International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.1, Jan-Feb 2012 pp-417-423 ISSN: 2249-6645

Advanced Query Evaluation Techniques for Preserving Privacy and Efficiency of Mobile Objects

K.Ruth Ramya[1],K.Priyanka[2],K.Anusha [2],K.Brahmini [2], G.Mohana Lakshmi[2],K.Abraham[2]

1(Assistant Professor, Dept. of Computer Science And Engineering, KL University.)
2(B.Tech Scholars, Dept. of Computer Science And Engineering, KL University.)

ABSTRACT

This paper presents a novel approach to overcome the difficulty and complexity in addressing the issues of location updating in terms of monitoring accuracy, efficiency, and privacy. Various distressing privacy violations caused by sharing sensitive location information with potentially malicious services have highlighted the importance of location privacy research aiming to protect users' privacy while interacting with Location Based Services. This paper presents a taxonomy of different approaches proposed to enable location privacy in LBS. Location privacy may be obtained at the cost of query performance and query accuracy. The challenge addressed is how to obtain the best possible performance, subjected to given requirements for location privacy and query accuracy. Our proposed framework uses the spacetwist and SCUBA techniques to obtain the privacy and efficiency for continuously moving objects. This approach is flexible, needs no trusted middleware, and requires only well-known incremental NN query processing on the server. This framework offers very good performance and high privacy, at low communication cost thereby providing higher efficiency.

***Keywords*- Cloaked region, Clusters, Granular Search, Spatiotemporal.**

I.INTRODUCTION

Every day we witness technological advances in wireless communications and positioning technologies. These developments paved the way to a tremendous amount of research in recent years in the field of real-time streaming and spatio- temporal databases[9,2]. As the number of users of location-based devices (e.g., GPS) continues to soar, new applications dealing with extremely large numbers of moving objects begin to emerge. These applications, faced with limited system resources and near real time response obligation call for new real-time spatiotemporal query processing algorithms[4].Such algorithms must efficiently handle extremely large numbers of moving objects and efficiently process large numbers of continuous spatiotemporal queries. The challenge addressed is how to obtain the best possible performance, subjected to given requirements for location privacy and efficiency at low query evaluation cost.

In a location based service(LBS) scenario, users query a server for nearby points of interest but they may not want to disclose their locations to the service. The benefits of LBS come at the cost of sharing private identity and location information of users with potentially untrusted entities offering such services. Sharing such sensitive information with untrusted servers has recently resulted in various distressing violations of users' privacy. To protect against various privacy threats while using LBS, several studies have proposed different approaches to protect the privacy Intuitively, location privacy may be obtained at the cost of query performance and query accuracy.

A taxonomy of approaches have been used for the location privacy problem.These approaches are based on anonymity/cloaking, transformation and private information retrieval (PIR) techniques. Our paper uses SpaceTwist to rectifiy the shortcomings of above techniques for k nearest neighbor (kNN) queries. This approach is flexible, needs no trusted middleware, requires only well-known incremental NN query processing on the server and also offers very good performance and high privacy, at low communication cost.

As for efficiency, two dominant costs are: the wireless communication cost for location updates and the query evaluation cost at the database server, both of which depend on the frequency of location updates. Present range and knn queries process and materialize every location update individually there by increasing query evaluation cost. With an extremely large number of objects and queries, this may simply become impossible. In order to reduce the cost i.e to increase the efficiency, here we now propose a two-pronged strategy towards combating this scalability problem. Our solution is based on the fact that in many applications objects naturally move in clusters. We take the concept of moving micro-clusters and exploit this concept towards the optimization of the execution of the spatio-temporal queries on moving objects.We propose the Scalable Cluster-Based Algorithm (SCUBA) for evaluating continuous spatio-temporal queries on moving objects.

II. RELATED WORK

We review existing location privacy protection techniques, which use either spatial cloaking or transformation-based matching.

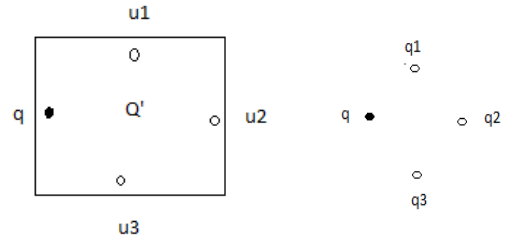
2.1.Related Work on Preseving Efficiency

2.1.1.Spatial Cloaking

With cloaking, the user location q is enlarged into a cloaked region Q^1 that is then used for querying the server [16]. This way, q is hidden in Q^1. The existing cloaking solutions differ with respect to (i) the representation of Q^1, (ii) the architecture for cloaking, and (iii) the query processing.

Cloaked Region Representation: Cloaked regions come in two forms: they are either plain, connected regions (e.g., rectangles) or they are discrete and posses "multiple parts" (e.g., sets of point locations). Q^1 by a K-anonymous [13] rectangle, which contains the query location q and at least $K - 1$ other user locations. Figure 1a illustrates a 4-anonymous region Q^1, where u1,u2, and u3 are $(4 - 1)$ user locations.Other work uses circular cloaked regions. The study of Ardagnae al. [17] takes location positioning inaccuracy into account, models the user location as a circular region, and develops several geometric operators for deriving cloaked regions. The cloaked region has also been represented by a point set containing q and a number of dummy locations (generated by the client) . In Figure 1b, q1, q2, q3 are dummy locations, and the cloaked region is Q0 = {q, q1, q2, q3}.

Cloaking Architecture:A simple approach to construct a cloaked region is to do so at the client [16], [17], [11].However, client-based cloaking does not support the use of K-anonymous regions. This requires knowing the locations of other users, which may be achieved by introducing a trusted, third party, location anonymize that knows the locations of a population of users.



(a) K-anonymous (b) dummies

Fig1. Example Cloaked Regions.

Server Side Query Processing: A discrete cloaked region query may be processed by processing each point in the query in turn, returning the union of results[12].In this setting, a negotiation protocol for trading between privacy and result accuracy has also been proposed.The processing of plain cloaked regions is more complex. Specialized (server-side) algorithms have been proposed for identifying a candidate set that includes the NN for any location in a cloaked region These algorithms go beyond well-known point NN algorithms and introduce complexity.

2.1.2.Transformation-Based Matching

Recently, transformation-based matching techniques have been proposed to enable location privacy. However, these do not offer query accuracy guarantees. A theoretical study on a client-server protocol for deriving the nearest neighbor of q has recently been reported. Its communication cost is asymptotic to pN, where N is the number of POIs. No experimental evaluation of the communication cost and result accuracy of the protocol with real data is available.

2.2.Related Work on Preseving Efficiency

Spatio-Temporal Query Processing: Efficient evaluation of spatio-temporal queries on moving objects has been an active area of research for quite some time. Several optimization techniques have been developed. These include Query Indexing and Velocity Constrained Indexing (VCI) [5],, shared execution[14], incremental evaluation [14], and query- aware moving objects involving high cost.To reduce wireless communication and query reevaluation costs, Hu et. al utilize the notion of safe region, making the moving objects query aware. Query reevaluation in this framework is triggered by location updates only. In this case, the authors combine it with different join policies to filter out the objects and queries that are guaranteed not to join. The limitations of this approach is that the devices may not have enough battery power and memory capacity to perform the complex computations. Our study falls into this category and distinguishes itself from these previous works by focusing on utilizing moving clusters abstracting similar moving entities to optimize the execution and minimize the individual processing[1, 25, 41]. We apply clustering as means to achieve scalable processing of continuous queries on moving objects.

III.FRAMEWORK OVERVIEW

As shown in Fig.2, the PAM[19] framework consists of components located at both the database server and the moving objects. At the database server side, we have the moving object index, the query index, the query processor, and the location manager. The object index is the server-side view on all objects. More specifically, to evaluate queries, the server must store the spatial range, in the form of a bounding box, within which each object can possibly locate. For each registered query, the query index stores: 1) the query parameters (e.g., the rectangle of a range query, the query point, and the k value of a kNN query); 2) the current query results; and 3) the quarantine area of the query. The quarantine area is used to identify the queries whose results might be affected by an incoming location

update. At moving objects' side, we have location updaters.

Without loss of generality, we make the following assumptions for simplicity:

- The number of objects is some orders of magnitude larger than that of queries. As such, the query index can accommodate all registered queries in main memory, while the object index can only accommodate all moving objects in secondary memory. This assumption has been widely adopted in many existing proposals[7], [10], [11].
- The database server handles location updates sequentially; in other words, updates are queued and handled on a first-come-first-serve basis. This is a reasonable assumption to relieve us from the issues of read/write consistency.
- The moving objects maintain good connection with the database server. Furthermore, the communication cost for any location update is a constant. With the latter assumption, minimizing the cost of location updates is equivalent to minimizing the total number of updates.

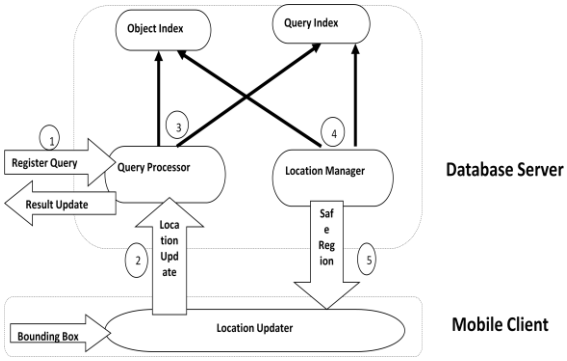


Fig2. PAM framework overview.

PAM framework works as follows (see Fig2): At any time, application servers can register spatial queries to the database server (step _1). When an object sends a location update (step_2), the query processor identifies those queries that are affected by this update using the query index, and then, reevaluates them using the object index (step _3). The updated query results are then reported to the application servers who register these queries. Afterward, the location manager computes the new safe region for the updating object (step _4), also based on the indexes, and then, sends it back as a response to the object (step _5). The procedure forprocessing a new query is similar, except that in step _2 , the new query is evaluated from scratch instead of being reevaluated incrementally, and that the objects whose safe regions are changed due to this new query must be notified.

Algorithm 1 summarizes the procedure at the database server to handle a query registration/ deregistration or a location update.

Algorithm 1: Overview of Database Behavior

1: while receiving a request do
2: if the request is to register query q then
3: evaluate q;
4: compute its quarantine area and insert it into the query index;
5: return the results to the application server;
6: update the changed safe regions of objects;
7: else if the request is to deregister query q then
8: remove q from the query index;
9: else if the request is a location update from object p then
10: determine the set of affected queries;
11: for each affected query q^1 do
12: reevaluate q^1;
13: update the results to the application server;
14: recomputed its quarantine area and update the query index;
15: update the safe region of p;

IV.SPACE TWIST

In this section we present different aspects of privacy related techniques and algorithms. Our proposed framework, called SpaceTwist, rectifies these shortcomings for k nearest neighbor(kNN) queries. Starting with a location different from the user's actual location, nearest neighbors are retrieved incrementally until the query is answered correctly by the mobile terminal.

4.1.The SpaceTwist Client Algorithm

We proceed to present the client-side algorithm for accurate kNN retrieval. We use the notation dist(q, p) to denote the Euclidean distance between two points q and p. The client (i.e., user) executes Algorithm 1 to obtain its k nearest objects from the server (i.e., query processor). The anchor location q^1 is first sent to the server. On the other hand, the user location q is known only by the client. Intuitively, if q and $q^1$are close then few objects are retrieved (i.e., low cost) but less location privacy is achieved. A max-heap W_k, initialized with k virtual objects, maintains the k nearest objects (of q) seen so far. Let γ be the maximum distance in Wk. The demand space is then the circle with radius γ and center q (see Line 3). Let τ be the largest distance to $q^1$of any object examined so

far. The supply space is then the circle with radius τ and center q^1 (see Line 4). Next, the server is requested to return incremental nearest neighbors (INNs) [1] of q^1.

Algorithm 2 Space Twist Client (for kNN query)
algorithm SpaceTwistClient(Value k, Point q, Point q^1)system parameter: packet capacity β

1: W_k ⟵ new max-heap of pairs <p, dist(q, p)>;

2: insert k pairs of <NULL,1> into W_k;

3: γ ⟵ the top distance of Wk; (kth best distance from q)

4: τ ⟵ 0; (furthest distance seen from q^1)

5: send an INN query with q^1 to the server;

6: while γ + dist(q, q1) > τ do

7: S⟵ get the next packet of points from the server;

8: τ ⟵ $\max_{p\in S}$ dist(q^1, p); (update supply space)

9: for all p€ S do

10: if dist(q, p) < γ then (check demand space)

11: update W_k (and γ) by using p;

12: terminate the INN query at the server;

13: return W_k;

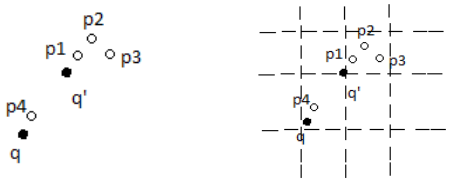
The following discuss a search technique that supports user specified granularities.

4.2. Granular Search

We develop a server-based granular search technique that is capable of retrieving data points from the server with a user specified granularity. This technique enables communication cost reduction and location privacy improvement while providing strict guarantees on the accuracies of the query results. Section 2.1 describes granular search for the case k=1; its implementation is covered in Section 2.2.

4.2.1.Basic Granular NN Search

Recall that the client-side algorithm requests POIs from the server in ascending order of their distance to anchor q1. For the example in Figure 3a, the server returns points in the order: p1, p2, p3, p4. Although p4 is the actual NN of q, it cannot be obtained early by the client.



(a) Set of points (b) Grid cells Fig. 3. Granular Search

The communication cost can be reduced by returning only a sample of the reported POIs. A threshold ϵ is then introduced for controlling the result accuracy.

4.2.2.Implementation of Granular Search

We proceed to consider the implementation of the above method. If the error bound €is given in advance, then it is possible to preselect a data point from each (non-empty) cell and index those points by another (small) R-tree, which is then used at query time. This pre-computation approach becomes impractical when different users use different values for € and may choose these values at run time. In the context of data streams, efficient main-memory data structures for maintaining relaxed results for NN queries with fixed error bounds have been proposed . We are unable to use these because (i) we deal with large, disk-based point sets, and (ii) they require the error bound to be known in advance. Algorithm 2 shows our granular incremental NN algorithm, which takes the user-specified error bound € as input. A conceptual grid with cell extent λ (=€/√2) is imposed on the returned points during runtime. The algorithm also takes an R-tree R (of the data points) and an anchor q^1 as arguments. The notation mindist(q1, e) (maxdist(q^1, e)) represents the minimum (maximum) possible distance between q^1 and an Rtree entry e [1], [1]. Next, $C_\lambda(p)$ denotes the cell containing point p.The algorithm applies INN search [1] around anchor q^1, with two modifications: (i) a set V is employed (Line 3) for tracking the grid cells of the reported points (Line12),and (ii) only qualifying entries that are not covered by the union of cells in V are further processed (Line9).

Algorithm 3 Granular Incremental NN
algorithm GranularINN(R-Tree R, Point q^1, Value €)

1: λ ⟵ €/√2;

2: H ⟵ new min-heap (mindist to q^1 as key);

3: V new set; (cells of reported points)

4: for all entries e € R.root do

5: insert <e,mindist(q^1, e)> into H;

6: while H is not empty do

7: deheap <e,mindist(q^1, e)> from H;

8: remove each cell c from V satisfying maxdist(q^1, c) < mindist(q^1, e);

9: if e is not covered by the union of cells in V then

10: if e is a point p then

11: report p to the client;

12: V ⟵ V € {C_λ(p)};

13: else

14: read the child node CN^1 pointed to by e;

15: for all entries e^1 € CN^1 do

16: insert <e^1,mindist(q^1, e^1)>into H;

These client-side processing algorithm and a server-side granular search technique supports user-defined (relaxed) query accuracies. SpaceTwist offers systematic support for managing the tradeoffsamong location privacy, query performance, and query accuracy in mobile services. Empirical studies with real-world datasets demonstrate that SpaceTwist is capable of providing high degrees of location privacy as well as very accurate results at low communication cost.

V.SCALABLE CLUSTER BASED ALGORITHM(SCUBA)

The Scalable Cluster-Based Algorithm (SCUBA) is used for evaluating continuous spatio-temporal queries on moving objects. SCUBA exploits a shared cluster-based execution paradigm, where moving objects and queries are grouped together into moving clusters based on common spatio-temporal attributes. Then execution of queries is abstracted as a join-between clusters and a join-within clusters executed periodically (every time units). In join-between, two clusters are tested for overlap (i.e., if they intersect with each other) as a cheap pre- filtering step. If the clusters are filtered out, the objects and queries belonging to these clusters are guaranteed to not join at an individual level. Thereafter, in join-within, individual objects and queries inside clusters are joined with each other. This two-step filter-and-join process helps reduce the number of unnecessary spatial joins.

5.1.The Notion of Moving Clusters

A moving cluster abstracts a set of moving objects and moving queries. We group both moving objects and moving queries into moving clusters based on common spatiotemporal properties i.e., with the intuition that the grouped entities travel closely together in time and space for some period. We consider the following attributes when grouping moving objects and queries into clusters: (1) speed, (2) direction of the movement (e.g., connection node on the road network), (3) relative spatial distance, and (4) time of when in that location. Moving objects and queries that don't satisfy conditions of any other existing clusters form their own clusters, single-member moving clusters.

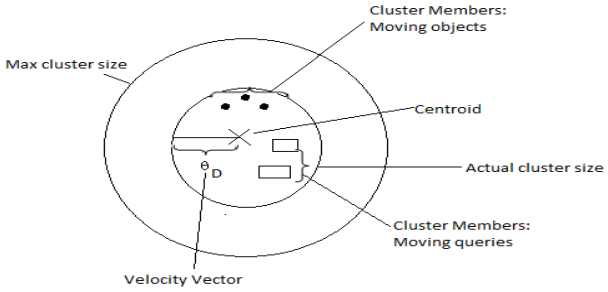


Fig4:Moving cluster in SCUBA

5.2.The SCUBA Algorithm

SCUBA execution has three phases: (1) cluster prejoin maintenance, (2) cluster based joining, and (3) cluster post-join maintenance as depicted in Fig. 5.The cluster pre-join maintenance phase is continuously running where it receives incoming information from moving objects and queries and applies in memory clustering. In this phase, depending on the incoming location updates, new clusters may be formed, "empty" clusters may be dissolved, and existing clusters may be expanded. The clusterbased joining phase is activated every time units where join-between and join within moving clusters is executed. The cluster post join phase is started by the end of the joining phase to perform a cluster maintenance for the next query evaluationtime.

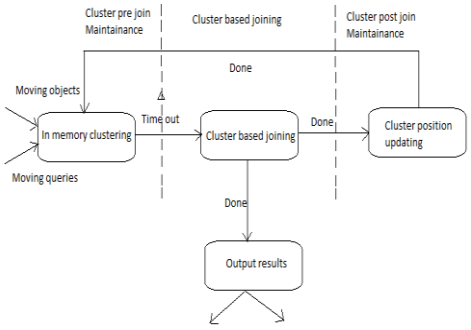


Fig5. State Diagram

Algorithm4 SCUBA()

1: loop

2: //*** CLUSTER PRE-JOIN MAINTENANCE PHASE ***

3: T_{start} = current time //initialize the execution interval start time

4: while (current time - Tstart) < ∆ do

5: if new location update arrived then

6:△ Cluster moving object o/query q //procedure described in Section 3.2// expires. Begin evaluation of queries

7: //*** CLUSTER-BASED JOINING PHASE ***

8: for c = 0 to MAX GRID CELL do

9: for every moving cluster m_L € G_c do

10: for every moving cluster m_R € G_c do

11: //if the same cluster, do only join-within

12: if (m_L == m_R) then

13: //do within-join only if the cluster contains members of different types

14: if ((m_L.OID_s > 0) && (m_L.QID_s > 0)) then

15: Call DoWithinClusterJoin(m_L,m_L)

16: else

17: //do between-join only if 2 clusters contain members of different types

18: if ((m_L.OID_s > 0) && (m_R.QID_s > 0)) ||

((m_L.QID_s > 0) && (m_R.OID_s > 0)) then

19: if DoBetweenClusterJoin(m_L,m_R) == TRUE then

20: Call DoWithinClusterJoin(m_L,m_R)

21: Send new query answers to users

22: //*** CLUSTER POST-JOIN MAINTENANCE PHASE ***

23: Call PostJoinClustersMaintenance() //do some cluster maintenance

Algorithm5 DoBetweenClusterJoin(Cluster mL, Cluster mR)

1: //Check if two circular clusters m_L and m_R overlap

2:if((m_L.Loc_t.x-m_R.Loc_t.x)2+(m_L.Loc_t.y -m_R.Loc_t.y)2) < (m_L.R - m_R.R)2 then

3: return TRUE; //the clusters overlap

4: else

5: return FALSE; //the clusters don't overlap

Algorithm4 shows the pseudo code for SCUBA execution. For each execution interval , SCUBA fi st initializes the interval start time (Step 3). Before time interval expires, SCUBA receives the incoming location updates from moving objects and queries and incrementally updates existing moving clusters or creates new ones (Step 6). When time interval expires (location updating is done), SCUBA starts the query execution (Step 8) by performing join-between clusters and join-within clusters. If two clusters are of the same type (all objects, or all queries), they arenot considered for the join-between. Similarly, if all of the members of the cluster are of the same type, no join-within is performed. The join-between checks if the circular regions of the two clusters overlap (Algorithm 2), and join-within performs a spatial join between the objects and queries of the two clusters (Algorithm 3). If join-between does not result in intersection, join-within is skipped.

Algorithm6 DoWithinClusterJoin(Cluster mL, Cluster mR)

1: R = Ø//set of results

2: S_q = Set of queries from m_L U mR //query members from both clusters

3: S_o = Set of objects from m_L U m_R //object members from both clusters

//join moving objects with queries from both clusters

4: for every moving object o_i €So do

5: for every moving query q_j € S_q do

6: spatial join between object o_i with query q_j (o_i q_j)⋈

7: S_r=Set of queries from joining oi with queries in S_q

8: for each Q €S_q do

9: add (Q, o_i) to R

10: return R;

After the joining phase, cluster maintenance is performed (Step 23). Due to space limitations, we don't include the pseudo-code for PostJoinClusters Maintenance().The operations performed during post-join cluster maintenance include dissolving "expiring" clusters and re-locating the "non-expiring" clusters (in the ClusterGrid) based on their velocity vectors for the next execution interval time (i.e.T + .).If at time T + the cluster △asses its dest△nation node, the cluster gets dissolved.SCUBA combines motion clustering with shared execution for query execution optimization. Given a set of moving objects and queries, SCUBA groups them into moving clusters based on common spatio-temporal attributes. To optimize the join execution, SCUBA performs a two-step join execution process by first pre-filtering a set of moving clusters that could produce potential results in the join-between moving clusters stage and then proceeding with the individual join-within execution on those selected moving clusters. Comprehensive experiments show that the performance of SCUBA is better than traditional grid-based approach where moving entities are processed individually. In particular the experiments demonstrate that SCUBA: (1) facilitates efficient execution of queries on moving objects that have common spatio-temporal attributes, (2) has low

cluster maintenance/overhead cost, and (3) naturally facilitates load shedding using motion clusters while optimizing the processing time with minimal degradation in result quality.

VI.CONCLUSION

This paper proposes a framework for monitoring continuous spatial queries over moving objects. The framework is the first to holistically address the issue of location updating with regard to monitoring accuracy, efficiency, and privacy. We provide detailed algorithms for query evaluation/Re evaluation and for protecting location privacy. Existing location privacy solutions either incur high server load, require specialized server implementations, or produce results without practical guarantees on accuracy bounds of query results. This paper concerns the efficient support for location privacy protection by using space twist algorithm. In this paper, we also proposed a unique algorithm for efficient processing of large numbers of spatio-temporal queries on moving objects termed SCUBA. SCUBA combines motion clustering with shared execution for query execution optimization.

VII.FUTURE WORK

Several promising research directions exist. First, it is relevant to extend the cost model to cover real data distributions, as the current model assumes uniform data and may not accurately reflect the distributions found in real-world data. Second, our proposal considers snapshot k nearest neighbor queries. It is of interest to extend them to support also continuous queries. As future work, we plan to further refine and validate moving cluster-driven load shedding, enhance SCUBA to produce results incrementally and explore further through additional experimentation.

REFERENCES

1. G. R. Hjaltason and H. Samet, "Distance Browsing in Spatial Databases," TODS, 24(2): 265–318, 1999.
2. S. E. Hambrusch, C.-M. Liu, W. G. Aref, and S. Prabhakar. Query processing in broadcasted spatial index trees. In SSTD, pages 502–521, 2001.
3. A. Okabe, B. Boots, K. Sugihara, and S. Chiu, Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, 2nd ed. Wiley, 2000. 51(10), 2002.
4. M. F. Mokbel and et. al. Towards scalable location-aware services: requirements and research issues. In GIS, pages 110–117, 2003.
5. S. Prabhakar and et.al. Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects. IEEE Trans. Computers.
6. Y. Li, J. Han, and J. Yang. Clustering moving objects. In KDD, pages 617–622, 2004.
7. D.V. Kalashnikov, S. Prabhakar, and S.E. Hambrusch, "Main Memory Evaluation of Monitoring Queries over Moving Objects".
8. M. F. Mokbel, X. Xiong, and W. G. Aref. Sina: Scalable incremental processing of continuous queries in spatio-temporal databases. In SIGMOD, pages 623–634, 2004.
9. B. Gedik and L. Liu. Mobieyes: Distributed processing of continuously moving queries on moving objects in a mobile system. In EDBT, pages 67–87, 2004.
10. H. Hu, J. Xu, and D.L. Lee, "A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects," Proc. ACM SIGMOD, pp. 479-490, 2005.
11. X. Yu, K.Q. Pu, and N. Koudas, "Monitoring k-Nearest Neighbor Queries over Moving Objects," Proc. IEEE Int'l Conf. Data Eng. (ICDE), 2005.
12. M. Duckham and L. Kulik, "A Formal Model of Obfuscation and Negotiation for Location Privacy," in PERVASIVE, pp. 152–170, 2005.
13. H. Kido, Y. Yanagisawa, and T. Satoh, "An Anonymous Communication Technique using Dummies for Location-based Services," in IEEE
14. 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, pages 643–654, 2005.
15. C.-Y. Chow, M. F. Mokbel, and X. Liu, "A Peer-to-Peer Spatial Cloaking Algorithm for Anonymous Location-based Services," in GIS, pp. 171–178, 2006.
16. M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The New Casper: Query Processing for Location Services without Compromising Privacy," in VLDB, pp. 763–774, 2006.
17. C. A. Ardagna, M. Cremonini, E. Damiani, S. D. C. di Vimercati, and P. Samarati, "Location Privacy Protection Through Obfuscation-Based Techniques,".
18. G. Ghinita, P. Kalnis, and S. Skiadopoulos, "PRIV´ E: AnonymousLocation-Based Queries in Distributed Mobile Systems," in WWW, pp. 371–380, 2007.
19. Haibo Hu, Jianliang Xu, Senior Member, IEEE, and Dik Lun Lee "PAM: An Efficient and Privacy-Aware Monitoring Framework for Continuously Moving Objects" IEEE ,TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, March 2010.