Prioritized Sequencing for Efficient Query on Broadcast Geographical Information in Mobile-Computing

Jianting Zhang Le Gruenwald

The University of Oklahoma, School of Computer Science, Norman, OK, 73019

Contact author email: ggruenwald@ou.edu, Phone: 1-405-325-3498

ABSTRACT

Broadcasting is an effective and efficient way to disseminate public geographical information to massive users. In this paper, we use R-tree to index broadcast geographical data. Based on our observation that geographical data and their accesses are often concentrated (clustered) in real applications, we propose a novel prioritized sequencing method by considering access frequencies based on application semantics to reduce average access time (latency). The key idea is to put "hot" data items ahead of "cold" ones in the broadcast sequence while still maintain R-Tree indexing structure.

We evaluate our method using MapInfo Census 2000 data sample with 576 service locations and 111 of them are hot data items. The average reduction percentages of access time are reduced by 60% for both point queries and range queries based on 10 rounds experiments, each consists 10 hot data items and 5 cold items that are randomly selected. We conclude that our simple method is very effective in improving query efficiency in geographical information broadcast systems.

We also proved the linear relationship between wait time and the Aggregate Data Affinity (ADA) (Lee, 2000), a parameter measuring the "eagerness" for data experienced by a client during a query process in broadcast wireless mobile computing environment. Thus our conclusion also holds when using ADA for evaluating broadcast system performance

Categories and Subject Descriptors

H.2.4 **[Database Management]**: Systems

General Terms

Algorithms, Management, Measurement, Performance, Design

Keywords

Prioritized Sequencing, Geographical Information, Query Processing, Data Broadcast, Mobile Computing

1. INTRODUCTION

Most current research on mobile computing assumes point-to-point communication. This scheme suffers from scalability problem and is inefficient when many clients request the same data at the same time. Public geographical information, such as service locations and traffic conditions are playing important roles in our everyday life. As Internet services go from fixed networks to wireless mobile networks, services of these kinds of information will likely to be major consumers of wireless resources. It is our vision that digital broadcasting of public geographical information over wireless mobile network will be an important means in building intelligent urban information infrastructure that can greatly reduce overall wireless resource consumption and provide cheaper and better services.

First, broadcasting is an efficient and effective way to make information accessible to a large quantity of users because of its well-known excellent scalability (Imielinski, 1997). This is especially important due to the extremely limited nature of wireless resources. Second, as far as end user is concern, broadcasting is more economically viable than establishing exclusive point-to-point connection between clients and servers in terms of bandwidth available and price rate. One additional reason is that broadcasting is distributed in nature and can provide good services for local users. On the other hand, servers in a point-to-point communication architecture might be far way from clients and thus the services provided are more expensive. Third, much of the geographical information changes infrequently and is mostly read-only, thus is especially suitable for broadcasting. Finally, Digital broadcasting has significant advantages over traditional analog voice broadcasting, such as automatic information retrieval instead of having to tuning to a channel all the time by a user. It is possible to integrate geographical and non-geographical information by planning broadcast carefully and using smart receivers when all the broadcast information involved is in digital format.

Different from disk or memory based data processing that allows random access to data storage media, broadcasting sequence is one dimensional in nature and only allows sequential access. At the same time, mobile devices at the client side have significant constraints in terms of energy, storage and computation power. Data organization at the server side and query processing at the client side are two important issues in a data broadcast system. In this paper, we present a prioritized sequencing method to efficiently query broadcast geographical information for personal mobile computing.

The rest of the paper is arranged as follows. Section 2 states the problem and our motivation. Section 3 presents our prioritized sequencing method. Section 4 is experimental studies using real data. Section 5 gives a brief overview of related work. Finally, section 6 is conclusion and future work directions.

2. PBROBLEM AND MOTIVATION

Broadcasting can be categorized into two main categories, namely pull-based broadcasting and push-based broadcasting. Our application belongs to the later one because geographical data are usually public information and the system is designed to disseminate them to a large volume of users without explicit requests. It is extremely difficult, if not impossible, to schedule broadcasting for the number of users at such a scale.

An introduction as well as several indexing approaches for push-based data broadcasting is presented in (Imielinski, 1997). There are two most important parameters in measuring the performance of broadcast schemes namely Tune In Time (TT) and Access Time (AT) or latency. Broadcast data indexing plays an important role in the tradeoff between them. In this paper, we do not intend to develop new indexing techniques that are better for data broadcasting nor to evaluate which of the existing techniques are more suitable. Rather, our focus is to incorporate application semantics into indexing techniques to achieve better performance. We consider Location Dependent Queries (LDQ) (Seydim, 2001) against broadcast geographical information. Our motivation is from the following simple example.

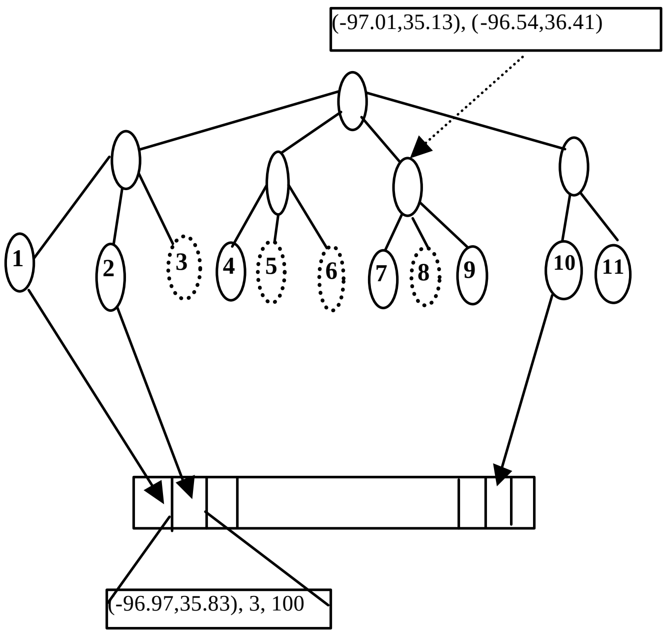

Fig. 1 A Simple Illustrative Example

Fig. 1 represents an R-Tree (Guttman, 1984) for indexing geographical data. The dotted and numbered circles represent leaf nodes pointing to frequently accessed data items (or hot data items), solid circles with numbers represent leaf nodes pointing to less frequently accessed data items (or cold data items) and non-numbered circles represent internal nodes. An internal node has several branches in the format of (MBR, Pointer) where MBR (Minimum Bounding Rectangle) consists of the x and y coordinates of the upper and the lower corner points. Each branch of leaf nodes point to a real data record, such as a restaurant record contains its geometric coordinates in the longitude/latitude format (-96.97,35.83), service level (3 star) and available seats (100).

The conventional broadcasting sequence would be [1,2,3,4,5,6,7,8,9,10,11] by traveling the leaf nodes of the R-Tree. Suppose [3,5,6,8] are hot data items. If we put the hot data before the rest data items then the sequence would be [3,5,6,8, 1,2,4,7, 9,10,11]. If we query data item 3, in the conventional sequence the access time is 3 while it would only be 1 in the prioritized sequence. Another example is to query data item 8. In the conventional sequence, the access time is 8 while it would only be 4 in the prioritized sequence which is a significant saving.

3. R-TREE BASED PRIORITIZED SEQUENCING

For geographical data, in many cases, both data content and data utilization is highly concentrated. For example, there are more gas stations and restaurants along highways. On the other hand, users are more likely to query on these service locations when they are driving on highways. We call data items that are accessed more often as "hot" data items. From the simple example illustrated above, we can see that giving "hot" data items more priority can improve system performance. We use R-Tree for geographical data indexing primarily because of its popularity in practice (Oracle Spatial [HREF1] for example).

3.1 Algorithm for Generating Prioritized Sequence

The algorithm for generating a prioritized broadcast sequence based on R-Tree explore four data structures: an R-Tree to store the index of the whole data set; a hash table to store the IDs of all hot data items and two vectors to store the sequences of IDs of hot and cold data items respectively. The algorithm consists of the following steps:

1. Generate a hot data item set based on application semantics. For point data along a linear object (such as highway), the hot data item set can be defined as the points that fall into an area within a user-defined radius called a buffer of the linear object .
2. Build a R-Tree for the whole data item set and a hash table for the IDs of hot data items.
3. Travel the leaves in the R-Tree. For each leaf node:
   - o Lookup the ID of the item in the hash table for the hot data item set.
   - o If found then put the item ID in the vector of hot items.
   - o Else put the item ID in the vector of cold items.
4. Output the contents of items in the hot data vector followed by the ones in the cold data vector.

3.2 Performance Analysis

We compare our method with an intuitive broadcasting method that sequencing data items by traveling the leaf nodes of the R-Tree. We call this method BD-INT and our proposed method BD-PRI. In BD-PRI, we do not change the structure of the R-tree index. Rather we only change the sequence of data items and their corresponding pointers in the R-Tree leaf nodes. Thus we do not change the tune in time. For latency, we do not change probe wait (i.e. time waiting for index block) either compared to BD-INT since both of them need to access the index block first. The major difference between BD-PRI and BD-INT is the average duration between the point the index is encountered and the point the required records are downloaded (bcast wait). In the following analysis, for the sake of simplicity, we assume the starting point for calculating latency is the end of index block and all the data items are of unit length.

As in web searches, often only a partial list of LDQ results is really needed. The client may stop the query at any time after he/she feels the retrieved data items are sufficient. For example, a client looking for a used car at Yahoo.com might only need to browse the first 20 used cars in the first retrieved page while there might be 10 pages of cars meet his/her search criteria. It is very likely that all the data items the client wants are in the hot data item set. If not, the cold data item set is explored subsequently. Assume all the required data are in the hot data set, then it is easy to see that BD-PRI reduces bcast time from $(L_{ind}+Lw)/2$ to $(L_{ind}+L_h)/2$ where L_{ind} is the length of the index blocks, L_w is the total length of the whole data items and L_h is the total length of the hot data items. If the required data items include both "hot" and "cold" ones, it is possible that BD_PRI has greater latency than BD_INT for cold data items due to the prioritized sequencing scheme. However, since hot data items are accessed more frequently, the average latency is reduced. The greater the ratio of access frequencies of hot data items to cold ones, the greater average latency reduction. In the running example shown in Fig. 1, the average latency for retrieving data items 1 and 8 is reduced from 5.5 to 2.5, more than 55% reduction.

Two other parameters called data affinity index and aggregate data affinity are adopted from (Lee, 2000) to measure the "eagerness" for data experienced by a client Intuitively, as client browsing through a partial list of search results, his/her eagerness for information might drop significantly and eventually decides to stop accepting more results. Data Affinity Index (DAI) is defined as $A_i=(1-N_i/N)$ where Ni is the total number of data items received so far and N is the number of total data items in the query results. The Aggregate Data Affinity (ADA) is defined as the summation of A_i over the process of receiving data items. In the running example shown in Fig.1, let us include a cold data item in the query result. Suppose the query result consists of data items 6 and 7. For the whole broadcast cycle, from time slice 1 to 11, the DAIs for BD-INT are 1, 1, 1, 1, 1, 0.5, 0.5, 0, 0, 0, 0 and DAIs for BD-PRI are 1, 1, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0, 0, 0 respectively. Thus the aggregate data affinity is reduced from 6 to 5. If the client decides to stop receiving data items at time slice 5, then the aggregate data affinity is further reduced from 5 to 2.5. In Fig. 2, we have proved that ADA (A) has a linear relationship with bcast wait time (T) as A=T/N. Due to the linear relationship between the two parameters, we only use latency for evaluation in our experiments and make similar conclusion on ADA.

Note that, in multidimensional indexing and query, data items in the query results are not necessarily consecutive. They might fall in different branches from the root to the leaves in the multidimensional index tree. In this case, both average latency and ADA are expected to be larger in the multidimensional case than in the one-dimensional case.

3.3 Algorithm Complexity

In this subsection we analyze the time complexity of the proposed BD-PRI algorithm. We do not include the costs required to build the R-Tree in the comparison since it is needed by both BD-INT and BD-PRI as discussed before. We only discuss the extra overhead in achieving improvements by prioritized sequencing. The overhead of the proposed method is very low. The major overhead is to traverse the R-Tree and lookup on the hash table to identify whether an item is hot or not.

Assume clients begin to tune in the broadcast channel at time 0. Suppose there are N items in the query result, each is accessed at time t_i. Then the total wait time is obviously $T=\sum t_i$ ($i=1\ldots N$; $t_0=0$). On the other hand, according to the definition of ADA (A) we have the following equations:

$$A=\sum A_i=\sum_{i=0}^{N-1}(1-\frac{i}{N})*(t_{i+1}-t_i)$$

$$=\sum_{i=0}^{N-1}(t_{i+1}-t_i)-\sum_{i=0}^{N-1}\frac{i}{N}*(t_{i+1}-t_i)$$

$$=(t_N-t_0)-\frac{1}{N}[1*(t_2-t_1)+2*(t_3-t_2)+....+(N-1)*(t_N-t_{N-1})]$$

$$=(t_N-t_0)-\frac{1}{N}(-t_1-t_2-t_3...+N*t_N)$$

$$=(\frac{1}{N}*\sum t_i)-t_0$$

$$=\frac{T}{N}$$

Fig. 2 Proof of Relationship between ADA and Wait Time

Assume the number of data items to index is N and the minimum number of branches in each node is M, then the height of the R-tree is at most $H=\lceil \log_M(N)\rceil-1$ (Guttman, 1984). The maximum number of internal nodes is as follows:

$$\mathrm{I}=1+\mathrm{M}+\mathrm{M}^2+\ldots+\mathrm{M}^{\mathrm{H}}=\frac{\mathrm{M}^{H+1}-1}{\mathrm{M}-1}=\frac{\mathrm{N}-1}{\mathrm{M}-1}$$

, thus I is in the order of O(N/M).

Since the number of leaf nodes of the R-tree is N, the time complexity of traversing the whole R-Tree is O(N/M)+N=O(N). Since looking up a hash table is O(1) using a reasonable good quality hash function, the total time complexity of the hash table lookup is also O(N). Thus the overall time complexity of the proposed method is O(N). Given the rich computation power at the server side compared to the stringent access time requirement at the client side, the overhead of the proposed method is negligible. Indeed, our focus is to find out how well the proposed approach is in reducing latency at the client side that is resource constrained. We will evaluate the performance by experiments using real data in the following section.

4. EXPERIMENTS AND RESULTS

We use a data set from the MapInfo census 2000 data samples ([HREF 2]). There are 586 points in the area representing service locations, such as hospitals and parks. We first build 0.5 mile buffers for the highways in the study area, i.e., areas within 0.5 miles radius of the highways. We then select all the points that falls into the buffer as the hot data items, which results in 111 points (Fig. 4), and treat the rest as cold data items. Fig. 5 shows the R-tree built for the area. Exact point query and spatial range query are two most frequently used query types for spatial information (Gaede, 1998). We perform two groups of tests for each of them respectively.

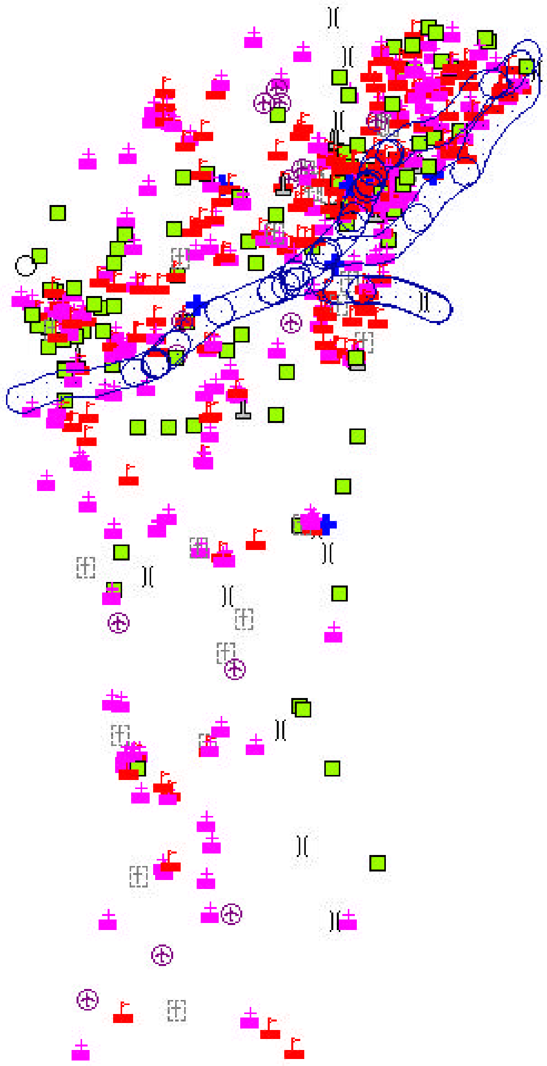

Fig. 3 Distributions of service locations in the study area

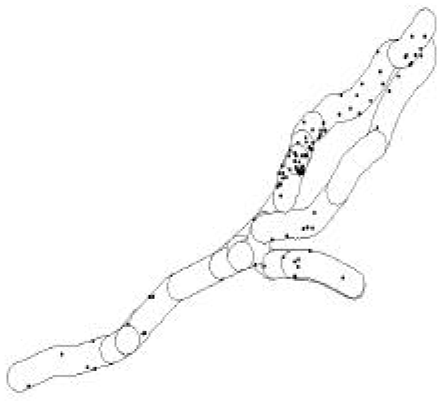

Fig. 4 Buffers (0.5 mile radius) from highways and their associated hot locations

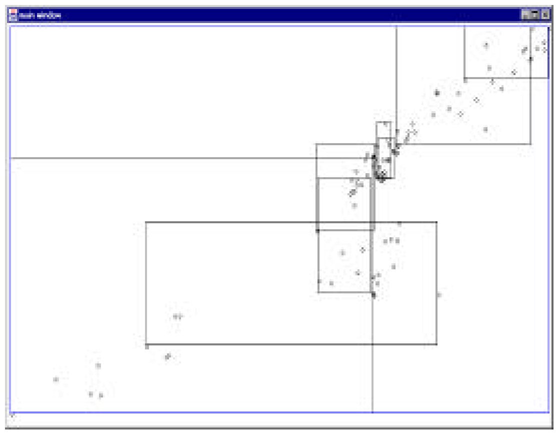

Fig 5 Visualization of the R-Tree of the service locations in the study area

4.1 Point Query Result

For a point query, we know the exact location of the data item to query on. An example of such a query is the following:

```
Select nLevel, nSeats
from Restaurant
where location=(-96.97,35.83)
```

In our test, we randomly select 10 points from the hot data items and 5 points from the cold data items. For each point query, we compute the latency for both BD-INT and BD-PRI. The test result is shown in Table 1. From the result, we can see that latency for the 10 hot data items are reduced by 2581 time units at the expense of an increase of 393 time units for cold data items. The overall improvement percentage is 52.24%.

We perform this group of tests for 10 times. The improvement ranges from 49% to 64% with an average of 57.65%. From the test, we can safely say that the average latency is reduced by more than a half in BD-PRI for point queries.

Table 1. Result of the for Point Query Test

Data Item ID	Type	Latency (BD-PRI)	Latency (BD-INT)	Latency Improv.	Latency Improv. (%)
14	Hot	61	381	320	83.99%
375	Hot	52	548	496	90.51%
534	Hot	27	405	378	93.33%
316	Hot	45	219	174	79.45%
24	Hot	49	217	168	77.42%
414	Hot	68	225	157	69.78%
244	Hot	66	384	318	82.81%
277	Hot	10	94	84	89.36%
166	Hot	8	438	430	98.17%
56	Hot	84	140	56	40.00%
20	Cold	346	273	-74	-27.11%
443	Cold	398	333	-65	-19.52%
379	Cold	152	53	-99	-186.79%
41	Cold	388	323	-65	-20.12%
407	Cold	247	157	-90	-57.32%
Avg		133.4	279.3	145.9	52.24%

4.2 Range Query Result

For a range query, given a query center and a query range, we want to find all the points that fall in the area defined by these two parameters. The region could be either a circle or a square. Since distance calculation to decide whether points fall inside a circle is more expensive than coordinate comparison to decide whether points fall inside a square, we use the later approach in considering compute resource constraints at the mobile client side. An example of such query is the following:

```
Select nLevel, nSeats
from Restaurant
where
(
(location.x >=center.x –range)
 and  (location.x <=center.x +range)
 and (location.y >=center.y –range)
 and (location.y <=center.y +range)
)
```

Unlike the point query result which consists only one data item, the range query result might consist of multiple (including 0) data items (possibly include both hot and cold data items). For demonstration purposes, in each test, we randomly select 10 hot data items and 5 cold data items and use their locations as the centers in the test. The ranges are determined as follows: for hot data items the range is a random number between 0.2 miles and 0.5 miles; for cold data items the range is a random number between 0.5 miles and 1 mile. The reason for using different ranges for hot and cold data items is to have the approximately the same number of data items in the query result. The range values used here are quite arbitrary since they are highly application dependent.

A test result is shown in Table 2. From the result, we can see that the weighted average latency is reduced from 355.65 to 133.3 and the reduction percentage is 60.29%. As in point queries, we also perform the test for range queries 10 times. The overall reduction percentage of latency is 56.86%.

Table 2. Result of the Range Query Test

Item ID (Center Point)	Range (mile)	#Data Items in Results	Latency (BD-PRI)	Latency (BD-INT)	Latency Improve-ment.	Latency Improve-ment(%)
212	0.24	2	126	759	633	83.40%
294	0.50	1	67	376	309	82.18%
100	0.29	3	122	1386	1264	91.20%
24	0.29	1	49	217	168	77.42%
404	0.29	7	1033	3642	2609	71.64%
360	0.43	1	23	13	-10	-76.92%
294	0.34	1	67	376	309	82.18%
534	0.25	4	85	1510	1425	94.37%
392	0.25	1	53	241	188	78.01%
98	0.46	1	2	74	72	97.30%
151	0.37	4	405	1002	597	59.58%
356	0.64	1	234	144	-90	-62.50%
478	0.33	6	1211	2416	1205	49.88%
346	0.39	3	573	294	-279	-94.90%
479	0.76	4	1282	976	-306	-31.35%
Avg	0.39	2.67	133.3	355.65	202.35	60.29%

5. Related Work

A comprehensive overview of multidimensional indexing and access methods is presented in (Gaede, 1998). The original R-Tree method is proposed by (Guttman, 1984). Two R-Tree variations, namely R+-Tree (Sellis, 1987) and R*-Tree (Beckmann, 1990) are introduced later. There are several implementations of R-Tree indexing techniques available ([HREF 3], [HREF 4]). Our experiment is based on ([HREF 4]) with necessary modifications and extensions.

A good introduction of data broadcasting is presented in (Imielinski, 1997). The authors also propose algorithms for multiplexing clustering and non-clustering indexes along with data on a broadcast channel. However, their work only takes one-dimensional tree indexing (B-tree) into consideration. For the multiple-attributes case, they propose to build multiple indices for each interval of the first attribute. This is actually using one-dimensional indexing methods consecutively for multidimensional indexing which is inefficient (Kriegel, 1984). In our study, we apply multi-dimensional R-tree indexing for geographical data.

(Hambrusch, 2001) studies the execution of spatial queries on broadcast tree-based spatial index structures. Their work assumes the client has very limited memory that the whole R-tree cannot fit into the client memory and the client has to discard some retrieved R-Tree nodes to hold more useful ones during the query process. Their work focuses on reducing extra access time incurred by having to access multiple broadcast cycles due to the replacement. We do not have such a strict requirement and we assume the whole R-Tree can fit into the client's memory. Our focus is how to reduce access time within a broadcast cycle by considering access frequencies. Since memory is becoming cheaper and a 16M configuration is now a standard, we believe our assumption is more realistic.

There are several works on handling access frequencies in broadcast data organization. (Shivakumar, 1996) proposes a method based on alphabetic Huffman coding that handles access frequency and keeps key ordering for fast search. However, the method only works for one-dimensional and categorical data and cannot be applied for geographical data. Further more, access frequency in our application is a "semantic" one rather than accurate statistics. Since it is generally hard to find accurate statistics of access frequencies in a push-based broadcast system, we believe our method is more practical than Huffman coding tree based methods in handling access frequencies.

Broadcast disks (Acharya, 1995) based methods have been widely used for broadcast data scheduling for items with different access frequencies, however generally broadcast disk method itself does not involve indexing. (Hu, 2001) proposes to build index for each minor cycle in broadcast disks but it works for one attribute only.

6. CONCLUSION AND FUTURE WORK

We proposed a simple yet effective prioritized sequencing method for reducing total access time (latency) in R-Tree based geographical data broadcasting. Based on our experiments using the real MapInfo Census 2000 sample data set, the total access time is reduced by 60% in both point queries and range queries. Since we have proved the linear relationship between total access time and Aggregated Data Affinity (ADA), the same conclusion can be applied to ADA as well.

Our method essentially provides progressive data access in wireless broadcast channels that is very similar to progressive multimedia transfer in web-based applications. Since progressive data transfer has been proved to be effective and accepted by users, we are expecting our method to have a similar good performance in wireless mobile computing.

For future work, we will consider moving continuous LDQ against broadcast geographical information. We will also consider semantic caching of retrieved data and index based on the client's mobility, such as speed and direction. A further challenge is to consider queries against broadcast spatial and temporal information for moving objects.

7. REFERENCES

[1] S.Acharya, R.Alonso, M.Franklin, S.Zdonik, Broadcast disks: data management for asymmetric communication environments, ACM SIGMOD Conference, 1995:199-210

[2] N.Beckmann, H.-P. Kriegel, R.Schneider, B.Seeger,The R*-tree: An efficient and robust access method for points and rectangles, ACM SIGMOD Conference, 1990:322-331

[3] V.Gaede, O.Günther ,Multidimensional access methods, ACM Computing Survey, 30(2), 1998:170-231

[4] A.Guttman, R-trees: A dynamic index structure for spatial searching, ACM SIGMOD Conference, 1984:47-54

[5] S. Hambrusch, C.-M. Liu, W. Aref, S. Prabhakar, Query Processing in Broadcasted Spatial Index Trees, 7th International Symposium on Advances in Spatial and Temporal Databases (SSTD), 2001:502-521

[6] Qinglong Hu, Wang-Chien Lee, Dik Lun Lee: A Hybrid Index Technique for Power Efficient Data Broadcast, Distributed and Parallel Databases 9(2), 2001:151-177

[7] *T.Imielinski, S Viswanathan, B.R.Badrinath,* Data on air: organization and access**,** IEEE Transactions on Knowledge and Data Engineering, 9(3), 1997:353 –372

[8] H.-P.Kriegel, Performance comparison of index structures for multikey retrieval, SIGMOD Conference, 1984:186-196

[9] Ken Lee, Hong Va Leong, Antonio Si: A Semantic Broadcast Scheme for a Mobile Environment based on Dynamic Chunking, the 20th International Conference on Distributed Computing Systems (ICDCS), 2000: 522-529

[10] T. Sellis, N. Roussopoulos and C. Faloutsos. The R+-Tree: A Dynamic Index for Multi-Dimensional Objects, the VLDB Journal, 1987:507-518

[11] A.Y.Seydim, M.H.Dunham, V.Kumar: Location dependent query processing. the Second ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE), 2001: 47-53

[12] N.Shivakumar, S.Venkatasubramanian, Efficient Indexing for broadcast based wireless systems, ACM Baltzer Mobile Networks and Applications **(**MONET), 1(4), 1996:433-446

[HREF 1] http://download-west.oracle.com /otndoc/oracle9i/901_doc/appdev.901/a88805/sdo_inde.htm

[HREF 2] http://www.mapinfo.com

[HREF 3] http://www.dbnet.ece.ntua.gr/~theodor/files/rtrees/

[HREF 4] http://www.cs.ucr.edu/~marioh/rtree/index.html