Independent Safe Region Updation and Meeting point Notification using Tile based Safe Region Approach

Soundarya D.S. Rajan[1], J.Jeeva Santhana Sevi [2]

soundaryadsrajan@gmail.com

M.E Scholar, Dept of Computer Science and Engineering, RIIT,Kanyakumari District, India[1]

Associate Professor, Dept of Computer Science and Engineering, RIIT,Kanyakumari District, India[2]

Abstract— In applications like social networking services and online games, multiple moving users which form a group may wish to be continuously notified about the best meeting point from their locations.A promising technique for reducing the communication frequency of the application server is to employ safe regions, which capture the validity of query results with respect to the users locations.Unfortunately, the safe regions in the problem exhibit characteristics such as irregular shapes and inter-dependencies,We design efficient algorithms for computing these safe regions, as well as develop compression techniques for representing safe regions in a compact manner.This system proposes the Circular Safe Region Approach and Tile-Based Safe Region Approach.

Index Terms—Query processing, spatial databases

I. INTRODUCTION

RECENTLY, social networking services in the ad-hoc mobile environment have attracted significant attention. Such services exist in many popular social websites including Facebook and Foursquare.1 Managing the moving data arising from such services brings new challenges due to both spatial and social constraints.In this paper, we propose a novel monitoring problem,Meeting Point Notification (MPN) for multiple moving users: given a group of moving users U and a set of points of interest (POI) P, MPN continuously reports the optimal meeting point p^o €P to users in U such that the maximum distance between any user and p^o is minimized.MPN is motivated by many applications in social networks, location-based games and massively multiplayer online (MMO) games.

A real application relevant to MPN is EchoEcho,invented by Google Venture. EchoEcho assists users to browse their friends' real-time locations and share their own. As a highlight feature,

EchoEcho allows a user to continuously observe her friends' locations regarding to a predetermined meeting point. Mobile users with such interests have also been investigated in the collaborative system research.

A promising technique for reducing the communication frequency of the application server is to employ safe regions, which capture the validity of query results with respect to the user's locations.Unfortunately, the safe regions in our problem exhibit characteristics such as irregular shapes and inter-dependencies,which render existing methods that compute a single safe region inapplicable to our problem. To tackle these challenges, we first examine the shapes of safe regions in our problem's context and propose feasible approximations for them. We design efficient algorithms for computing these safe regions. We also study a variant of the problem called the sum-optimal meeting point and extend our solutions to solve this

variant. Experiments with both real and synthetic data demonstrate the effectiveness of our proposal in terms of computational and communication costs. Limitations of bandwidth and battery power raise challenges for mobile search problems, including MPN. Thus, the main optimization goal for these applications is to minimize Communication frequency. This goal also reduces unnecessary computational workload at the server because the communication cost between the clients and the server is reduced.We adopt the same goal: minimize the communication frequency, i.e., the frequency by which users issue update messages to the server.

we propose novel solutions based on the safe region concept. Safe regions are a set of geographical regions, one for each user, such that if each user stays inside her region, the query result will remain the same.The use of safe regions for multiple users raises several challenges. First, existing safe region computation techniques for a single user are not applicable for computing safe regions for a group of users, because these regions are not independent. Second, the safe regions have irregular shapes, unlike simple-shaped safe regions considered in previous work (e.g., a Voronoi cell). Third, it is infeasible to pre-compute the safe regions for multiple users because multiple safe regions depend on the multiple locations of moving users, which are unpredictable.

In our preliminary work, we have proposed circular safe regions that are easy to compute, and tile-based safe regions that offer better approximations of maximal safe regions. In this paper, our new contributions include:

- a buffering optimization that avoids repeated index accesses,
- a problem variant called the sum-optimal meeting point and our solutions for it, and
- additional experiments that demonstrate effectiveness and efficiency of our new contributions

II. RELATED WORK

Previous work on processing moving queries over mobile data can be classified into two categories :i) report query results to a single user continuously, e.g., kNN queries,circular range queries, moving window queries;ii) detect relationships among moving objects, e.g., proximity detection and constraints monitoring.

The safe region concept has been widely used in moving query processing to reduce the communication cost between clients and servers.When a user registers a continuous query, the server will return POIs along with a safe region. The query result remains the same if the user stays inside the current safe region. Upon leaving the safe region, the user requests from the server a updated result together with a new safe region. The shape of the safe region depends on the query type, e.g., an order-k Voronoi cell for a kNN query, or an arc-based region for arrange query. Defining safe regions for our problem is challenging because the safe regions for MPN have irregular shapes and are thus hard to compute; the safe regions of users are interdependent and the users change their locations dynamically and unpredictably,rendering pre-computation techniques (e.g.,as Verona cells) inapplicable.

Proximity detection helps a user to maintain a list of friends who are within a distance threshold from her. Since both the user and her friends are moving, Yiu et al. propose self-tuning policies to automatically assign an adjustable safe region for each user. However, the work of does not consider POIs where the users are supposed to meet.

III. PROBLEM SETTING

We first introduce the preliminary concepts and the system architecture. Then, we illustrate the unique characteristics of the search space and safe regions in our problem. In the end, we state our main objectives in this paper.

A. System Architecture

In this paper, we adopt the client-server architecture which is widely used in moving query processing.The server manages a data set P of points-of-interest (e.g., restaurants, cafes) and indexes it by an R-tree. A group of users U wish to receive notifications of their optimal meeting point p^o €P from the server continuously. Besides the result p^o, the server also reports a safe region R_i to each user u_i €U. The optimal meeting point remains unchanged if every user u_i moves within her safe region R_i. Therefore,these safe regions serve to reduce the communication frequency of the server (and its computational overhead)significantly.

The system is triggered when a user u_i € U leaves her safe region R_i. Then, u_i sends her current location to the server (Step 1). Next, the server probes the current locations of other users in the group U (Step 2). Having received replies from all users in U, the server recomputes and notifies each user u_i about the optimal meeting point p^o and her corresponding safe region Ri (Step 3). In summary, the server and users communicate via three types of messages.Maximal safe regions have irregular shapes and raise challenges in computation and representation. Our objectives are as follows:

1) Design concise representations for safe regions;

2) Develop efficient algorithms for computing them.

In this paper, we will investigate conservative approximations for maximal safe regions. Specifically, we will study circular safe regions and tile-based safe regions.

B.Characteristics of Safe Region Group

This section describes the unique characteristics exhibited by the safe regions in our problem.The possible groups of safe regions indeed form a huge search space: a m _ d dimensional space, where m is the number of users and d is the number of spatial dimensions. For example, for two users and the planar space, the search space becomes four-dimensional.

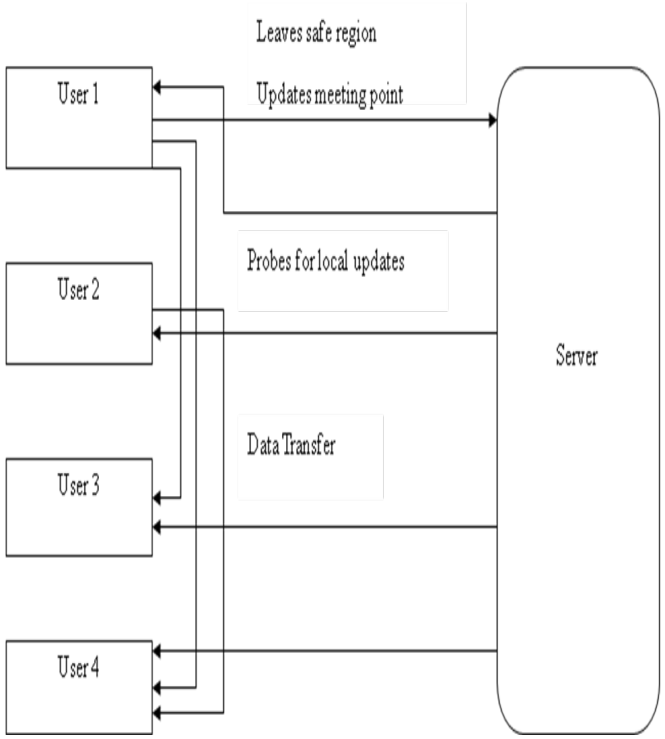

Fig:System Architecture

IV. CIRCULAR SAFE REGION APPROACH

In this section, we approximate the maximal safe regions of users by circles due to simplicity. We first study the condition for verifying a set of safe regions. Then, we design an algorithm for computing circular safe regions.

The advantage of circular safe regions is that they can be computed efficiently. However, they suffer from the drawback that they cannot serve as tight approximations of maximal safe regions.In the next section, we propose a tighter approximation of maximal safe regions, named *the tile-based safe regions.*

A. Verification of Safe Regions

An essential task in our problem is to verify whether a set of regions satisfies Definition.By definition, there are infinitely many instances of user locations in those regions.Thus, it is infeasible to test all the instances one-by-one.In this section, we plan to establish a conservative conditionfor verifying safe regions in an efficient manner. Before that,we first define *dominant distances* and *dominant user*.

B. Algorithm

Although maximal safe regions have irregular shapes, they can be conservatively approximated as circles. We now assign each user *ui* a circular safe region, where *ui* is the current user location and *r* is the radius. Note that the same radius *r* is used across different *Ri*.To reduce the communication cost between the server and the users, the value *r* should be as large as possible. The following theorem decides the maximum radius *r* such that the safe regions remain valid.

We are aware of a tight pruning technique [25] that utilizes half-spaces for deciding whether every point in a query rectangle R is closer to an object rectangle rather than another object rectangle. Nevertheless, this technique is not applicable to our problem because: i) we use multiple safe regions for multiple users respectively, instead of using a single rectangle R, and ii) a safe region group can be valid even when some part of R do not take as the nearest neighbor.

V. TILE-BASEDSAFE REGION APPROACH

A tile, as its name implies, is a square region.Tiles can be assembled to represent an irregular shape and thus serve as a tighter approximation of maximal safe regions. A tile-based safe region can be represented in a concise manner, as shown in our preliminary work; we omit these techniques here due to space limitations. In the remainder of this section, we first show a tighter verification method for tiles. Next, we design an algorithm for computing such tile-based safe regions. Then, we propose techniques to optimize the efficiency of tile verification. Finally, we suggest a buffering optimization that avoids repeated accesses to an R-tree.

Compression of Safe Regions

A tile-based safe region *Ri* may contain a large number of tiles, thus incurring significant communication cost to report it to the user. For example, a region that contains 15 large tiles and 32 small tiles. In this section, we investigate techniques that compress the description of *Ri* in order to reduce its packets for communication.

Loseless compression. :

First, we propose a lossless compression technique that reduces the description of *Ri* while preserving the exact area covered by *Ri*. The idea is to combine adjacent tiles together to form a rectangle. Interestingly, by allowing overlaps among rectangles, we are able to reduce the total number of rectangles for describing *Ri*. In this example, *Ri* can be compressed into 7 rectangles. This lossless compression can be implemented by adapting a greedy algorithm for the set-cover problem.

5.1 DIVIDE-AND-CONQUER VERIFICATION FOR TILES

To tackle this problem, we propose a divide-and-conquer method for verification.The initial size of the tile s will

be discussed in the next section. The parameter L is used to control the number of recursion levels (and thus the computation cost). The algorithm aims to check whether s is a valid safe region for user ui with respect to the existing safe regions of other users in R.

5.2 ALGORITHM

Having introduced a divide-and-conquer verification method Divide-Verify, we are ready to present an algorithm for computing tile-based safe regions. Each safe region Ri is modeled as a set of tiles, so it can be used to approximate an irregular shape. The main idea of the algorithm is to browse the tiles around each user ui in a systematic way, apply verification on them, and then add valid tiles into a safe region Ri.

We call a function Next-Tile to get the next tile s for user ui. The implementation of Next-Tile will be discussed shortly. Then, it tests the new tile s with other users' safe regions by calling Divide-Verify (line 9). The loop terminates either when i) the test returns true, or ii) s is empty, i.e.,Next-Tile has exhausted all tiles for ui.. At the end, the algorithm returns a safe region Ri to each user ui.

We examine two possible orderings for NextTile to select the next tile. In Fig. 8, the tiles are numbered by the their insertion order. The first tile centered at ui is numbered as 0.

Undirected ordering.

This approach picks the next tile based on the anti-clockwise order.When all tiles in the current layer have been exhausted, it checks whether some tile in the current layer has been inserted in the safe region. If yes, then it picks the next tile in an outer layer and repeats the process. Otherwise, it returns a null tile, meaning that any subsequent tile cannot become a valid tile for the user.

Directed ordering.

Existing studies show that the travel direction of a user ui in the near future has a limited angle deviation u from his current one. u is learned from ui's recent travel directions. We can exploit this feature and examine only the tiles whose subtended angles at ui deviate by less than u. By incorporating this idea into the above.

5.3 EFFICIENT IMPLEMENTATION OF TILE VERIFICATION

The running time of Algorithm is dominated by the time for verifying tiles, i.e., the recursive Divide-Verify function.This function needs to invoke the Tile-Verify function for every point p. In this section, we optimize this step in order to reduce the overall running time.

We first study how the Tile-Verify function can be implemented efficiently. Then, we propose a technique for pruning a large portion of points in P without processing
them one-by-one.

Individual Tile Verification (IT-Verify).

This is a basic technique for verifying a new tile s to be allocated to user ui.IT-Verify would enumerate all possible tile groups (as defined above) and verify them. If any group fails, then s is not valid as part of the safe region of user ui. However, such an implementation suffers from high computation cost due to the huge number of tile groups formed by the safe regions of other users uj

Group Tile Verification (GT-Verify).

Instead, we propose an optimized verification method for the new tile s. The mainidea of GT-Verify is to group tiles and test entire groups collectively,reducing the total number of checks significantly.We illustrate two main types of grouping strategies.

VI. EXPERIMENTS

In this section, we experimentally evaluate the performance of our proposed techniques. All methods were implemented in C++ and the experiments were performed on an Intel Core2Duo 2.66 GHz CPU machine with 8 GB memory,running on Ubuntu 10.04.

Measures.We evaluate our performance in three aspects:

i) update frequency, which reflects the frequency for users to issue update messages to the server, and

ii) average running time, which is the computation time for safe regions per update.

iii) Communication cost (packet count), measures the number TCP packets for messages sent between the server and the clients.

6.1 Scalability Experiments (for MPN)

In this section, we compare the circle-based safe regions and the tile-based safe regions.

Effect of user group size m.

We vary the group size m in experiments. The update frequency of Tile is less than half of Circle. Tile-D reduces the

update frequency further, since it applies the directed ordering and covers more tiles for future possible locations. Due to the lossless safe-region compression technique in , our methods require only a few packets per sending a tile-based safe region. Thus, our methods still incur lower communication cost than Circle.As expected, the running time grows with m. Circle is efficient to compute but has a larger update frequency than tile-based safe regions; our tile-based safe regions are much more effective in optimizing the update frequency.

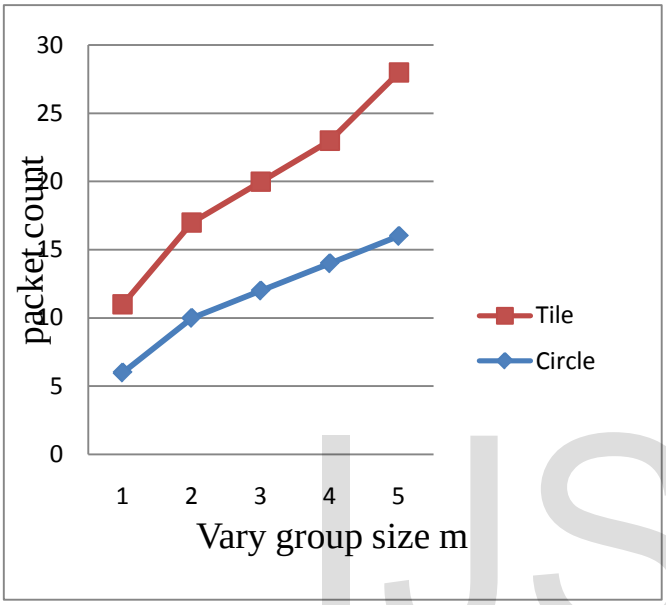

Fig: Vary group size m

Effect of data size n.

We vary the data size (i.e., the number of POIs) . As depicted in both data sets, the update frequencies of the methods increases because more POIs become as the candidates for the optimal points. Besides,Circle has a larger increase than those of methods based on the tile-based safe regions. Note that the communication costs of the methods are proportional to their corresponding update frequencies.

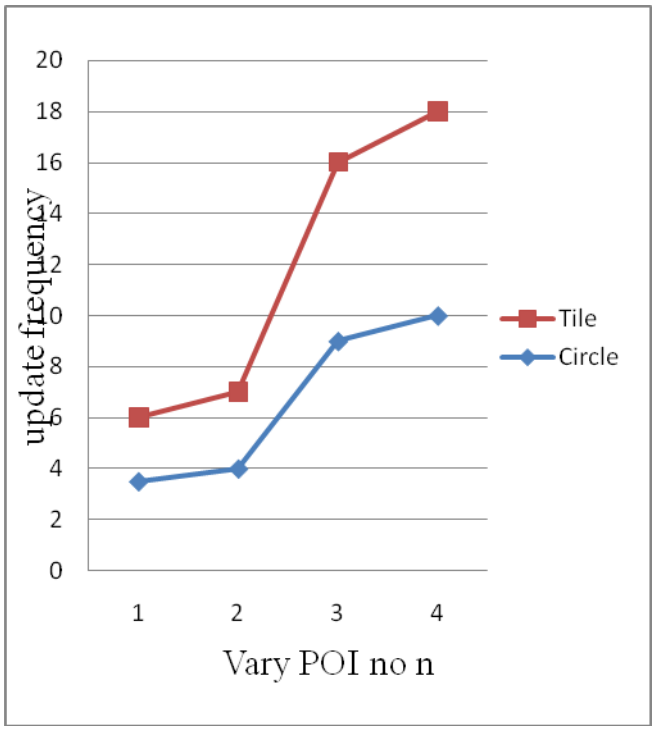

Fig: Vary POI number n, as a fraction of data size N.

Effect of user speed.

We proceed to vary the speed of users in this experiment. Fig. shows the update frequency and the communication cost of the methods with respect to the speed . Intuitively, as users move faster, they escape their safe regions quickly. Thus, all the methods have a large update frequency and communication cost at a high speed.

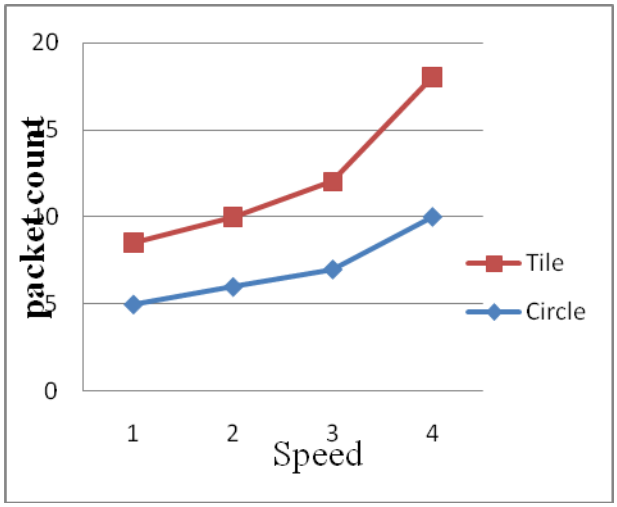

Fig:Vary speed

Configurations.

We study our proposed solutions with different variations.Circle denotes the Circle-MSR method. Tile denotes the Tile-MSR method using undirected ordering on tiles and lossless compression.Tile-D is a variant of Tile using directed ordering on tiles. Both Tile and Tile-D apply the GT-Verify function and index pruning technique. Our proposed methods require two extra parameters: i)the tile limit a, and ii) the split limit L.

Circle has the lowest running time, but it incurs higher update frequency and communication cost (packet count) than our tile-based methods.

Tile-D achieves the best update frequency and communication cost. Furthermore, our buffering optimization offers a substantial saving in the running time while only slightly increases the update frequency

VII. CONCLUSION

In this paper, we focus on minimizing the communication cost for monitoring the optimal meeting point for a group of users. We propose the concept of independent safe region group,in order to reduce the communication frequency of users.We design efficient algorithms and various optimizations to compute these safe regions. Also, we have studied a problem variant of the optimal meeting point based on the sum of distances.

In future, we plan to extend our techniques to the road network space. For Circle, we may replace a circular region by a range search region over road segments. For Tile, we may replace recursive tiles by recursive partitions of the road network. Also, we will develop a cost model for estimating the update frequency, the communication cost, and the running time of our methods.

REFERENCES

[1] E. Sarig€ol, O. Riva, P. Stuedi, and G. Alonso, “Enabling social networking in ad hoc networks of mobile phones,” Proc. VLDB Endowment, vol. 2, no. 2, pp. 1634–1637, 2009.

[2] N. Gupta, A. J. Demers, and J. Gehrke, “Semmo: A scalable engine for massively multiplayer online games,” in Proc. ACM SIGMOD Int. Conf. Management Data, 2008, pp. 1235–1238.

[3] A. J. Demers, J. Gehrke, C. Koch, B. Sowell, and W. M. White,“Database research in computer games,” in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2009, pp. 1011–1014.

[4] C. Lampe, N. B. Ellison, and C. Steinfield, “A face(book) in the crowd: Social searching versus social browsing,” in Proc.20th Anniv. Conf. Comput. Supported Cooperative Work, 2006,pp. 167–170.

[5] H. Hu, J. Xu, and D. L. Lee, “A generic framework for monitoring continuous spatial queries over moving objects,” in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2005, pp. 479–490.

[6] K. Mouratidis, D. Papadias, S. Bakiras, and Y. Tao, “A thresholdbased algorithm for continuous monitoring of k nearest neighbors,” IEEE Trans. Knowl. Data Eng., vol. 17, no. 11, pp. 1451–1464, Nov. 2005.

[7] S. Nutanong, R. Zhang, E. Tanin, and L. Kulik, “The v*-diagram:A query-dependent approach to moving knn queries,” Proc. VLDB Endowment, vol. 1, no. 1, 2008, pp. 1096–1106.

[8] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee, “Locationbased spatial queries,” in Proc. ACM SIGMOD Int. Conf. Manage.Data, 2003, pp. 443–454.

[9] M. L. Yiu, L. H. U, S. Saltenis, and K. Tzoumas, “Efficient proximitydetection among mobile users via self-tuning policies,” Proc.VLDB Endowment, vol. 3, no. 1, 2010, pp. 985–996.

[10] L. Qin, J. X. Yu, B. Ding, and Y. Ishikawa, “Monitoring aggregate k-nn objects in road networks,” in Proc. 20th Int. Conf. Sci. Statist.Database Manage., 2008, pp. 168–186.

[11] Y. Tao, D. Papadias, and Q. Shen, “Continuous nearest neighbor search,” in Proc. 28th Int. Conf. Very Large Data Bases, 2002,pp. 287–298.

[12] J. Li, M. L. Yiu, and N. Mamoulis, “Efficient notification of meeting points for moving groups via independent safe regions,” in Proc. IEEE Int. Conf. Data Eng., 2013, pp. 422–433.

[13] G. S. Iwerks, H. Samet, and K. P. Smith, “Continuous k-nearest neighbor queries for continuously moving points with updates,” in Proc. 28th Int. Conf. Very Large Data Bases, 2003,pp. 512–523.

[14] K. Mouratidis, M. Hadjieleftheriou, and D. Papadias, “Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring,” in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2005,pp. 634–645.

[15] X. Yu, K. Q. Pu, and N. Koudas, “Monitoring k-nearest neighbor queries over moving objects,” in Proc. IEEE Int. Conf. Data Eng.,2005, pp. 631–642.

[16] X. Xiong, M. F. Mokbel, and W. G. Aref, “Sea-cnn: Scalable processing of continuous k-nearest neighbor

queries in spatiotemporal databases," in Proc. IEEE Int. Conf. Data Eng., 2005,pp. 643–654.

[17] 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," in Proc. IEEE Int. Conf. Data Eng., 2010,pp. 189–200.

[18] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee, "Locationbased spatial queries," in Proc. ACM SIGMOD Int. Conf. Manage.Data, 2003, pp. 443–454.

[19] Z. Xu and H.-A. Jacobsen, "Adaptive location constraint processing,"in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2007,pp. 581–592.

[20] B. Zheng and D. L. Lee, "Semantic caching in location-dependent query processing," in Proc. 7th Int. Symp. Adv. Spatial Temporal Databases, 2001, pp. 97–116.

[21] D. Papadias, Y. Tao, K. Mouratidis, and C. K. Hui, "Aggregate nearest neighbor queries in spatial databases," Trans. Database Syst., vol. 30, no. 2, pp. 529–576, 2005.

[22] F. Li, B. Yao, and P. Kumar, "Group enclosing queries," IEEE Trans. Knowl. Data Eng., vol. 23, no. 10, pp. 1526–1540, Oct., 2011.

[23] H. G. Elmongui, M. F. Mokbel, and W. G. Aref, "Continuous aggregate nearest neighbor queries," GeoInformatica, vol. 17,pp. 63–95, 2011.

[24] D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis, "Group nearest neighbor queries," in Proc. IEEE Int. Conf. Data Eng., 2004,pp. 301–312.

[25] T. Emrich, H.-P. Kriegel, P. Kr€oger, M. Renz, and A. Z€ufle,"Boosting spatial pruning: On optimal pruning of mbrs," in Proc.ACM SIGMOD Int. Conf. Manage. Data, 2010, pp. 39–50.