Progressive Supervising of Distance Based Range Queries
With Minimal I/O Cost

A. V. Seetha Lakshmi, Dr. S. P. Victor

1Dept. of IT, G.T.N Arts College, Dindigul, Tamil Nadu, India
2Dept. of Computer Science, St.Xavier’s College (Autonomous), Palayamkottai, Tamil Nadu, India

Abstract
We focus on the distance based range queries that continuously change their locations in a Euclidean space. We present an efficient and effective monitoring technique based on the concept of a safe zone. Given a positive value r, a distance based range query returns the objects that lie within the distance r of the query location. The safe zone of a query is the area with a property that while the query remains inside it, the results of the query remain unchanged. Hence, the query does not need to be re-evaluated unless it leaves the safe zone. We propose a technique based on powerful pruning rules and a unique access order which efficiently computes the safe zone and minimizes the I/O cost.

Keywords
Safe Zone, Range Queries, Spatial Data, Continuous Queries

I. Introduction
We consider a set O of objects, a query point q and a positive value r. We use dist(o; q) to denote the distance between an object o 2 O and the query q. A distance based range query returns every object o 2 O that lies within distance r of the query location, q, i.e., every object such that dist(o; q) r. Our main focus in this paper is on Euclidean distance based range queries. Since the search space around the query is a circle in this case, such queries are also called circular range queries. We also consider the case when dist(o; q) is the network distance between o and q (e.g., queries moving in a road network).

Another variation of the range query, which we term “rectangular range query” (also called window query), returns the objects that lie within a rectangle around the query location. Distance based range queries and rectangular range queries are inherently different and have different applications. When clear by context, we use the term range query to refer to the distance based range queries. Due to availability of inexpensive position locators, cheap network bandwidth and mobile devices with computation and storage capabilities, location based services are gaining increasing popularity. Consequently, continuous monitoring of spatial queries has received significant research attention in the past few years [1-8].

In this paper, we study the continuous monitoring of moving range queries over static data objects, i.e., a scenario where the queries are constantly moving whereas the data objects do not change their locations. Such scenario has many interesting applications. Consider the example of a family travelling by car. Suppose they need to reach their final destination by a certain time and only have up to 90min available for lunch. They may want to continuously monitor restaurants within 10km of their current location so that they can choose a restaurant that serves their favorite meals, and will not take more than 15 min to reach. As another example, a bomber plane might want to continuously monitor the enemy targets (e.g., airport, arms depot) that are within its attack range.

We next discuss two models to monitor spatial queries. Client-server model. In this model, the clients issue queries and the central server is responsible for the computation of these queries. For example, a person walking down the street may issue a query to his mobile service provider to continuously report the coffee shops within 1km of the issuer’s location. It may be assumed that the server processes the query in the main-memory, i.e., the data objects are stored in the main-memory along with other relevant information needed to efficiently update the results. However, such systems require that the server continuously maintains this information in the main-memory in order to provide the service.

- The computation of the safe zone reduces the overall computation time because the query needs to be re-evaluated only when it leaves the safe zone. Our experiments indicate that the cost of computing the safe zone is small compared to the cost of the range query.
- Although the shape of the safe zone may be arbitrarily complex, we can still efficiently check whether the query lies within it. If the query is based on network distance, the safe zone itself is a small network that is a subset of the original network and it has to be determined whether the query lies within the safe zone or not. For the circular range queries, we utilize the fact that the safe zone only depends on the so-called guard objects. Checking whether the query lies within the safe zone takes k distance computations, where k is the number of guard objects. Our experimental results demonstrate that the average number of guard objects is around 5. This makes our proposed approach applicable for the clients that have limited computational power. We also present a theoretical analysis and give an upper bound on the expected number of guard objects for the queries with the diameter of the safe zone no more than a constant times its expected value.
- We do not require the data objects to be stored in the main-memory, which allows our approach to work on systems with limited main-memory (e.g., GPS navigation systems).
- When an update request is received, the server computes the new safe zone and the results for the circular range queries. After updating the results, the server only sends new information to the clients. For example, if the client was informed that an object oi is within its range, the object oi is not sent again in updated results if it still lies within the range. If in the future such object oi ceases to be within the range, the client is informed that oi is out of the range. Our experimental results demonstrate that this significantly reduces the amount of data transmitted from the server to the clients.
- In the client-server paradigm, our proposed approach does not require the server to maintain or record any information related to the queries, yet it efficiently updates the safe zones. This enables the server to run this service on-demand.

Note that some computation models require queries to get registered at the server and report their locations after every t time units. Our approach can be readily applied to such systems. In the rest of the paper, we assume a model where a query contacts the server only if it leaves the safe zone. Although there exists a safe zone based solution for moving window queries [6], this technique is not applicable to the moving model.
circular range queries. In Section 2 we show that it is not possible to extend this technique to the case of the distance based range queries as the problems of monitoring moving window queries and the distance based range queries are inherently different. We apply an aggressive approach to prune the objects/entries that cannot affect the results and/or the safe zone. Our pruning rules are tight and the performance of our solution is close to optimal.

II. Related work

A. Spatial Queries in Euclidean Space

Continuous monitoring of spatial queries has been extensively studied in the recent past [2-3, 6, 9-13]. Prabhakar et al. [14] proposed velocity constrained indexing and query indexing for continuous evaluation of static queries over moving objects. Mokbel et al. [15] introduced an algorithm (SINA) for evaluating a set of concurrent spatial queries, which reduces the overall cost by shared execution and incremental evaluation. Several distributed processing techniques to continuously monitor range queries have also been proposed [7, 16-17]. Gedik et al. [7] introduce a technique called MobiEyes, which reduces the computation load on the server and communication costs between the clients and the server by delegating some computation load to the client objects (e.g., mobile devices). In [18], the authors propose a motion adaptive indexing scheme that uses the concept of motion sensitive bounding boxes to model moving objects and queries. Hu et al. [2] propose a generic framework to monitor continuous range queries and kNN queries over moving objects. They define the safe zones for each object such that the query results remain unchanged if the object does not leave the region. However, their approach is not designed for moving queries. Wu et al. [19] use a new query indexing method called CES-based indexing to minimize the total query evaluation time.

We now present the related techniques that are specifically designed for moving spatial queries. Several techniques have been proposed to construct safe zones for moving kNN queries [6, 20-23] and moving window queries [6]. However, to the best of our knowledge, there does not exist any safe zone based technique to continuously monitor moving circular range queries. We next show that the existing work cannot be extended to monitor moving circular range queries continuously.

Tao et al. [8] introduce Time-Parameterized queries (TP queries). A TP query assumes that the motion pattern (e.g., path and speed) of the query is known and retrieves the current results along with a future time at which the current results will become invalid. A TP query also reports the object that invalidates the results. In [8], the techniques to answer TP kNN queries, TP window queries and TP join queries are presented.

Fig. 1 shows an example of a window query where the Current location of the query is q and its window is shown with a solid line (the search space is shown in a dark shade). The current result of the window query q is the object o1. A TP window query is issued to find the object that invalidates the current result when the query is moving in the direction shown by the arrow. The query returns the object o2 as it invalidates the current result when the query reaches the location q. In other words, when the query reaches q, it has objects o1 and o2 within its window and not only o1. The minimal area searched by the TP query is shown shaded in fig. 1.

Based on TP queries, Zhang et al. [6] present a solution to continuously monitor kNN queries and the window queries. They use TP queries to identify the safe zones for moving queries. The algorithm starts by assuming that the whole space is the safe zone. TP queries are then issued towards the corners of the current safe zone. If a TP query retrieves an object that has not already been considered, the safe zone is trimmed using that object (for details, see [6]); otherwise, the corner is marked as confirmed. The algorithm terminates when all the corners are confirmed.

We note that there does not exist any reported work on TP circular range queries and the technique presented in [6] cannot be applied to such queries. Even if the technique to answer TP window queries are extended to answer the TP circular range queries, the TP circular range queries cannot be used to construct the safe zone. The reason is as follows. The key observation used in the technique presented in [6] is that if none of the TP queries issued towards corners of a region returns a new object, the region is guaranteed to be the safe zone. This observation does not hold for the moving circular range queries. Consider the example in Fig. 2 where the current region is shown dark shaded. The TP range queries are issued towards each of the two corners A and B and they search the space shown shaded in the figure. No object is returned by either of the TP range queries. However, the region cannot be guaranteed to be the safe zone. Consider that the query moves to the location q. Then the object o2 lies within its range, which invalidates the results.

Fig. 2: TP Circular Queries Cannot be Used to Construct Safe Zone

B. Spatial Queries in Road Networks

Significant research attention has been given to developing techniques for spatial queries in road networks. kNN queries [24-31] and range queries [28, 32-34] are among the most studied spatial queries in road networks. Chen et al. [35] study the path k-NN queries that returns kNNs with respect to the shortest path connecting the destination and the user’s current location. Papadias et al. [28] propose a framework to support nearest neighbor queries, closest pairs queries, range queries and distance joins on a road network. However, they assume that the queries...
and the objects have fixed positions in the spatial network. Wang et al. [34] propose a solution to answer static range queries over moving objects. They utilize a disk resident R-tree to store the network and a grid structure to store the positions of moving objects. The main idea is to first find the edges that may contain the objects within the range and then the grid cells that overlap with the edges are used to retrieve the objects. Liu et al. [33] present a distributed processing technique to solve the moving range queries over moving objects. Their approach relies on the computation power of the moving objects and each moving object reports to the server when it affects the results of one or more queries. Stojanovic et al. [32] propose technique for continuous monitoring of range queries over moving objects. The range of the query may be defined by a user selected area, a map window, a polygon, a circle or a part of the road segment.

III. Framework

A. Solution Overview

Consider the example in Fig. 3 where a range query q is shown. Its range is r and the area within its range is shown shaded. Some objects around it are also shown. The objects that lie within the range form the result set and are called internal objects (e.g., the objects o1 and o2). The objects that do not lie within the range are called external objects (e.g., the object o3). Let $C_i$ be a circle of radius r with centre at the location of the object $o_i$. Fig. 3 shows the circles for the objects $o_1$, $o_2$ and $o_3$. The objects that do not intersect the current safe zone and consequently do not affect its shape. For this reason, the final safe zone can be defined without using the circles of $o_4$ and $o_5$. In this paper, the objects that contribute to the shape of the final safe zone are called guard objects (e.g., $o_1$, $o_2$ and $o_3$). An internal (external) object that contributes to the final safe zone is called an internal (external) guard. Internal guards in this example are $o_1$ and $o_2$ whereas $o_3$ is an external guard. For the sake of simplicity, in what follows we refer to both “current safe zone” and “final safe zone” simple as “safe zone”.

1. Data Structure at a Glance

All objects are indexed by a disk-resident R-Tree [30]. For each query, the server keeps the following information in its memory during the computation of the safe zone: 1) its location; 2) the list of internal objects called answer list; 3) the list of guard objects. For each guard object, the server stores its arcs that contribute to the safe zone. In the example in Fig. 3, the object $o_1$ has an arc with two end vertices $v_1$ and $v_3$. We use this arc (or vertices) for effective pruning. Note that the server stores this information in its memory only during the construction of the safe zone, and discards this information after the safe zone has been computed and sent to the client.

2. Checking Whether q Lies in the Safe Zone

Since the clients that issue queries (e.g., mobile devices) have limited computational power, it is desirable that checking whether the client is inside the safe zone is not computationally expensive. Although the shape of a safe zone may be complex, the cost of checking whether q lies in the safe zone takes only k distance computations where k is the number of guard objects. More specifically, the query q computes its distance from each of the guard object. If it lies within the circle of every internal guard and lies outside the circle of every external guard then it lies within the safe zone. Our experimental results show that the average number of guard objects is around 5. We also present a theoretical analysis to give an upper bound on the expected number of guard objects for the queries that satisfy certain constraints. A simple approach to compute the safe zone is to consider all objects and find the objects that actually contribute to the safe zone. However, the number of objects that are considered must be reduced in order to reduce the I/O cost and to improve the
B. Pruning Rules

As shown in the example in fig. 4, some objects do not affect the safe zone. More specifically, if the circle of an object contains the safe zone (such as o4 in fig. 4) or lies completely outside the safe zone (such as o5 in fig. 4), that object does not affect the shape of the safe zone. In this section, we present some effective pruning rules to prune such objects. Note that only the circles of internal objects may contain the safe zone and only the circles of external objects may completely lie outside the safe zone. Hence, some pruning rules are specific to the internal objects and some are to be applied only on external objects.

First, we present pruning rules based on the approximation of the safe zone by a rectangle. Let a and b be two rectangles or points; we use mindist(a; b) and maxdist(a; b) to denote the minimum and maximum distances between them, respectively.

1. Using Approximation of the Safe Zone

Let RS be the minimum bounding rectangle of the current safe zone as shown in Fig 5. Let Rcnd be a rectangle that contains some candidate objects.

PRUNING RULE 1: If maxdist(Rcnd; RS) < r then no object in Rcnd can affect the safe zone.

Proof: Let o be an object in Rcnd. For every point p 2 RS, dist(o; p) < r because maxdist(Rcnd; RS) < r. Hence, the circle of o contains every point p of the safe zone, i.e., o does not affect the safe zone.

PRUNING RULE 2: If mindist(Rcnd; RS) > r then no object in Rcnd can affect the safe zone.

Proof: Let o be an object in Rcnd. For every point p 2 RS, dist(o; p) > r because mindist(Rcnd; RS) > r. Hence, the circle of o does not contain any point p of the safe zone, i.e., o does not affect the safe zone.

Fig. 5: Pruning using the Approximation of Safe Zone

In the example of Fig. 5, where maxdist(R2; RS) < r, it can be immediately verified that any object in R2 contains the safe zone in its circle. Similarly, mindist(R1; RS) > r and every object in R1 can also be pruned.

2. Using the Guard Objects

Although the rectangle based pruning is inexpensive, it is unfortunately not very tight. We present tighter pruning rules below, based on the positions of the guard objects.

PRUNING RULE 3: If mindist(Rcnd; oi) > 2r for any internal guard object oi then no object in Rcnd can affect the safe zone.

Proof: An object can only affect the safe zone if its circle intersects the safe zone. Safe zone is the area defined by the intersection of the circles of the internal guard objects minus the circles of the external guard objects. Hence, the circle of any internal guard object contains the whole safe zone. Thus a circle can only intersect the safe zone if it intersects the circles of all internal guard objects. Consequently, if an object oj lies at a distance greater than 2r from any internal guard object, it cannot intersect the safe zone.

IV. Algorithm

We use an R-Tree [30] to index the objects. Each leaf and index node of an R-Tree contains pointers to its entries and a minimum bounding rectangle that contains all its objects. For details, please see [30].

Algorithm 1 outlines the solution. A min-heap is initialized with the root entry of the R-Tree. The entries are de-heaped iteratively until the heap becomes empty. If a de-heaped entry e has maxdist(e; q) < r, then all the objects in it are internal and we apply pruning rules 1 and 4. If the entry is pruned, we do not need to check any objects within it for the construction of the safe zone. However, as these objects are internal, they contribute to the answer to be sent to the query. Therefore, we insert all the objects that are within this entry to the answer list (lines 4 - 7).

Range Query (q; r)

Input: q: the query point; r: range of the query

Description:
1. initialize a min-heap H with root of the R-Tree
2. while H is not empty do
3. deheap an entry e
4. if maxdist(e; q) < r then
5. if pruned using rules 1 and 4 then
6. insert all objects of e in the answer list
7. continue
8. else if mindist(e; q) > r) then
9. Pruning used rules 2, 3 and 5, continue;
10. if e is an object then
11. TrimSafeZone(e,q,S) /* Algorithm 2 */
12. if e is a leaf or index node then
13. for each entry c in e do
14. insert c into H with key set to its minimum distance from boundary
15. send guard objects and answer list to the query q

If the de-heaped entry e has maxdist(e; q) > r, then all the objects in it are external objects and we apply pruning rules 2, 3 and 5 (lines 8 and 9). If the entry is pruned, we continue the algorithm by de-heaping the next entry. Note that an entry e for which mindist(e; q) < r cannot be pruned by any of the pruning rules. This is because such entries may contain both internal and external objects, while all the proposed pruning rules are applicable either to internal objects or to external objects. For this reason, we do not consider such entries for pruning.

If e is an object and cannot be pruned, we use it to trim the safe zone; if it is an internal object, we also insert it into the answer list (lines 10 - 12). Otherwise, if e is a leaf or index node, we insert its entries into the heap with key of each entry set to minimum distance of the entry from the boundary of the range query (lines 13 - 15). The algorithm stops when the heap becomes empty.
V. Conclusion
In this paper, we presented a safe zone based approach to efficiently monitor distance based range queries in Euclidean space and in road networks. We conducted a rigorous theoretical analysis to study the effectiveness of our safe zone based approach for Euclidean distance based range queries. The experiment results also demonstrated that the proposed approach for Euclidean distance based range queries is close to optimal. We also showed that our network distance based algorithm is an order of magnitude faster than a naive approach.

References

Mrs. A.V. Seetha Lakshmi received her B.Sc., degree from S.R.C college Trichy, which is affiliated to Bharathidhasan University in 1993. She received her MCA degree from Madurai Kamaraj University in 2005. She received her M.Phil degree from Madurai Kamaraj University in 2007. She was working as assistant professor in Mangayarkarsi college from 2005 to 2007. At present she is working in G.T.N Arts college, Dindigul from 2007 as Assistant Professor. She has registered her Ph.D. in M.S. University, Tirunelveli, and her research work is progressively going on in the area of Data Mining.