Adaptive Safe Regions for Continuous Spatial Queries over Moving Objects

Yu-Ling Hsueh[†] Roger Zimmermann[‡] Wei-Shinn Ku[§]
[†]Dept. of Computer Science, University of Southern California, USA
[‡]Computer Science Department, National University of Singapore, Singapore
[§]Dept. of Computer Science and Software Engineering, Auburn University, USA
{hsueh@usc.edu, rogerz@comp.nus.edu.sg, weishinn@auburn.edu}

Abstract. Continuous spatial queries retrieve a set of time-varying objects continuously during a given period of time. However, monitoring moving objects to maintain the correctness of the query results often incurs frequent location updates from these moving objects. To address this problem, existing solutions propose lazy updates, but such techniques generally avoid only a small fraction of all unnecessary location updates because of their basic approach (e.g., safe regions, time or distance thresholds). In this paper, we introduce an *Adaptive Safe Region (ASR)* technique that retrieves an adjustable safe region which is continuously reconciled with the surrounding dynamic queries. In addition, we design a framework that supports multiple query types (e.g., range and c-kNN queries). In this framework, our query re-evaluation algorithms take advantage of *ASRs* and issue location probes only to the affected data objects. Simulation results confirm that the *ASR* concept improves scalability and efficiency over existing methods by reducing the number of updates.

1 Introduction

Significant research attention has focused on efficiently processing continuous queries and its extension work that supports location-based services during the recent past. Existing work [2–4] has provided significant insight into the cost of the communication overhead by assuming a set of computationally capable moving objects that cache query-aware information (e.g., thresholds or safe regions) and locally determine a mobile-initiated location update. However, the focus of these solutions is mainly on static queries or simple types of queries (e.g., range queries). In this paper, we propose a framework to support multiple types of dynamic, continuous queries. Our goal is to minimize the communication overhead in a highly dynamic environment where both queries and objects change their locations frequently. When a new query enters the system we leverage the trajectory information that it can provide by registering its starting and destination points as a movement segment for continuous monitoring. This assumption can be easily extended to a more realistic scenario which may approximate a curved road segment with several straight-line sub-segments. We propose an *adaptive safe region* that reconciles the surrounding queries based on their movement trajectories such that the system can avoid unnecessary location probes to the objects in the vicinity (i.e., the ones which overlap with the current query region). Furthermore, our incremental result update mechanisms allow a query to issue location probes only to a minimum area where the query answers are guaranteed to be fulfilled. In particular, to lower the amortized communication cost for c-kNN queries, we obtain extra nearest neighbors (n more NNs) which are buffered and reused later to update the query results. Thus, the number of location updates incurred from the query region expansion due to query movement is reduced.

2 Related Work

Continuous monitoring of queries over moving objects has become an important research topic, because it supports various useful mobile applications. Prabhakar et al. [4] designed two approaches named query indexing and velocity constrained indexing with the concept of *safe regions* to reduce the number of updates. Hu et al. [2] proposed a generic framework to handle continuous queries with safe regions through which the location updates from mobile clients are further reduced. However, these methods only address part of the mobility challenge since they are based on the assumption that queries are static which is not always true in real world applications. A threshold-based algorithm is presented in [3] which aims to minimize the network cost when handling c-kNN queries. To each moving object a threshold is transmitted and when its moving distance exceeds the threshold, the moving object issues an update. However, the system suffers from many downlink message transmissions for refreshing the thresholds of the entire moving object population due to frequent query movements. Cheng et al. [1] proposed a time-based location update mechanism to improve the temporal data inconsistency for the objects relevant to queries. Data objects with significance to the correctness of query results are required to send location updates more frequently. The main drawback of this method is that an object will repeatedly send location updates to the server when it is enclosed by a query region.

In contrast, our proposed techniques aim to reduce the communication cost of dynamic queries over moving objects and also support multiple types of queries. We utilizes adaptive safe regions to reduce the downlink messages of location probes due to query movements. Our *ASR*-based techniques surpass the aforementioned solutions with higher scalability and lower communication cost.

3 The System Overview

The location updates of a query result point (result point for short) and a non-result point (data point for short) are handled with two different mechanisms. An *adaptive safe region* (*ASR*) is computed for each data point. A mobile-initiated voluntary location update is issued when any data point moves out of its safe region. For a result point, to capture its possible movement at the server side, we use a *moving region* (MR) whose boundary increases by the maximum moving distance per time unit. The location updates of a result point are requested only when the server sends server-initiated location probes triggered when the moving region of a result point overlaps with some query regions. The details of the ASR computation and an efficient mechanism that uses location probes for continuous query updates are described in Sections 3.1 and 3.2, respectively.

3.1 Adaptive Safe Region Computation

We propose a novel approach to retrieve an ASR, which is effective in reducing the amortized communication cost in a highly dynamic mobile environment. The key observation lies in the consideration of some important factors (e.g., the velocity or orientation of the query objects) to reconcile the size of the safe regions. The following lemma establishes the ASR radius based on the observation.

Lemma 1:

$$p_i.ASR.radius = \min(CDist(p_i, q_j) - q_j.QR.radius), \forall q_j \in Q, \text{ where}$$

$$CDist(p_i, q_j) = \begin{cases} \overline{p_i f'} & \text{if } \theta_j \leq \frac{\pi}{2} \text{ and } \exists f', \text{ or} \\ \overline{p_i q_j^s} & \text{if } \theta_j > \frac{\pi}{2} \text{ or } \nexists f' \end{cases}$$

As an illustration of Lemma 1 (and to explain the symbol notation), consider Figure 1, where the set of queries $Q = \{q_j, q_k\}$ are visited for retrieving the adaptive safe region (the dashed circle) of the data point p_i. The movement trajectory of q_j (and q_k) is denoted by $\overrightarrow{q_j} = [q_j^s, q_j^e])$, where q_j^s and q_j^e are the starting and ending points, respectively. We measure the Euclidian distance between a query and a data point ($CDist$ in Lemma 1) and then deduct the query range (denoted by $q_j.QR.radius$, where QR represents the query region of q_j). Lemma 1 captures two cases of $CDist$. The first case ($CDist(p_i, q_j)$) computes a distance $\overline{p_i f'} = \overline{q_j^s f}$ in the worst-case scenario where both p_i and q_j move toward each other (under the constraint of the maximum speed). f' represents the border point (on the border of $q_j.QR$ while q_j arrives at f on its movement segment), after which p_i would possibly enter the query region of q_j. f is the closest point to q_j^s on the trajectory of q_j, which satisfies the condition that the distance from p_i to f is equal to $\overline{p_i f'} + \overline{f' f}$, where $\overline{f' f} = q_j.QR.radius = r_j$. Let $\overline{p_i f'} = x$ for short. We can obtain the f and f' points by computing x first, which is considered the safe distance for p_i with respect to q_j. x can be easily computed with the trajectory information of q_j by solving the quadratic equation: $(x + r_j)^2 = h^2 + (\overline{q_j^s m} - x)^2$ (h is the height of triangle $\triangle p_i q_j^s m$). f on $\overrightarrow{q_j}$ exists only when θ_j ($\angle p_i q_j^s q_j^e$) is less or equal to $\frac{\pi}{2}$ and $\overline{p_i q_j^e} - q_j.QR.radius < \overline{q_j^s q_j^e}$ (triangle inequality). If the first case is not satisfied, we consider the second case ($CDist(p_i, q_k)$), which finds the maximum non-overlapping area with $q_j.QR$. Since $\theta > \frac{\pi}{2}$ in the second case, the query range of q_j can never cover p_i due to the opposing movement of q_j. In this example, the safe distance x (with respect to q_j) is smaller than y (with respect to q_k), so x is chosen as the radius of the ASR of p_i. In our system, since a c-kNN query can be considered an order-sensitive range query, we use the same principle to compute safe regions for each data object with respect to range queries and c-kNN queries. In case of a query insertion or query region expansion of a c-kNN query, the $ASRs$ of the affected data objects must be reassigned according to current queries to avoid any missing location updates.

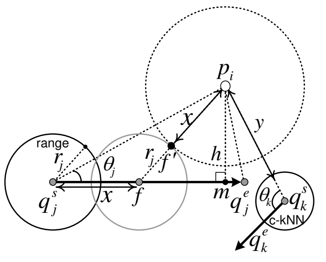


Fig. 1. An adaptive safe region

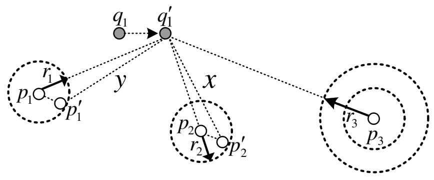


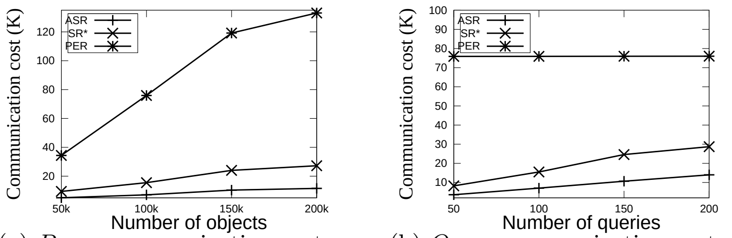
Fig. 2. The order checks of a c-kNN query

3.2 Query Evaluation with Location Probes

We propose an incremental query re-evaluation algorithms for both range and c-kNN queries. While updating the query answers, on-demand server-initiated location probes are issued whenever any location ambiguity exists. To handle a range query, the query processor sends the on-demand location probes to those result points that might move out of the current query regions. A *MR* for each result point is indexed on the grid and the boundary increases at each time step by the maximum moving distance until the result point is probed by the server. Since the number of result points are relatively small, indexing *MRs* does not significantly increase the overall server workload. For a data point, in addition to its adaptive safe region, we also consider the current possible moving boundary to serve as an additional indicator for the server to determine a necessary location probe. To handle a c-kNN query, the cost of updating c-kNN queries is usually higher than updating range queries. In our approach, the strategy to handle such increasing unnecessary location updates incurred from a c-kNN query is that the query processor computes $(k + n)$ NNs for a c-kNN query instead of evaluating exactly k NNs. This approach helps to reduce the number of future query region expansions to retrieve sufficient NNs for the queries. Since a c-kNN query is treated as an order-sensitive range query, we adopt the same principle that is used for a range query to find the new answer set in the current query regions first. A query region is expanded only when there are less than k NNs in the result set. Finally, an order-checking procedure is performed to examine the order of the result points and determine necessary location probes. Let q_j' be the last reported position of the query object q_j. The *OrderCheck* procedure checks each result point p_i (the i^{th} result point sorted by the *mindist to* q_j'), where $i = 1$ to k. A necessary location update of a kNN result point p_i is issued when the following condition is satisfied: $dist(q_j', p_i) + p_i.MR.radius \geq dist(q_j', p_{i+1}) - p_{i+1}.MR.r$, where $p_i.MR.radius$ (and $p_{i+1}.MR.radius$) is the maximum moving distance since the last update of p_i. An example is shown in Figure 2. The result set of q_1' is $\{p_2, p_1, p_3\}$ sorted by the distance between q_1' and their positions at the server since the last updates. The *OrderCheck* procedure first checks p_2 and p_1. Since $dist(q_1', p_2) + r_2 > dist(q_1', p_1) - r_1$, the order of p_2 and p_1 might need to be switched. The system needs to probe p_2 and p_1. After the location probes, the order of the NNs becomes $\{p_1', p_2', p_3\}$. Thus, the procedure checks the next pair of p_2' and p_3. Since $dist(q_1', p_2') < dist(q_1', p_3) - r_3$, the location probe of p_3 is not necessary.

4 Experimental Evaluation

We evaluated the performance of the proposed framework that utilizes *ASRs* and compared it with the traditional safe region approach [2, 4] and a periodic update approach (*PER*). The periodic technique functions as a baseline algorithm where each object issues a location update (only uplink messages are issued in this approach) every time it moves to a new position. We extended the safe region approach (*SR**) to handle dynamic range and c-kNN queries where the result points are monitored the same way as in *ASR*. We preserve the traditional safe region calculations (maximum non-overlapping area) for the SR* approach. Figure 3(a) shows the communication overhead of *ASR*, *SR** and *PER* with respect to the object cardinality, where all query and object sets move with a mobility rate of 50%. *ASR* outperforms *SR** and *PER*. The difference increases as the number of objects grows. Figure 3(b) shows the impact of the number of queries. Our algorithm achieves about 50% reduction compared with *SR** and 90% reduction compared with *PER*.



(a) P v.s. communication cost (b) Q v.s. communication cost

Fig. 3. Object and Query Cardinality

5 Conclusions

We have designed an *ASR*-based framework for highly dynamic environments where mobile units may freely change their locations. The novel concept of an adaptive safe region is introduced to provide a mobile object with a reasonable-sized safe region that adapts to the surrounding queries. Hence, the communication overhead resulting from the query movements is greatly reduced. An incremental result update mechanism that checks only the set of affected points to refresh the query answers is presented. Experimental results demonstrate that our approach scales better than existing techniques in terms of the communication cost and the outcome confirms the feasibility of the *ASRs* approach.

References

1. R. Cheng, K. yiu Lam, S. Prabhakar, and B. Liang. An Efficient Location Update Mechanism for Continuous Queries Over Moving Objects. *Inf. Syst.*, 32(4):593–620, 2007.
2. H. Hu, J. Xu, and D. L. Lee. A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects. In *SIGMOD Conference*, pages 479–490, 2005.
3. K. Mouratidis, D. Papadias, S. Bakiras, and Y. Tao. A Threshold-Based Algorithm for Continuous Monitoring of k Nearest Neighbors. *IEEE Trans. Knowl. Data Eng.*, 17(11):1451–1464, 2005.
4. 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 Trans. Computers*, 51(10):1124–1140, 2002.