An Efficient and Robust Access Method for Spatial Queries in Mobile Environments

T. KalaiSelvi[1] Mr .G.Ravi [2]
[1]Research Scholar [2]Associate Professor & Head of Department
[1,2]Department of computer science engineering
[1,2]Jamal Mohamed College, Trichirapalli, Tamilnadu,India

Abstract— A proxy-based approach to continuous nearest neighbor search query and window queries has been implementing in this paper and get a better result over mobile environment. The proxy creates estimated valid regions for mobile clients by exploiting spatial and temporal locality of spatial queries. For Nearest Neighbor queries, to devise two new algorithms to accelerate Estimated Valid Region growth, leading the proxy to build effective Estimated Valid Regions even when the cache size is small. On the other hand, to propose to represent the Estimated Valid Regions of window queries in the form of vectors, called estimated window vectors, to achieve larger estimated valid regions the above techniques developed for .net application and display for better result, efficient output in spatial search in around nearest location based on spatial queries.

Key words: Nearest neighbor query, window query, spatial query processing, Estimated Valid Region

I. INTRODUCTION

LOCATION-BASED services, also known as location dependent information services (LDISs), have been recognized as an important context-aware application in pervasive computing environments. Spatial queries are one of the most important LBSs. According to spatial constraints, spatial queries can be divided into several categories including nearest neighbours queries and window queries. An Nearest query is to find the nearest data object with respect to the location at which the query is issued (referred to as the query location). For example, a user may launch an Nearest query like “show the nearest coffee shop with respect to my current location.” On the other hand, a window query is to find all the objects within a specific window frame. An example window query is “show all restaurants in my car navigation window.”

In general, a mobile client continuously launches spatial queries until the client obtains a satisfactory answering. For example, a query “show me the rate of the nearest hotel with respect to my current location” is continuously submitted in a moving car so as to find a desired hotel. The naive method answering continuous spatial queries is to submit a new query whenever the query location changes. The naïve method is able to provide correct results, but it poses the following problems:

High power consumption: The power consumption of a mobile device is high since the mobile device keeps submitting queries to the Location Based Server.

Heavy server load: A continuous query usually consists of a number of queries to the Location Based Server, thereby increasing the load on the server.

Fortunately, in the real world, the queries of a continuous query usually exhibit spatial locality. Thus, caching the query result and the corresponding valid region in the client side cache was proposed in to mitigate the above problems. The valid region, also known as the valid scope, of a query is the region where the answering of the query remains valid. Subsequent queries can be avoided as long as the clientis in the valid region. Note that as associated with a query by definition. However, as pointed out in, by associating a Valid Region with the corresponding object, the Valid Region of an object p can be used to resolve all the queries whose answering object is p.

The valid region, also known as the valid scope, of a query is the region where the answer of the query remains valid. Subsequent queries can be avoided as long as the client is in the valid region**.** Although VRs are useful, an LBS server may not provide VRs in practice since the server may simply provide query results only or would not compute VRs under heavy load.

In these situations, mobile service providers (e.g., Verizon Wireless and AT&T) or smartphone makers (e.g., Apple and RIM) can utilize a proxy architecture where the proxy provides estimated valid regions (EVRs), which are the sub regions of the corresponding VRs, for the clients. With the proxy architecture, the clients still can enjoy the benefits of EVRs even if the LBS server does not provide VRs and thus the mobile service providers and smartphone makers can attract more clients.

The contributions of this paper are summarized as follows:

For NN queries, we devise new algorithms to efficiently create new and extend existing EVRs. The devised algorithms not only enable mobile clients to obtain effective EVRs immediately but also lead the proxy to build effective EVRs even when the proxy cache size is small.

For window queries, different from others we propose to index the positions of data objects, instead of EVRs, by a grid index. Since VRs of window queries may not be convex polygons, creating effective polygonal EVRs by previous queries is inherently difficult. Thus, we propose to represent the EVRs of window queries in the form of vectors, called estimated window vectors (EWVs), to achieve larger estimated valid regions. Such novel representation and indexing lead the proxy to efficiently create more effective EVRs of window queries.

This paper introduce two index structures, an EVR-tree for NN queries and a grid index for window queries, due to the distinct characteristics of NN and window queries. We also develop algorithms to make these two index structures mutually support each other.

II. Literature Survey

A. Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments

Compared to traditional computing paradigms, mobile computing enables clients to have unrestricted mobility while maintaining network connection. The ability of users to move and identify their own locations opens up a new kind of information services, called location-dependent information services (LDISs), which produce the answer to a query according to the location of the client issuing the query. Mobile clients in wireless environments suffer from scarce bandwidth, low-quality communication, frequent network disconnections, and limited local resources. Data caching on mobile clients has been considered an effective solution to improve system performance. There are two common issues involved in client cache management: A cache invalidation scheme maintains data consistency between the client cache and the server; A cache replacement policy determines which data item(s) should be deleted from the cache when the cache does not have enough free space to accommodate anew item.

B. On Semantic Caching and Query Scheduling for Mobile Nearest-Neighbor Search

The movement of mobile clients presents many new research problems for location-dependent query processing there are several technical issues involved with the implementation of an LBS, which include locating the position of a mobile user, tracking and predicting movements, processing queries efficiently, and bounding location errors. In this paper, we focus on the efficient processing of location dependent queries and, in particular, a sub-class of queries called *mobile nearest-neighbor (NN) search*. The maintenance cost is high, especially for high dimensional space, thus hindering the application of the VD structure for complex and dynamic datasets.

C. Continuous Monitoring of Spatial Queries in Wireless Broadcast Environments

In this setting, there is a central server that monitors the locations of both objects and queries. The task of the server is to report and continuously update the query results as the clients and the objects move. As an example, consider that the data objects are vacant cabs and the clients are pedestrians that wish to know their *k* closest free taxis until they hire one. As the reverse case, the queries may correspond to vacant cabs, and each free taxi driver wishes to be continuously informed about his/her *k* closest pedestrians. Several monitoring methods have been proposed, covering both range and *k*NN queries. Some of these methods assume that objects issue updates whenever they move, while others consider that data objects have some computational capabilities, so that they inform the server only when their movement influences some query.

Snapshot queries over static data. Their adaptation to continuous queries and dynamic data would be inefficient, since they cannot utilize previous results and require query re-computation from scratch at the beginning of each broadcast cycle.

The processing load at the server side increases with the number of queries. In applications involving numerous clients, the server may be overwhelmed by their queries or take prohibitively long time to answer them.

III. System Analysis

A. Existing System

Caching valid regions of spatial queries at mobile clients is effective in reducing the number of queries submitted by mobile clients and query load on the server. However, mobile clients suffer from longer waiting time for the server to compute valid regions. We propose in this paper a proxy-based approach to continuous nearest-neighbor (NN) and window queries. The proxy creates estimated valid regions (EVRs) for mobile clients by exploiting spatial and temporal locality of spatial queries. For NN queries, we devise two new algorithms to accelerate EVR growth, leading the proxy to build effective EVRs even when the cache size is small.

Although VRs are useful, an LBS server may not provide VRs in practice since the server may simply provide query results only or would not compute VRs under heavy load. In these situations, mobile service providers (e.g., Verizon Wireless and AT&T) or smartphone makers (e.g., Apple and RIM) can utilize a proxy architecture where the proxy provides estimated valid regions (EVRs), which are the sub regions of the corresponding VRs, for the clients.

B. Proposed System

We propose to represent the EVRs of window queries in the form of vectors, called estimated window vectors (EWVs), to achieve larger estimated valid regions. This novel representation and the associated creation algorithm result in more effective EVRs of window queries. In addition, due to the distinct characteristics, we use separate index structures, namely EVR-tree and grid index, for NN queries and window queries, respectively. To further increase efficiency, we develop algorithms to exploit the results of NN queries to aid grid index growth, benefiting EWV creation of window queries. Similarly, the grid index is utilized to support NN query answering and EVR updating. We conduct several experiments for performance evaluation. The experimental results show that the proposed approach significantly outperforms the existing proxy-based approaches

In view of this, we propose in this paper a proxy architecture as well as several companion algorithms to provide EVRs of NN and window queries on static data objects for mobile clients. We aim to reduce the number of queries submitted by mobile clients, the time of obtaining query results and corresponding EVRs, and load on the LBS server.

IV. System Specification

A. Software Requirement

Front End/GUI Tool	: Microsoft Visual studio 2008
Operating System	: Windows Family
Language	: C#
Application	: Windows Application
Back end	: SQL-Server 2005

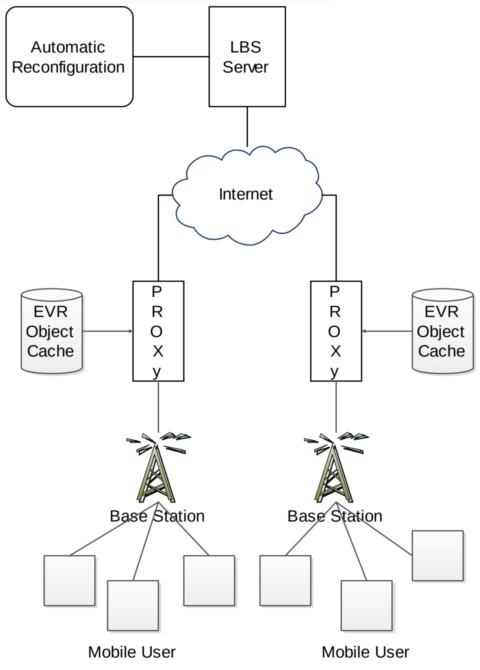
B. Hardware Requirement

Processor	: Pentium dual core
RAM	: 1 GB
Hard Disk Drive	: 80 GB
Monitor	: 17" Color Monitor

V. V. SYSTEM DESIGN

A. System Architecture

The proposed system architecture for NN and window query processing. The system architecture consists of three parts: 1) an external LBS server, 2) deployed proxies, and 3) the mobile clients. The LBS server is responsible for managing static data objects and answering the queries submitted by the proxies. Note that the LBS server can use any index structure (e.g., R-tree or grid index) to process spatial queries. The LBS server is assumed not to provide VRs. Each of the deployed proxies supervises one service area and provides EVRs of NN queries and EWVs (vector form of EVRs) of window queries for mobile clients in the service area. Each base station serves as an intermediate relay for queries and query results between mobile clients and the associated proxy. Base stations, proxies, and the LBS server are connected by a wired network. A mobile client maintains a cache to store the query results and the corresponding EVRs. When a mobile client has a spatial query, the mobile device first examines whether the current location is in the EVR of the stored result. If so, the stored result remains valid and the mobile device directly shows it to the client. Otherwise, the mobile device submits the query, which is received and then forwarded by the base station, to the proxy. For the received query, the proxy will return the query result as well as the corresponding EVR to the client. For example, the object info of a restaurant includes the cuisine type, hours of operation, phone number, and so on. When the EVR of p is inserted into the EVR-tree, the proxy updates the eID of the corresponding object entry.



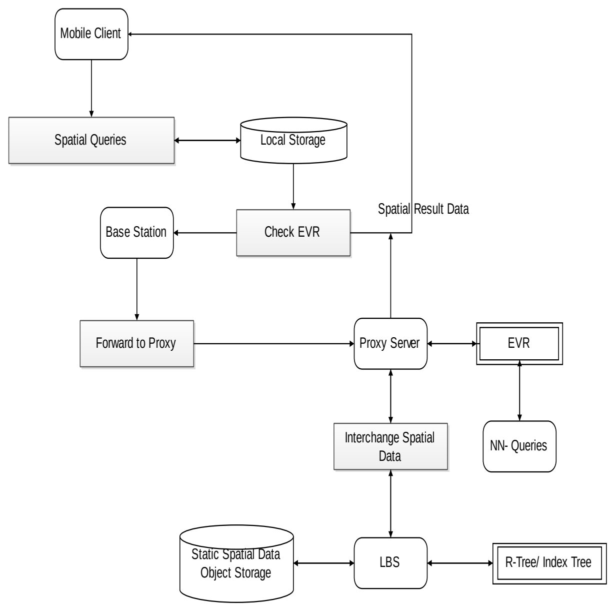
B. Proxy Design

A proxy builds EVRs of NN queries and EWVs of window queries based on NN query history and available data objects, respectively. The proxy maintains an object cache and two index structures: an EVR-tree for NN queries and a grid index for window queries. The two index structures share the data objects in the object cache. The EVR-tree is an R-tree (or its variants) composed of EVRs where each EVR is wrapped in a minimum bounding box (MBR). An EVR consists of the region vertices with respect to a data object and a pointer to the corresponding object entry in the object cache. When an NN query point q is located in an EVR of the EVR-tree, the proxy retrieves the corresponding object from the object cache to answer the query. On the other hand, the service area is divided into m _ n grid cells managed by the grid index. Grid cells are classified into two categories: fully cached cells and uncached cells. All grid cells are initialized to uncached. The proxy marks a cell as fully cached when all the objects within the cell are received. The corresponding grid index entry of a fully cached cell caches the object pointers to the associated object entries in the object cache. The purpose of fully cached and uncached cells is to realize the stored object distribution, enabling the proxy to create EWVs of window queries effectively. When receiving a window query, the proxy obtains the result and creates the corresponding EWV by retrieving stored objects in the surrounding fully cached cells. Although the EVR-tree and the grid index are designed for NN and window queries, respectively, these two index structures mutually support each other. For example, the object info of a restaurant includes the cuisine type, hours of operation, phone number, and so on. When the EVR of p is inserted into the EVR-tree, the proxy updates the eID of the corresponding object entry. Similarly, once the covering cell where p is located becomes fully cached, the cell flag is set to true by the proxy. With the information, when p has to be replaced, the proxy can remove the corresponding EVR from the EVR-tree or revert the fully cached cell to uncached. It is worth noting that, when a fully cached cell is reverted to uncached, the proxy sets the cell flag of all relevant object entries to false without removing the objects from the object cache for better performance because the objects are shared with the EVR-tree.

C. Data Flow Diagram

A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design).On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

A DFD provides no information about the timing or ordering of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored (all of which are shown on a DFD)

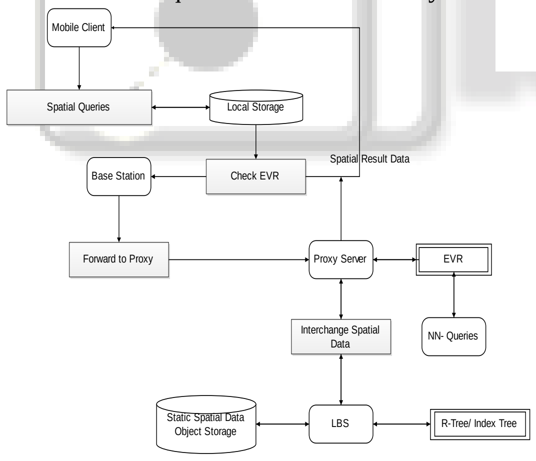


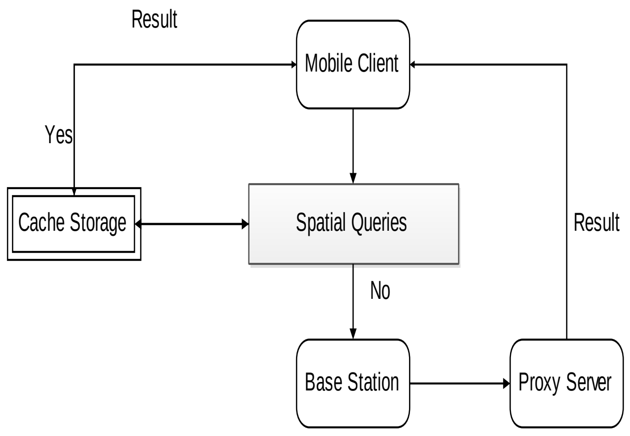
VI. SYSTEM IMPLEMENTATION

A. Module Description

1) Search Query Process Module

The mobile device submits the query, which is received and then forwarded by the base station, to the proxy. For the received query, the proxy will return the query result as well as the corresponding EVR to the mobile client.The mobile device submits the Continues query to proxy. A large number of queries submitted by mobile clients





2) Tree Creation Module

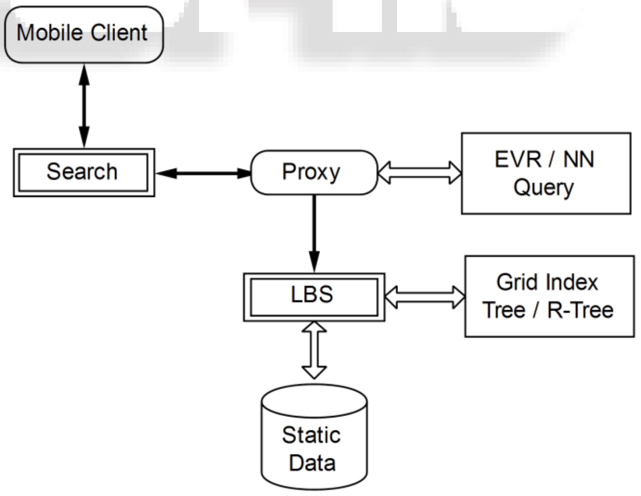
In this module build proxy server to estimate EVRs of NN queries and EWVs of window queries based on NN query history and available data objects. The LBS server can use any index structure e.g., R-tree or grid index to process spatial queries. The proxy maintains an object cache and two index structures: an EVR-tree for NN queries and a grid index for window queries. The two index structures share the data objects in the object cache.

a) EVR-Tree Generation For Nn

The EVR-tree is an R-tree (or its variants) composed of EVRs where each EVR is wrapped in a minimum bounding box (MBR). An EVR consists of the region vertices with respect to a data object and a pointer to the corresponding object entry in the object cache. When an NN query point q is located in an EVR of the EVR-tree, the proxy retrieves the corresponding object from the object cache to answer the query.

b) Generation of Grid Cell For Window Queries

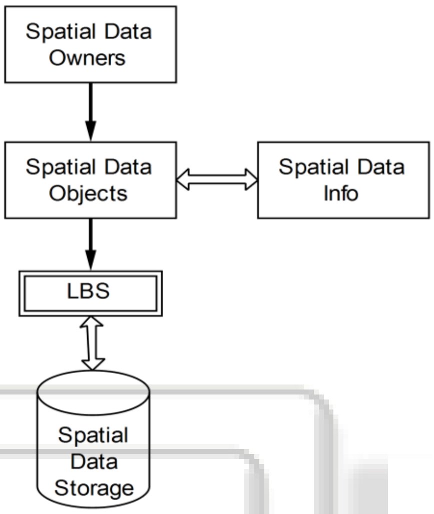
Grid cells are classified into two categories: fully cached cells and uncached cells All grid cells are initialized to uncached. The proxy marks a cell as fully cached when all the objects within the cell are received. The corresponding grid index entry of a fully cached cell caches the object pointers to the associated object entries in the object cache. The purpose of fully cached and uncached cells is to realize the stored object distribution, enabling the proxy to create EWVs of window queries effectively. When receiving a window query, the proxy obtains the result and creates the corresponding EWV by retrieving stored objects in the surrounding fully cached cells .



3) Object Storage Module

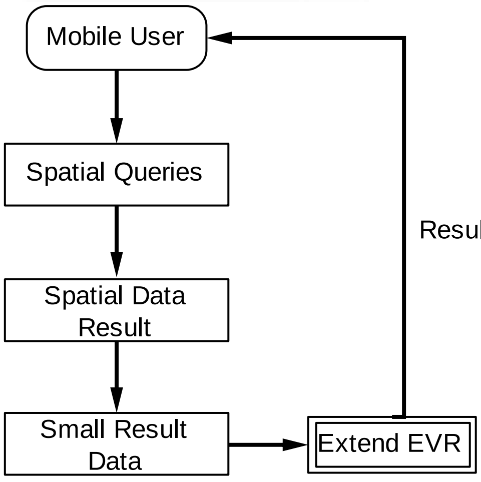
Location-based services (LBS) are a general class of computer program-level services used to include specific controls for location and time data as control features in computer programs. As such LBS is an information service and has a number of uses in social networking today as an entertainment service, which is accessible with mobile devices through the mobile network and which uses information on the geographical position of the mobile device. This has become more and more important with the expansion of the smartphone and tablet markets as well. LBS are used in a variety of contexts, such as health, indoor object search,entertainment, work, personal life, etc. LBS include services to identify a location of a person or object, such as discovering the nearest banking cash machine (*a.k.a.* ATM) or the whereabouts of a friend or employee. LBS include parcel tracking and vehicle tracking services. LBS can include mobile commerce when taking the form of coupons or advertising directed at customers based on their current location. They include personalized weather services and even location-based games. They are an example of telecommunication convergence.

The LBS server is responsible for managing static data objects and answering the queries submitted by the proxies.



4) Sharing-based nearest neighbor query Module

The sharing-based nearest neighbor query Module provides a rendering of the verification process of a sharing-based NN query in a step-by-step manner. Users can arbitrarily select a mobile host and launch a location-based NN query within region.



B. Algorithm

Algorithms/Techniques used:

This project mainly use for several techniques these are R Tree algorithm, Grid index algorithm, and Melkman's algorithm.

1) R Tree algorithm

R-trees can be more efficient for data storage and speed at search execution time, though they are generally tied to the internal structure of a given data storage system.

R-trees are tree data structures used for spatial access methods, i.e., for indexing multi-dimensional information such as geographical coordinates, rectangles or polygons. A common real-world usage for an R-tree might be to store spatial objects such as restaurant locations or the polygons that typical maps are made of: streets, buildings, outlines of lakes, coastlines,

2) Grid index algorithm

The individual cells of a grid system can also be useful as units of aggregation, for example as a precursor to data analysis, presentation, mapping, etc.

A grid index is a used for spatial indexing purposes. A wide variety of such grids have been proposed or are currently in use, including grids based on "square" or "rectangular" cells, triangular grids or meshes, hexagonal grids, grids based on diamond-shaped cells, and possibly more. The range is broad and the possibilities are expanding.

3) Melkman's algorithm

The Melkman's algorithm to compute the convex polygon of the updated EVR to remove the unnecessary vertices and achieve a larger region size. The convex polygon serves as the final updated EVR.

VII. CONCLUSIONS AND FUTURE WORK

The proxy takes advantage of spatial and temporal locality of spatial queries to create EVRs of NN and window queries. Different from prior work, we devised new EVR creation and extension algorithms for NN queries, enabling the proxy to build effective EVRs efficiently. The devised algorithms make the proxy achieve high performance even when the cache size is small. On the other hand, we propose to represent EVRs of window queries in the form of vectors, called estimated window vectors, to achieve larger estimated valid regions. Moreover, due to distinct characteristics, we introduce an EVR-tree and a grid index to process NN and window queries, respectively. The algorithms for mutual support of the EVR-tree and the grid index are developed to further enhance the system performance. The experimental results show that the proposed approach significantly outperforms the existing proxy-based approaches since our proposed algorithms create much larger EVRs for mobile clients. Compared with the representative server-based approach, the experimental results indicate that the proposed proxy-based approach achieves similar performance even though the proxy has only partial information of data objects. Besides, the results reveal that the proposed proxy based approach is suitable in a densely populated area, whereas the server-based approach is suitable when mobile clients move at high speeds. Although offering the above benefits, the inherent problem of data object updates needs further investigation for the proposed proxy-based approach.

The experimental results show that the proposed approach significantly outperforms the existing proxy-based approaches since our proposed algorithms create much larger EVRs for mobile clients. In future work, we will investigate the impact of data object updates on the proposed approach and extend the proposed approach to efficiently handle frequent objectupdates.

REFERENCES

[1] D. Lee, B. Zheng, and W.-C. Lee, "Data Management in Location-Dependent Information Services," IEEE Pervasive Computing, vol. 1, no. 3, pp. 65-72, July-Sept. 2002.

[2] B. Zheng, J. Xu, and D.L. Lee, "Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments," IEEE Trans. Computers, vol. 15, no. 10, pp. 1141- 1153, Oct. 2002.

[3] B. Zheng and D.L. Lee, "Processing Location-Dependent Queries in a Multi-Cell Wireless Environment," ProcSecond ACM Int'l Workshop Data Eng. for Wireless and Mobile Access, 2001.

[4] B. Zheng, J. Xu, W.-C. Lee, and D.L. Lee, "On Semantic Cachingand Query Scheduling for Mobile Nearest-Neighbor Search,"Wireless Networks, vol. 10, no. 6, pp. 653-664, Dec. 2004.

[5] X. Gao and A. Hurson, "Location Dependent Query Proxy," Proc. ACM Int'l Symp. Applied Computing, pp. 1120-1124, 2005.

[6] X. Gao, J. Sustersic, and A.R. Hurson, "Window Query Processingwith Proxy Cache," Proc. Seventh IEEE Int'l Conf. Mobile DataManagement, 2006.

[7] K.C. Lee, J. Schiffman, B. Zheng, and W.-C. Lee, "Valid ScopeComputation for Location-Dependent Spatial Query in MobileBroadcast Environments," Proc. 17th ACM Conf. Information andKnowledge Management, pp. 1231-1240, 2008.

[8] K.C.K. Lee, W.-C. Lee, H.V. Leong, B. Unger, and B. Zheng,"Efficient Valid Scope for Location-Dependent Spatial Queries inMobile Environments," J. Software, vol. 5, no. 2, pp. 133-145, Feb.2010.

[9] 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, vol. 51, no. 10, pp. 1124-1140, Oct. 2002.

[10] Y. Cai, K.A. Hua, and G. Cao, "Processing Range-MonitoringQueries on Heterogeneous Mobile Objects," Proc. Fifth IEEE Int'lConf. Mobile Data Management, pp. 27-38, 2004.