data types for a larger M , both I/Os and processing time increase with the increase of M for both approaches.

Figure 5.3(a) shows that $kGTPQ - NA_{sum}$ requires at least one and half

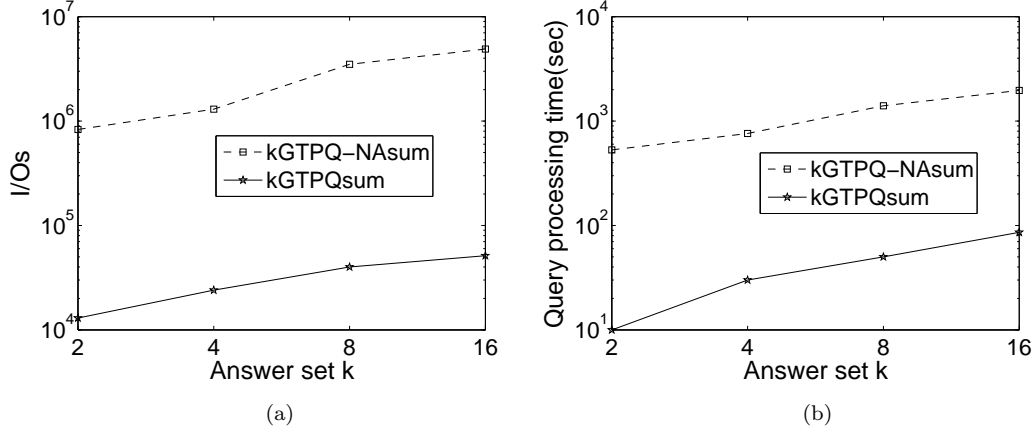


Figure 5.2: Effect of answer set k for California data (a) I/Os and (b) Query processing time

orders of magnitude more I/Os than that of $kGTPQ_{sum}$. Similarly, Figure 5.3(b) shows that the processing time of $kGTPQ - NA_{sum}$ is on average one order of magnitude higher than that of $kGTPQ_{sum}$.

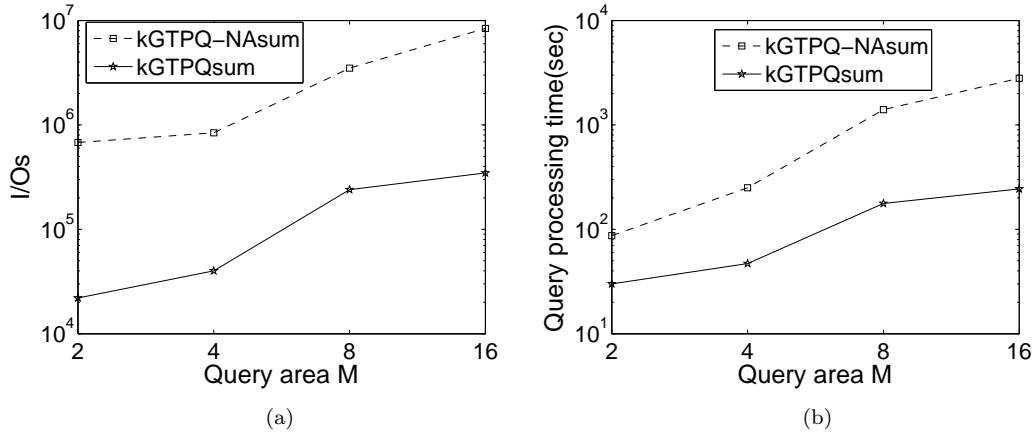


Figure 5.3: Effect of query area M for California data (a) I/Os and (b) query processing time

Effect of dataset size: In this set of experiments, we vary the road network size and number of POIs as stated in Table 5.2. In this case, we use the first column, i.e., varying the number of nodes as 5000, 10000, 15000 and 20000 to identify four synthetic datasets. Figure 5.4(a) and 5.4(b) show I/Os and processing time respectively for different synthetic dataset sizes. As expected, the experimental results show that $kGTPQ_{sum}$ outperforms $kGTPQ - NA_{sum}$

in a greater margin for a larger dataset size in terms of both I/Os and query processing time. Figure 5.4(a) shows the I/Os of $kGTPQ - NA_{sum}$ is at least on average one and half orders of magnitude higher than that of $kGTPQ_{sum}$ and from Figure 5.4(b), we see that the query processing time of $kGTPQ - NA_{sum}$ is on average one order of magnitude higher than that of $kGTPQ_{sum}$.

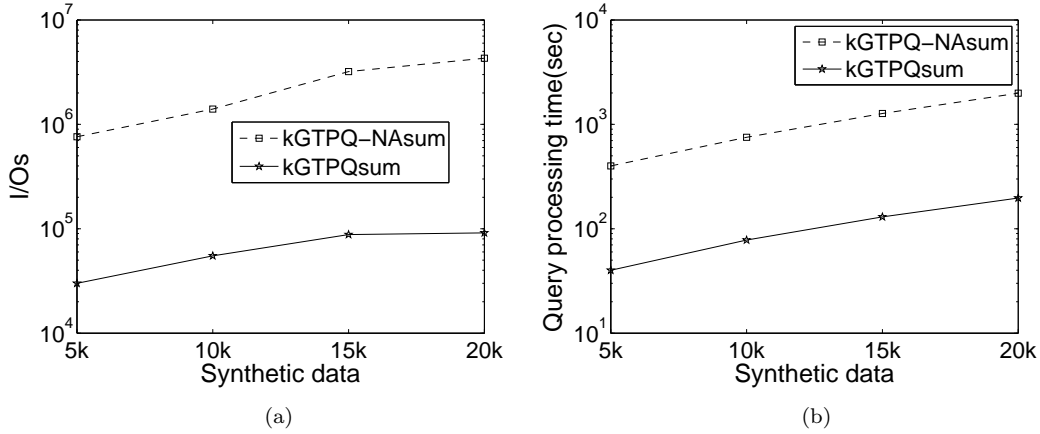


Figure 5.4: Effect of varying dataset size (a) I/Os and (b) query processing time

5.1.2 Aggregate function MAX

In this section, we present the experimental results for processing $kGTP$ queries for aggregate function MAX. For MAX, we name our proposed approach and the naive approach as $kGTPQ_{max}$ and $kGTPQ - NA_{max}$, respectively. In the following sets of experiments, we vary group size, the answer set size, query area, and dataset size, and compare the results of our approach with the naive approach.

Effect of group size n : Figure 5.5(a) and 5.5(b) show I/Os and query processing times, respectively, for processing $kGTP$ queries using $kGTPQ_{max}$ and $kGTPQ - NA_{max}$. For $kGTPQ - NA_{max}$, I/Os and query processing time almost remain constant with the increase of n . On the other hand for $kGTPQ_{max}$, I/Os and processing time slightly increase with the increase of group size from 4 to 8, and then it almost remains constant for a larger group. As expected, I/Os and query processing time of $kGTPQ_{max}$ are much less than those of $kGTPQ - NA_{max}$. Figure 5.5(a) and (b) show that I/Os and query

processing time of $kGTPQ_{max}$ are on average one and half orders of magnitude smaller than those of $kGTPQ - NA_{max}$.

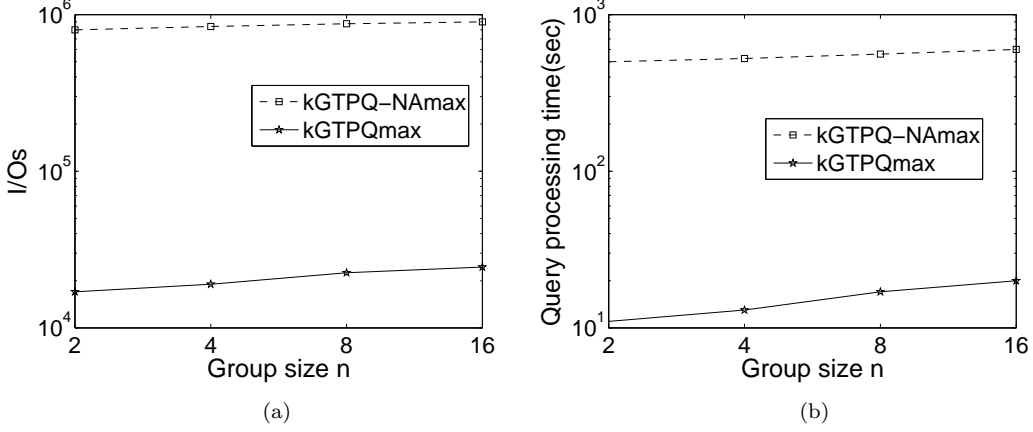


Figure 5.5: Effect of group size n for California data (a) I/Os and (b) Query processing time)

Effect of answer set k : In this set of experiments, we vary k as 2, 4, 8, and 16, and compare the experimental results $kGTPQ_{max}$ and $kGTPQ - NA_{max}$.

In Figure 5.6(a), we observe that I/Os almost remain constant with the increase of k . We find that $kGTPQ - NA_{max}$ takes almost two orders of magnitude more I/Os compare to $kGTPQ_{max}$. Figure 5.6(b) shows that the query processing time of $kGTPQ - NA_{max}$ is at least one and half order of magnitude higher than that of $kGTPQ_{max}$ because it facilitates ellipse based pruning to reduce the search space significantly.

Effect of query area M : In this set of experiments, we vary the query area M as 2%, 4%, 8%, 16% of entire data space. Figure 5.7(a) and 5.7(b) show that I/Os and processing time, respectively, for $kGTPQ_{max}$ and $kGTPQ - NA_{max}$. We observe that I/Os and query processing time increase with the increase of M for both algorithms as for a larger M both approaches need to access more data points from R-trees than those of a smaller M .

Figure 5.7(a) shows that $kGTPQ - NA_{max}$ requires at least two orders of magnitude more I/Os than that of $kGTPQ_{max}$. Similarly, Figure 5.7(b) shows that the query processing time of $kGTPQ - NA_{max}$ is on average one and half

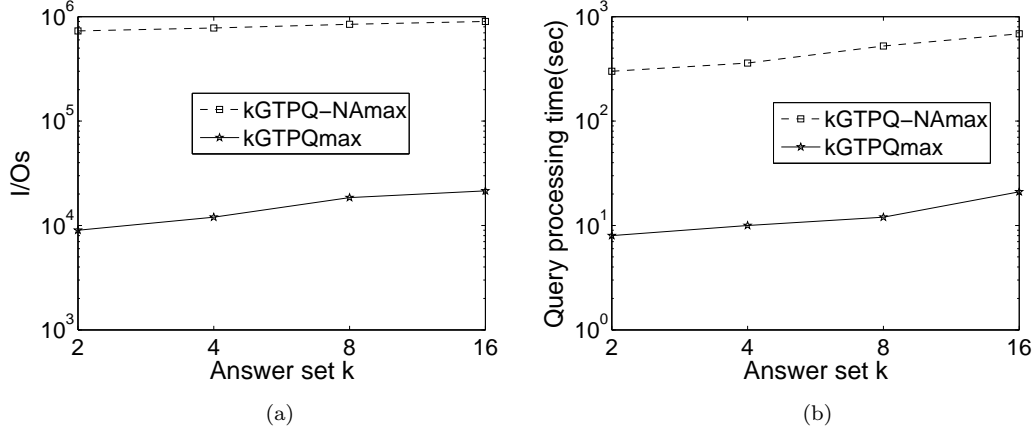


Figure 5.6: Effect of answer set k for California data (a) I/Os and (b) Query processing time

orders of magnitude higher than that of $kGTPQ_{max}$.

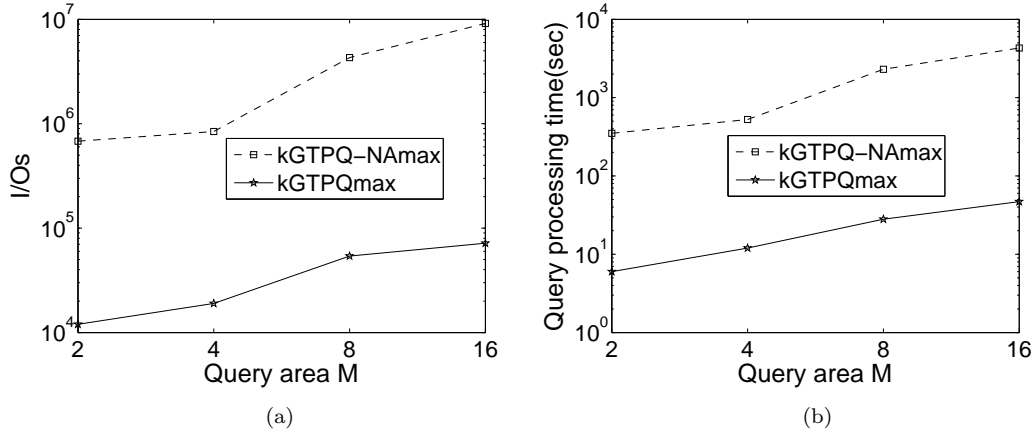


Figure 5.7: Effect of Query Area M for California data (a) I/Os and (b) Query processing time

Effect of dataset sizes: Similar to SUM, In this set of experiments, we vary the road network size and number of POIs as stated in Table 5.2.

Figure 5.8(a) and 5.8(b) show I/Os and processing time, respectively, for different dataset sizes. The experimental results show that $kGTPQ_{max}$ outperforms $kGTPQ-NA_{max}$ for all dataset sizes in terms of both I/Os and query processing time.

Figure 5.7(a) shows that I/Os of $kGTPQ-NA_{max}$ is on average one and half orders of magnitude higher than that of $kGTPQ_{max}$. Figure 5.7(b) shows

that query processing time of $kGTPQ - NA_{max}$ is on average one and half orders of magnitude higher than that of $kGTPQ_{max}$.

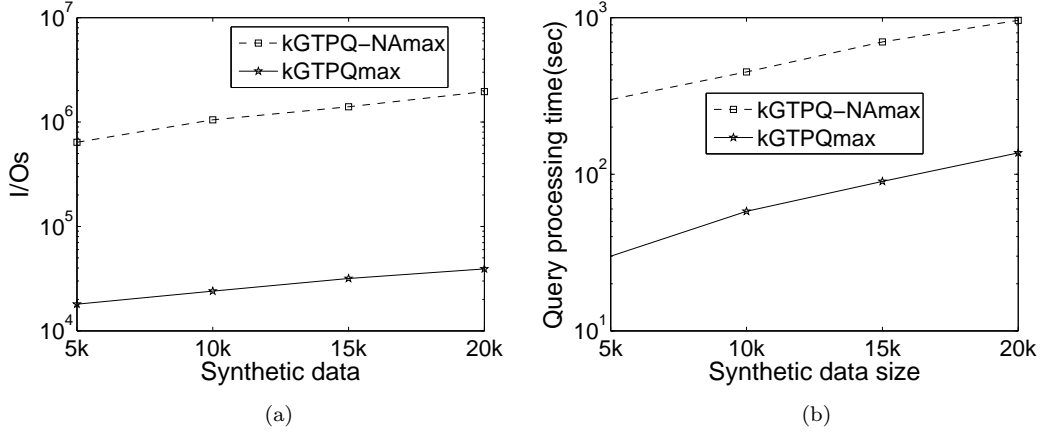


Figure 5.8: Effect of synthetic data set (a) I/Os and (b) Query processing time

| Parameter | $kGTPQ_{sum}$ | $kGTPQ_{max}$ |
|--------------------|---------------|---------------|
| Group size n | 78225 | 20737 |
| Query area M | 162250 | 39198 |
| k | 32086 | 15240 |
| Synthetic data set | 66080 | 28251 |

Table 5.3: Average I/Os for $kGTPQ_{sum}$ and $kGTPQ_{max}$ for different parameters

| Parameter | $kGTPQ_{sum}$ | $kGTPQ_{max}$ |
|--------------------|---------------|---------------|
| Group size n | 95s | 15.25s |
| Query area M | 102.13s | 23.29s |
| k | 44.03s | 12.76s |
| Synthetic data set | 111.21s | 78.37s |

Table 5.4: Average query processing time for $kGTPQ_{sum}$ and $kGTPQ_{max}$ for different parameters

5.1.3 Comparison between SUM and MAX

In this section, we compare I/Os and processing time of $kGTPQ$ queries for aggregate functions SUM and MAX. Tables 3 and 4 summarize I/Os and query processing time, respective, by varying different parameters.

We observe that $kGTPQ_{sum}$ requires much higher I/Os and processing time than $kGTPQ_{max}$. The underlying reason is as follows. $kGTPQ_{max}$ considers the maximum distance of any user while $kGTPQ_{sum}$ considers the total distance of all users. As the total distance of all users is larger than the maximum distance of any user, the area of the pruning ellipse for $kGTPQ_{sum}$ is much larger than that of $kGTPQ_{max}$. Hence, $kGTPQ_{sum}$ accesses higher number of R-tree nodes than $kGTPQ_{max}$.

We have also done the experiments for $m = 2$. We omit the experimental results for $m = 2$ as the behavior is similar to what we have described in previous sections for $m = 3$. However, the processing time and I/Os for $m = 3$ are higher than that of $m = 2$, which is expected. Since in real scenarios, the number of data point categories is typically limited to 2 or 3, the trend of increasing performance overhead with an increase of m would not effect the applicability of our algorithms.

Chapter 6

Conclusion

In this thesis, we introduced a novel approach for processing k group trip planing (k GTP) queries in road networks. To the best of our knowledge, this work is the first to address k GTP queries in road networks. We focused on both aggregate functions, SUM and MAX. We have developed an efficient pruning technique to refine the search space by exploiting elliptical properties and based on the pruning technique, we proposed an algorithm to evaluate k GTP queries. Experimental results show that our approach outperforms a naive technique with a large margin both in terms of I/Os and computational overhead.

6.1 Future Directions

Our thesis work can be extended to several interesting directions in future. Some noteworthy future directions for this research are proposed below:

- In future, we aim to work for protecting location privacy of users while processing k GTP queries.
- In this thesis, we have considered ordered point of interest (POI) types. In future we plan to work with flexible order of POI types. It will be our future challenge.

References

- [Bec90] Kriegel H.P. Schneider R. Seeger B Beckmann, N. The r^* -tree: An efficient and robust access method for points and rectangles. *In SIGMOND*, 1990.
- [BLIY09] Bhuvan Bamba, Ling Liu, Arun Iyengar, and Philip S. Yu. Distributed processing of spatial alarms: A safe region-based approach. *In 29th IEEE International Conference on Distributed Computing Systems (ICDCS 2009), 22-26 June 2009, Montreal, Québec, Canada*, pages 207–214. IEEE Computer Society, 2009.
- [Cal] Real data: <http://www.cs.fsu.edu/~lifeifei/tpq.html>.
- [CFM94] M. Ranganathan C. Faloutsos and Y. Manolopoulos. Fast subsequence matching in time-series databases. *In Proc. of ACM SIGMOD Intl Conference*, 1994.
- [CSS01] M.R. Kolahdouzan C. Shahabi and M. Sharifzadeh. A road network embedding technique for k-nearest neighbor search in moving object databases. *In Proc. ACM Intl Workshop Geographic Information Systems*, 2001.
- [CYJ01] K.-L. Tan C. Yu, B. Ooi and H. V. Jagadish. Indexing the distance: An efficient method to knn processing. *In Proc. of Very Large Data Bases Conference (VLDB)*, 2001.
- [DLNV10] Myungcheol Doo, Ling Liu, Nitya Narasimhan, and Venu Vasudevan. Efficient indexing structure for scalable processing of spatial alarms. In Divyakant Agrawal, Pusheng Zhang, Amr El Abbadi, and Mohamed F. Mokbel, editors, *18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Sys-*

- tems, *ACM-GIS 2010, November 3-5, 2010, San Jose, CA, USA, Proceedings*, pages 426–429. ACM, 2010.
- [DPM04] Y. Tao D. Papadias, Q. Shen and K. Mouratidis. Group nearest neighbor queries. *In IDME*, pages 301–312, 2004.
- [fac] Facbook: <http://www.facebook.com>.
- [FL11] Kumar P Feifei Li, Bin Yao. Group enclosing queries. *In IEEE Transactions on Knowledge and Data Engineering*, 23(10):1526–1540, 2011.
- [goo] Google+: <http://plus.google.com>.
- [Gut84] A. Guttman. A dynamic index structure for spatial searching. *In SIGMOND*, pages 47–57, 1984.
- [GYC⁺11] Yunjun Gao, Jiacheng Yang, Gang Chen, Baihua Zheng, and Chun Chen. On efficient obstructed reverse nearest neighbor query processing. In Isabel F. Cruz, Divyakant Agrawal, Christian S. Jensen, Eyal Ofek, and Egemen Tanin, editors, *19th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2011, November 1-4, 2011, Chicago, IL, USA, Proceedings*, pages 191–200. ACM, 2011.
- [HL05] Bo Huang Zhiyong Huang Hongga Li, Hua Lu. Two ellipse-based pruning methods for group nearest neighbor queries. *GIS '05 Proceedings of the 13th annual ACM international workshop on Geographic information systems*, pages 192–199, 2005.
- [HS99] G. Hjaltason and H. Samet. Distance browsing in spatial database. *ACM Trans. on Database Systems*, 24(2), 1999.
- [KF96] Faloutsos C. Siegel E Protopapas Z. Korn F., Sidiropoulos N. Fast nearest neighbor search in medical image databases. *In Proc. Very Large Data Bases Conference (VLDB)*, 1996.
- [KS04] Mohammad Kolahdouzan and Cyrus Shahabi. Voronoi-based k nearest neighbor search for spatial network databases. *VLDB conference*, 2004.

- [loo] loopt: <http://www.loopt.com>.
- [LYL⁺13] Kisung Lee, Emre Yigitoglu, Ling Liu, Binh Han, Balaji Palanisamy, and Calton Pu. Roadalarm: A spatial alarm system on road networks. In Christian S. Jensen, Christopher M. Jermaine, and Xiaofang Zhou, editors, *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pages 1372–1375. IEEE Computer Society, 2013.
- [MES99] H.-P. Kriegel M. Ester and J. Sander. Knowledge discovery in spatial databases. *Invited paper at German Conf. On Artificial Intelligence*, 1999.
- [ML10] Anand Murugappan and Ling Liu. An energy efficient middleware architecture for processing spatial alarms on mobile clients. *MONET*, 15(4):543–561, 2010.
- [NRV95] S. Kelley N. Roussopoulos and F. Vincent. Nearest neighbor queries. *In Proc. of ACM SIGMOD Intl Conference*, 1995.
- [Pap05] Tao Y. Mouratidis K. Hui C.K Papadias, D. Aggregate nearest neighbor queries in spatial databases. *In IEEE, ACM Transactions on Database Systems (TODS)*, 30(2):529–576, 2005.
- [PLL81] S. BING YAO. PHILIP L. LEHMAN. Efficient locking for concurrent operations on b-trees. *IN ACM INTERNATIONAL CONFERENCE*, 1981.
- [PM97] A. Papadopoulos and Y. Manolopoulos. Performance of nearest neighbor queries in r-trees. *In Proc. of Intl Conf. on Database Theory (ICDT)*, 1997.
- [Sel87] Roussopoulos N. Faloutsos C. Sellis, T. The r+-tree: a dynamic index for multidimensional objects. *In VLDB*, 1987.
- [SHK14] Nusrat Sultana, Tanzima Hashem, and Lars Kulik. Group nearest neighbor queries in the presence of obstacles. In Yan Huang, Markus

- Schneider, Michael Gertz, John Krumm, and Jagan Sankaranarayanan, editors, *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Dallas/Fort Worth, TX, USA, November 4-7, 2014*, pages 481–484. ACM, 2014.
- [SK98] T. Seidl and H. Kriegel. Optimal multi-step k-nearest neighbor search. *In Proc. of ACM SIGMOD Intl Conference*, 1998.
- [SY03] S. Shekhar and J.S. Yoo. In-route nearest neighbor queries: A comparison of alternative approaches. *Proc. ACM Intl Workshop Geographic Information Systems*, 2003.
- [SZ01] Roussopoulos N. Song Z. K-nearest neighbor search for moving query point. *SSTD*, 2001.
- [TH10] Rui Zhang Tanzima Hashem, Lars Kulik. Privacy preserving group nearest neighbor queries. *In ACM SIG-SPATIAL*, 2010.
- [THK13] Mohammed Eunus Ali Tanzima Hashem, Tahrima Hashem and Lars Kulik. Group trip planning queries in spatial databases. *In SSTD’13*, pages 259–276, 2013.
- [XL08] Lei Chen Xiang Lian. Probabilistic group nearest neighbor queries in uncertain databases. *In IEEE*, 20(6), 2008.
- [ZPMZ04] Jun Zhang, Dimitris Papadias, Kyriakos Mouratidis, and Manli Zhu. Spatial queries in the presence of obstacles. In Elisa Bertino, Stavros Christodoulakis, Dimitris Plexousakis, Vassilis Christophides, Manolis Koubarakis, Klemens Böhm, and Elena Ferrari, editors, *Advances in Database Technology - EDBT 2004, 9th International Conference on Extending Database Technology, Heraklion, Crete, Greece, March 14-18, 2004, Proceedings*, volume 2992 of *Lecture Notes in Computer Science*, pages 366–384. Springer, 2004.