Preserving User Location Privacy with Safe Zone for Spatial Queries

[1]Sakthi Priya, [2]Dinesh Babu, [3]Karthiga

[1, 2, 3] Department of Computer Science and Engineering, Pondicherry Engineering College
Pondicherry, India

Abstract

Now a days the use of Location Based Services (LBS) gained importance among the users. In most of the cases, users do not want to reveal their personal information to their service providers. Sometimes mobile devices are out of the specified range which cannot be processes by servers. Hence confidentiality, privacy and up to date and optimal answers will be important aspects in location based services. In this paper we extend two features called safe zone and anonymizer to overcome the problems faced by LBS now.

Keywords: *Spatial Objects, anonymizer, safe zone, database management.*

1. Introduction

With the extensive use of GPS devices, more and more people are using location based services. Various applications like digital battle field, highway patrol, traffic control rely on LBS. These applications are increasing day by day require query processing which should be done quickly and effectively. For example, when a moving user wants to go to nearby restaurant and issue a query, the call center may locate all the nearby restaurants and dispatch them to user. Since the user is moving, he/she requires immediate reply. After sometime the call center gives answers based on the location from which user gave initial query. But at that time user may be in another location but he cannot access the given answers. To prevent this we propose safe zone based on user's location over a range and an anonymizer. The anonymizer should not reveal the personal information to LBS. All the processing are done by transformed spatial queries. In this same situation, we have taken into account of quality attributes for queries and safe zone. For eg: in this, the quality attributes are price, star, etc of restaurants.

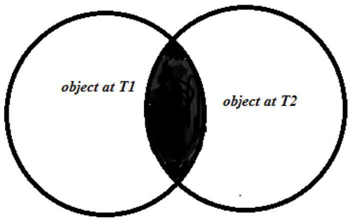


Fig. 1 Object at different time

In Fig 1 an object O is at one position at time T_1 and the same object is at another position at time T_2. The circles represent safe zones. The intersected part will be pruned from further search.

2. Related work

2.1 Location Selection Queries

In the past, different constraints have been combined with conventional spatial queries to select semantically optimal locations or objects. Du et al. [1] proposed the optimal location query. Involving a site set S, a weighted object set O, and a spatial region Q, an optimal-location query returns a location in Q with maximum influence. The influence of a location l is the total weights of objects in O, each of which has l as its nearest neighbor in S. In other words, the influence of a location is the sum of weights of all its reversed nearest neighbors (RNNs).

Yiu et al. [2] formalized the top-k spatial preference query, which returns the k spatial objects with the highest ranking scores. Objects are ranked based on an aggregate score function that is defined for the feature qualities in their spatial proximity. Such score functions, however, do not support multidimensional dominance relationship. Therefore, the top-k spatial preference query is essentially different from our FDL query.

Continuous monitoring of spatial queries has been extensively studied in recent past . Prabhakar et al. [3] proposed velocity constrained indexing and query indexing for continuous evaluation of static queries over moving objects. Mokbel et al. [4] introduced an algorithm (SINA) for evaluating a set of concurrent spatial queries, which reduces the overall cost by shared execution and incremental evaluation.

Gedik et al. [5] 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).

3. System architecture

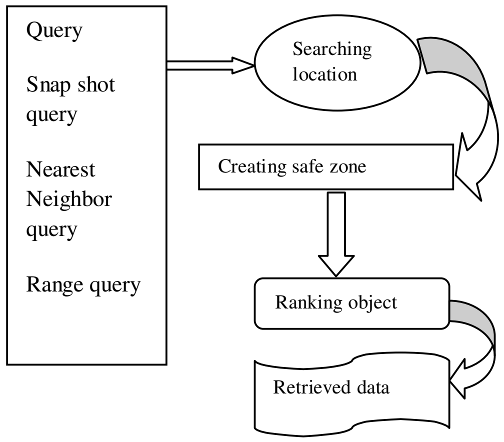


Fig. 2 System Architecture

In Fig 2, the initial stage for the system architecture is giving query. Here, snap shot query, nearest neighbor query, range query will be taken. After sending the query to server it mainly searches for location. Then it creates one safe zone for the object to retrieve the result. 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. The shape of the safe zone is defined by the so-called guard objects. After getting the result server rank location and send to querying user.

4. Framework

In this section, we first give a solution overview and introduce the terms and notations used in this paper.

4.1 Solution Overview

In fig 3, the first phase of proposed system is shown. At first the user sends a spatial query via anonymizer. It transforms the respective query and gives it to Location Based Server(LBS). The LBS processes query and stores it. Database stores the user details.

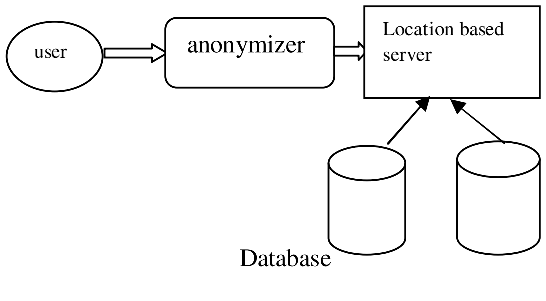


Fig. 3 Phase I of system

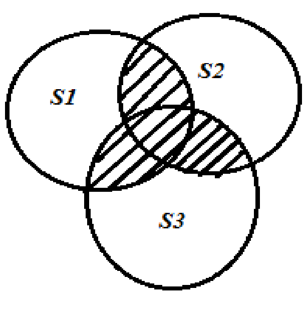


Fig. 4 Safe Zone

Since we consider a moving query, the user changes his/her position frequently. So we create safe zone for certain range and process query. The safe zone is periodically created at same time intervals. At first, LBS processes the spatial query and gives the result to user by utilizing safe zone. The results will not be changed within the zone. Once the user gets result, his/her mobile device sends one reply to LBS. After certain time user moves and will be in different position. Then another safe zone is created and same process is done. The shaded part in Fig 4 represents intersected part of three safe zones. The LBS do not process those regions because they are already searched which leads to decrease in time.

If the LBS do not receive any reply from mobile device, it stores the result. Sometimes, mobile devices are out of range and could not reply to servers and sends reply once it comes within range. At that instant, LBS look whether the device within that safe zone or not. If it is in the safe zone results will not be changed.

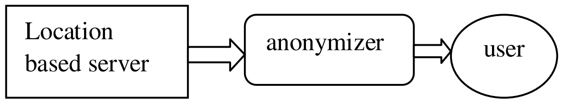


Fig. 5 Phase II of system

Fig 5 shows phase II of system. After processing spatial queries like range or nearest neighbor LBS send back results to user via anonymizer. In this whole process, any of user's information will not be revealed to LBS. This assures privacy and confidentiality.

4.2 Framework and example for spatial K-anonymity

In this framework (fig 6), a user sends his location and query to the anonymizer through a secure connection. The anonymizer removes the id and personnel information of the user and transforms his location through a technique called *cloaking*. Cloaking hides the actual location by a *K-anonymizing spatial region* (*K*-ASR or ASR), which is an area that encloses the client that issued the query, as well as at least *K*-1 other users. The anonymizer then sends the ASR to the LBS, which returns to the anonymizer a set of *candidate results* that satisfy the query condition for any possible point in the ASR. The LBS may be compromised, i.e., an adversary may have complete knowledge of all queries received by the LBS.

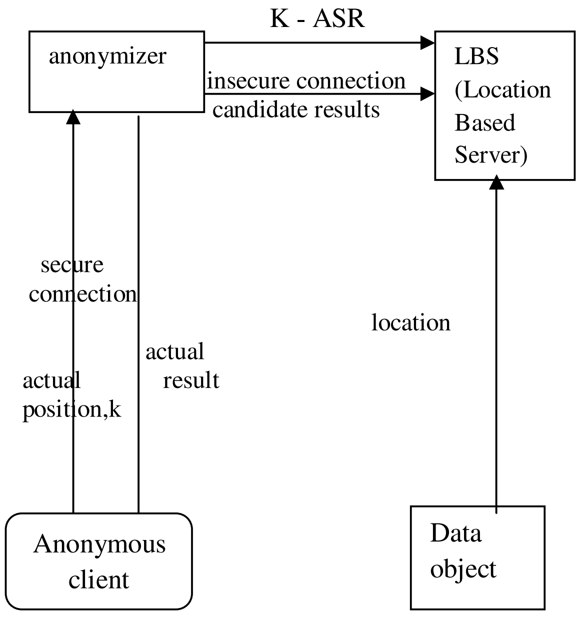


Fig. 6 General Framework

5. Safe zone construction

5.1 Algorithm outlines the solution (Safe Zone)

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.

Algorithm 1 Range Query (q, r)

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

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

Algorithm 2 TrimSafeZone (o, q, S)

Input: *o*: an object *o* to be used for updating the safe zone;
q: the query point; *S*: the list of current guard objects;
Description:

```
        for each guard object oi in S do
            for each intersection point vi of circles of o and oi do
                add vi to vertices list if vi lies on the boundary of the safe zone
        add o to the list of guard objects S
        if o is an internal object then
            remove every vertex v if dist(o, v) > r
        else if o is an external object then
            remove every vertex v if dist(o, v) < r
        remove every guard object o from S if all its related
        vertices have been removed.
```

5.2 Algorithm The calculation of safe region for range query

The purpose of this algorithm is to calculate the safe region radius and decide the query result set. The safe region radius *r* is returned for the object *o* under query *q*. The Boolean return value of *true* or *false* indicates whether object *o* is within the query result set.

```
SafeRegion(q, o, & r)
{
if (q is a rectangular range query) then
{
//Three circumstances exist,
//A: o.p is inside query region I
//B: o.p is inside query region II
//C: o.p is inside query region III
if (o.p is inside query region I) then
{
        r = distance from o.p to the closest edge of the rectangular
query region;
return true;
}
        else if (o.p is inside query region II) then
        r = distance from o.p to the closest edge of the rectangular
query region;
        else // o.p is inside query region III
        r = distance from o.p to the closest vertex of the
rectangular query region;
return false;
}
        else if (q is a circular range query) then
{
//Two circumstances exist,
//A: object o is inside the
//circular query region
//B: object o is outside of the
//circular query region
doq= dist(o.p, q.circle.p);
//dist(a, b) represents the distance
//between point a and point b. doq is the
 //distance between object o and query q.
r = ABS(doq - q.circle.r);
if (doq <= q.circle.r) then
return true;
        else
return false;
}
}
```

5.3 Safe zone techniques

Initially, the whole space is assumed to be the safe zone. We then access each object that cannot be pruned, and use its circle to trim the safe zone. The algorithm stops when all the objects that cannot be pruned are accessed. The order in which the objects are accessed is important as better access order retrieves fewer objects that affect the safe zone. We first present our proposed access order. Secondly, we present our query processing algorithm followed by the algorithm to trim the safe zone. Finally, we present an efficient technique to update the safe zone when the query leaves it.

5.4 Access order

After applying the pruning rules presented above, there may be several objects left in the unpruned area. The order in which these objects are accessed is important. Intuitively, the objects that lie closer to the boundary of the range query have a more significant effect on the shape of the safe zone and should be accessed first.

5.5 Updating the safe zone when query leaves it

When the query leaves its safe zone, it sends its current location and current guard objects to the server. The server updates the answer list (the list of internal objects), computes the new safe zone and sends it to the query. A straightforward approach is to compute the safe zone and answer list from scratch. However, this is not only expensive but can also cause a large amount of data to be transmitted from the server to the query if the answer list contains a large number of objects. In this section, we propose an effective approach to update the safe zone and the answer list, called *smart-update*.

6. Experiments

To evaluate the performance of our proposed approach, we compare our approach with an optimal algorithm and a naive algorithm. We assume that the optimal algorithm already knows the safe zone and updates the results only when the query leaves the safe zone. To compute the initial results, the optimal algorithm visits the objects that lie within the range. To update the results, the algorithm searches only the area that may contain the new answers. We only consider the I/O cost for the optimal algorithm (the CPU time is assumed to be zero).

The naive algorithm prunes every object o_i such that its circle does not intersect with the circle of any guard object. That is, an object or rectangle can be pruned if its distance from all guard objects is greater than $2r$.

All the experiments were conducted on Intel Xeon 2.4 GHz dual CPU with 4 GBytes memory. We used real dataset as well as synthetic dataset. The real dataset[3] contains 175, 813 points of interests in North America that corresponds to a data universe of 5000Km×5000Km. To verify the theoretical analysis, we created synthetic datasets consisting 50, 000 to 150, 000 points following uniform distribution within the same data universe size. The objects are indexed by R-tree with node size set to $2K$.

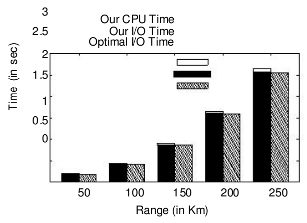
TABLE 1. Parameters and Range

Parameter	Range
Number of objects (×1000)	50, 75, **100**, 125, 150
Range (in Km)	50, 100, **150**, 200, 250
Average speed (in Km/hr)	40, 60, **80**, 100, 120

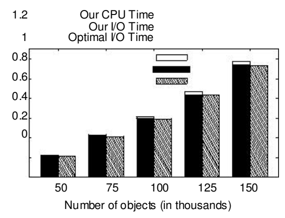
We simulated moving queries (moving cars) by using the spatio-temporal data generator [25]. The average speed of moving queries varies from 40 Km/hr to 120 Km/hr. All queries are continuously monitored for 5 minutes and the results shown correspond to the average monitoring cost for a single query for the 5 minutes duration. All the experiment results shown correspond to the real dataset except the results where we show the effect of number of objects. The table above shows the default parameters.

6.1 Cost comparison

The cost of each algorithm consists of I/O cost (by charging 2ms for each node access) and CPU cost (assumed zero for the optimal algorithm). The naive algorithm was at least 20 times slower than our algorithm for all settings so we exclude it from figures to better illustrate the comparison of our algorithm with the optimal algorithm. In Figure 7. A and Figure 7. B, we compare the cost of our algorithm with the cost of optimal algorithm for different ranges and different number of objects . The performance of our algorithm is close to the optimal algorithm. Main cost for our proposed approach is the I/O cost which is very close to the I/O cost of the optimal solution. This shows that the overhead of computing the safe zone is very small compared to the cost of the range query.



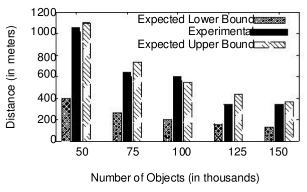
(a) Radius



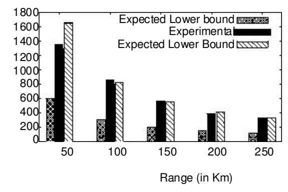
(b) Number of objects

Fig. 7 Efficiency

In Fig. 8, we show the expected distance for queries run on the synthetic dataset with increasing number of objects and increasing range of the query. It shows that the actual expected distance is close to the expected bounds. Moreover, the actual expected distance is from 300 meters to 1200 meters.



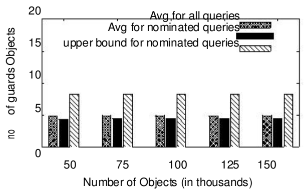
(a) Effect of data cardinality



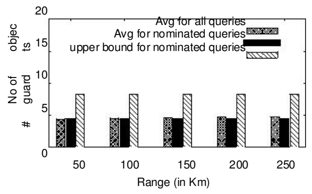
(b) Effect of range

Fig. 8 Expected Distance

Fig. 9 shows the average number of guard objects for all queries and compares the theoretical bound with the actual number of guard objects. As stated in Section V, our theoretical upper bound is valid for the queries for which maximum distance to the safe zone is smaller than $C \cdot m_{up}$ where C is a constant. We observed that when C is set to 2, 30% to 50% queries satisfy the constraint. We call such queries the nominated queries



(a) Effect of data cardinality



(b) Effect of range

Fig. 9 Number of guard objects

In Fig. 9, we show the average number of guard objects for all queries as well as the average number of guard objects for the nominated queries. It is interesting to note that the average number of guard objects for all queries is around 5 regardless of the experiment settings.

7. Conclusion

Our strategy does not require computation over the terminal devices. Therefore, cost of the terminal devices is reduced under the precondition of equal or better system performance. Secondly, terminal devices do not need to download the safe region information from the server which reduces the communication cost effectively. Finally, computation is simplified by applying circular safe regions. Hence the server workload is reduced which improves the system scalability. Possible future works of the research include implementation of the strategy in an applied MOD engine for a information system providing LBS to public transportation, taxis and private vehicle devices or pedestrians with hand-held mobile devices. Application of our strategy potentially provides real-time range queries and kNN queries to support LBS at a low cost with a high performance in addition to system design and implementation ease and flexibility.

References

[1] Y. Du, D. Zhang, and T. Xia, "The Optimal Location Query", Proceedings of International Symposium on Spatial and Temporal Databases, pp. 163-180, 2005.

[2] .L. Yiu, X. Dai, N. Mamoulis, and M. Vaitis, "Top-K Spatial Preference Queries," Proc. IEEE 23rd International Conference Data Engineering (ICDE), pp. 1076-1085, 2007

[3] S. Prabhakar, Y. Xia, D. V. Kalashnikov, W. G. Aref, and S. E. Hambrusch, "Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects," *IEEE Transaction Computers*, vol. 51, no. 10, pp. 1124–1140, 2002.

[4] M. F. Mokbel, X. Xiong, and W. G. Aref, "Sina: Scalable incremental processing of continuous queries in spatio-temporal databases," in *SIGMOD Conference*, 2004, pp. 623–634.

[5] X. Xiong, M. F. Mokbel, and W. G. Aref, "Sea-cnn: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases," in *ICDE*, 2005, pp. 643–654.

[6] G. S. Iwerks, H. Samet, and K. P. Smith, "Continuous k-nearest neighbour queries for continuously moving points with updates," in *VLDB*, 2003, pp. 512–523.

[7] X. Yu, K. Q. Pu, and N. Koudas, "Monitoring k-nearest neighbor queries over moving objects," in *ICDE*, 2005.

[8] Hua Lu, Member, IEEE, and Man Lung Yiu, "On Computing Farthest Dominated Locations", IEEE Transactions on Knowledge and Data Engineering. 23, No. 6, pp 928- 946 2011.

[9] M.A. Cheema, L. Brankovic, X. Lin, W. Zhang, and W. Wang, "Multi-Guarded Safe Zone: An Effective Technique to Monitor Moving Circular Range

Queries", Proceedings. IEEE 26th Int'l Conf. Data Eng. (ICDE), pp. 189-200, 2010.

[10] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest Neighbour Queries", Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 71-79, 1995.