Efficient Placement of Geographical Data Over Broadcast Channel for Spatial Range Query Under Quadratic Cost Model

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
Data broadcasting is well known for its excellent scalability. Most geographical data, such as weather and traffic, is public information that has a large amount of potential users which makes it very suitable for broadcast. The query response time is greatly affected by the order in which data items are being broadcast. This paper proposes an efficient method to place geographical data items over broadcast channel that reduces access time for spatial range queries on them. This paper then performs evaluation studies comparing different ordering methods: 1000 random orderings, R-Tree traversal ordering, Hilbert ordering and the optimized ordering based on the proposed method. The results show that the optimized ordering is significantly better than the others.

Categories and Subject Descriptors
H.2.4 **[Database Management]**: Systems

General Terms
Algorithms, Management, Measurement, Performance, Design

Keywords
Data Broadcast, Geographical Information, Cost Model, Query Processing, Optimization, Mobile Computing

1. INTRODUCTION
Data broadcasting is well known for its excellent scalability (*Imielinski, 1997*). Most geographical data, such as weather and traffic information, is public and has a large amount of potential users. It is attractive to broadcast geographical data in metropolitan areas to reduce the increasing demands for wireless bandwidth resources. Furthermore, for users that are able to be aware of their locations by using Global Position Systems (GPS), network infrastructures or their combinations (Konig-Ries, 2002), they can perform Location Dependent Queries (LDQ) (Seydim, 2001) to request Location Dependent Services (LDS). It is easy to see that LDQ on broadcast geographical data over air is particular interesting in the context of large-scale resource-efficient data dissemination in mobile computing. Spatial range query (Rigaux, 2002) processing on broadcast geographical data will be one of the most popular ways to provide LDS.

MobiDE'03, September 19, 2003, San Diego, California, USA.

In a broadcast system, a minimum logical unit in a broadcast sequence is called a bucket/frame, and a set of continuous buckets (either index or real data) is called a segment. Different from main memory or disk resident data accesses, accesses to a broadcast sequence are essentially one-dimensional. There are two important parameters in evaluating the performance of a broadcast system, namely Tune-in Time (TT) and Access Time (AT, or latency). TT is the amount of time spent by a client listening to the broadcast channel. AT is the average time elapsed from the time a client requests data to the time when all the required data is downloaded by the client. In Fig. 1, TT is equal to the summation of the lengths of the required data items (shaded) while AT is the duration between the initial access position and the last required data item.

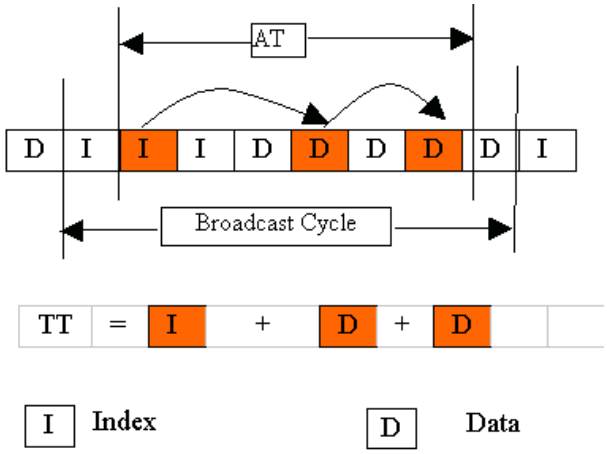

Fig. 1 Illustration of TT and AT

In (Imielinski, 1997), AT is the sum of the Probe Wait (PW) and the Bcast Wait (BW) where the former is the average duration for getting to the next index segment and the latter is the average duration between the time when the index segment is encountered and the time when all the required data items are downloaded. We argue that it might be more appropriate to divide AT into four components: Index-Probe Wait (IPW), Index-Bcast Wait (IBW), Data-Probe Wait (DPW) and Data-Bcast Wait (DBW). IPW is the same as PW. IBW is the time duration from the time when the first index segment is met to the time when the last index segment is met. DPW is defined as the duration from the time the last index segment is reached to the time when the first data segment is reached. DBW is defined as the duration from the time when the first data segment is reached to the time when the last data segment is downloaded. The summation of IBW, DPW and DBW is equivalent to BW. Using the four components allows us to compute access time to index and data separately. We assume a client has already had an ordered set of pointers to the data items in the broadcast channel by

performing a spatial range query on the index segments which are either in the same channel with the data or in a separate index channel. The scenario we consider is the one in which the index and data are broadcast using two separate channels where a client may begin access the data channel at any position. These four components under this scenario are illustrated in Fig 2.

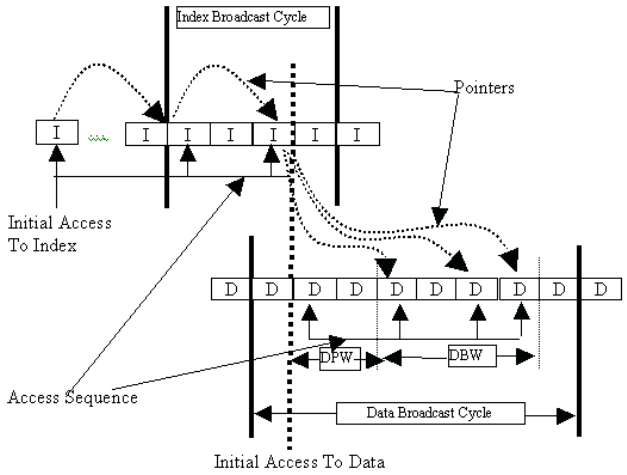

Fig. 2 The Four Components in Access Time for the Scenario that Index and Data Use Separate Channels

Spatial range query is a type of "Complex Query" (Lee, 2002) whose result set includes multiple data items. The query response time is greatly affected by the order in which data items are being broadcast. Suppose there are 6 data items {1,2,3,4,5,6} to broadcast and there are two data items {2,5} in a spatial range query result. It only takes two unit of time to retrieve the query result if data item 2 and 5 are placed next to each other. However, it would take 4 unit of time to retrieve them in natural ordering. The placement is complicated when there are multiple such complex queries with different access frequencies over broadcast data.

The objective of this paper is to provide an efficient and effective optimization method to generate broadcast sequences that reduce the access time for processing spatial queries on broadcast geographical data. The rest of this paper is arranged as follows. Section 2 reviews the related work. Section 3 presents the quadratic cost model and Section 4 provides the optimization method. Experiments using synthetic data sets are presented in Section 5, and finally Section 6 provides conclusions and future work directions.

2. RELATED WORK

Range queries are the most frequently used spatial queries and have been extensively studied in disk-resident data management research. Several cost models have been proposed for measuring the performance of spatial indexing on range queries (Pagel, 1993; Theodoridis, 1996; Theodoridis, 2000). The measurement used is the number of disk accesses which is equivalent to TT in broadcasting without considering paging and buffering effects. However, to the best of our knowledge, there is no previous work done on access time for spatial range queries on broadcast spatial data.

There have been several studies on general data broadcast. Many of them focus on indexing techniques to make tradeoffs between TT and AT, such as tree-indexing (Imielinski, 1994a), hashing (Imielinski, 1994b), signature (Lee, 1996) and hybrid (Hu, 2001a). These studies can support only queries on one-dimensional data and allow only one data item per access. Although (Imielinski, 1997) proposed to chain data items that have the same values in different meta-segments in its nonclustering index and multi-index methods, it cannot be applied to data items that have different values but are often in the same query results. Furthermore, in its performance analysis, it assumes that it takes a whole broadcast cycle to retrieve non-clustered data items of a particular value. That is an unnecessary overestimation. The issue of multi-attribute data broadcast and query was first addressed in (Hu, 2001b). However, this work can handle only conjunction/disjunction queries that involve fewer than three attributes. They are not suitable for spatial range queries on geographical data.

Recent works on object-oriented database broadcast (Chehadeh, 1999) and relational database broadcast (Lee, 2002) allow multiple data items to be accessed in a query. However, they assumed that accesses to data items have predefined orders. They are not suitable for spatial range queries since data items in a query result since the order of data items in a range query result is not important. The work presented in (Chung, 2001) is essentially similar to our cost model of data access time. However, it excludes TT from the access time for the items in the query result set which makes the total access time a summation of multiple quadratic terms. To simplify the result, it uses a linear function to approximate the quadratic cost, which renders the model inaccurate. Furthermore, its proof of the approximation is incorrect. None of the above cost models is designed for spatial range queries. We believe that the cost model we use, in which the access time for a single query is linear with respect to a single quadratic term (see Section 3 for details), is more concise and accurate.

From graph algorithms' perspective, sequencing graph nodes can be treated as a graph layout problem. A survey on graph layout problems was presented in (Daíz, 2002). A window-based vertex orderings with applications to circuit clustering was presented in (Alpert, 1996) where unordered vertices are iteratively added to the ordering based on their attractions to the previously ordered vertices. (Bar-Yehuda, 2001) presented a polynomial time algorithm for computing an optimal orientation (ordering) of a balanced decomposition tree for the graph linear arrangement problem. Although the theoretical approximation ratios were not improved, experiments showed good results. A multi-scale scheme for MinLA problem is presented in (Koren, 2002). Different from (Bar-Yehuda, 2001) which imposed global constraints on the ordering through the Binary Decomposition Tree (BDT), it imposed many local constraints restricting small sets of vertices throughout the entire multi-scale hierarchy.

The only previous work on geographical data broadcast we know is (Hambrusch, 2001). It studied the execution of spatial queries on broadcast tree-based spatial index structures. Their work assumes the client had very limited memory, the whole R-tree cannot be 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 focus on reducing extra access time incurred by having to access multiple broadcast cycles due to the discard and replacement. We assume that a client has already had the pointers to the data items in the data channel, either from another separate index channel or from the same channel that combines both the data and index. A client can sort the values of the pointers and thus only one scan of the data channel is sufficient to retrieve all the data items.

3. THE COST MODEL

To reduce the access time to the data broadcast channel, we first derive a cost model to compute the access time for spatial range queries over broadcast geographical data. It will be used in the optimization method presented in Section 4. The details of this cost model have been presented in our previous work (Zhang, 2003). Here we provide a summary of its results only which are given in the followings.

Let DS=[x_1,x_2)× [y_1,y_2) be the data space that defines all the geographical point data items. Suppose the range query window is (q_x,q_y). We define an Extended Region R_u of data item P_u as the rectangle of (q_x,q_y) centred at P_u. Let A_i be the area of R_i, $A_{i,j}$ be the intersection of area of R_i and R_j, … $A_{1,2...n}$ be the intersection area of R_1, R_2…R_n. Let $\widetilde{A}_i$ be the part of A_i query windows centered in which solely contains point P_i, $\widetilde{A}_{i,j}$ be the part of $A_{i,j}$ query windows centered in which solely contains points P_i and P_j, … $\widetilde{A}_{1,2...n}$ be the part of the intersection area of R_1, R_2...R_n that contains all the n points. Note $\widetilde{A}_{1,2...n} = A_{1,2...n}$.

We use hyper-graph to represent all possible spatial range query result sets. Given a set of nodes V, a hyper-edge in the hyper-graph consists of the nodes in a distinct spatial range query result set {n_1,n_2..n_k}. The access frequency of such a set, which is proportional to $\widetilde{A}_{n_1,n_2,...n_k}$, will be used as the weight of the corresponding hyper-edge.

Let function $\pi(u)$ map point u to its position in the broadcast sequence. Let L be the broadcast cycle length. For a single query result set contains k data items n_1, n_2 ... n_k, let L_2 denote the DBW of a query result set with a query window size of (q_x, q_y), formally,

$$L_2 = \max\{\pi(n_1), \pi(n_2),... \pi(n_k)\} - \min\{\pi(n_1), \pi(n_2),... \pi(n_k)\}$$

The average access time to data channel, i.e. from a client begin to access data channel to the time all the data items are downloaded (DPW+DBW) can be computed as follows and denoted as function g(L_2).

$$g(L_2) = \frac{1}{L}[L^2 - \frac{(L-L_2)^2 - (L-L_2)}{2}] :$$

The total data access time for a query window (q_x,q_y) can be written by summarizing the access time over all possible query result sets :

$$Cost^{(q_x,q_y)}$$

$$= \sum_{1\le i<j\le n} \widetilde{A}_{ij}^{(q_x,q_y)} * g(|\pi(i)-\pi(j)|)$$

$$+ \sum_{1\le i<j\le k\le n} \widetilde{A}_{i,j,k}^{(q_x,q_y)} * g(\max(\pi(i),\pi(j),\pi(k)) - \min(\pi(i),\pi(j),\pi(k)))$$

$$+ ...$$

$$+ \widetilde{A}_{1,2,...n}^{(q_x,q_y)} * g(\max(\pi(1),\pi(2)...\pi(n)) - \min(\pi(1),\pi(2)...\pi(n)))$$

Let Q be all possible query windows and let

$$w_{i,j} = \sum_{(qx,qy)\in Q} \widetilde{A}_{i,j}^{(q_x,q_y)}$$

$$w_{i,j,k} = \sum_{(qx,qy)\in Q} \widetilde{A}_{i}^{(q_x,q_y)}$$

...

$$w_{1,2,...n} = \sum_{(qx,qy)\in Q} \widetilde{A}_{1,2,...n}^{(q_x,q_y)}$$

The total access time can be computed as follows:

$$Cost =$$

$$\sum_{1\le i<j\le n} w_{i,j} * g(|\pi(i)-\pi(j)|)$$

$$+ \sum_{1\le i<j\le k\le n} w_{i,j,k} * g(\max(\pi(i),\pi(j),\pi(k)) - \min(\pi(i),\pi(j),\pi(k)))$$

$$+ ...$$

$$+ w_{1,2,...n} * g(\max(\pi(1),\pi(2)..\pi(n)) - \min(\pi(1),\pi(2)..\pi(n)))$$

4. THE OPTIMIZATION METHOD

Having presented the quadratic cost model in terms of total access time in processing all possible spatial range query result sets for a point data set, we next present the optimization method to reduce the total access time.

4.1 General Ideas

It can be observed that the cost model we have developed is structurally similar to the Minimum Linear Arrangement (MinLA) problem in graph theory defined as follows (Daíz, 2002):

$$la(G) = \sum_{(u,v)\in E} w(u,v) * |\pi(u) - \pi(v)|$$

Where $w(u,v)$ is the weight of an edge of nodes u and v in the graph. Function $\pi(u)$ is the same as that defined in Section 3 which maps node u to its position in an arrangement (an ordering or a sequence in our terms). The graph MinLA problem is a well-studied problem and several efficient approximation methods have been proposed (Bar-Yehuda, 2001; Koren, 2002). However, there are two problems concerning the differences between MinLA and our cost model which prevent us from using them directly. The first problem is that, our cost model is quadratic with respect to the differences in the positions of the beginning and ending nodes in a hyper-edge while it is linear in MinLA as its name suggests. The second problem is that there are multiple data items in a query result set and thus a hyper-graph representation (Section 3) is more appropriate for our cost model than a graph representation for MinLA.

To solve the first problem, we observe that the cost model for a single query in terms of DPW+DBW, i.e. $g(L_2)$, increases monotonically as DBW, i.e. L_2, increases and vice versa (Section 3). Thus L_2, which is the hyper-graph version of "edge length", is a good linear approximation of $g(L_2)$. By doing so we are expecting that the optimized ordering where the optimization is based on the definition of $la(G)$ which is linear with respect to L_2, is also a good ordering according to quadratic cost model respect to L_2. To solve

the second problem, we adopt the approximation algorithm proposed in (Bar-Yehuda, 2001) for our application. We have proved the correctness of using the algorithm for hyper-graphs. The proof can be found in ([HREF 1]) and is omitted here due to space limitation.

An ordering of data items using a mapping function π is called the ordering π. For the rest of this paper, we will call the access time of the ordering π "linear cost" if it is computed according to the definition of *la(G)*, or "quadratic cost" if it is computed according to our quadratic cost model. We next briefly introduce the approximate algorithm for optimization proposed in (Bar-Yehuda, 2001) and illustrate the method through a simple example.

4.2 The Approximation Algorithm

The approximation algorithm proposed in (Bar-Yehuda, 2001) adopts a divide-and-conquer strategy. It imposes a global ordering constraint on a graph by using a Binary Decomposition Tree (BDT). A BDT *T* is a binary tree that has all the nodes in a graph as its leaf nodes as shown in Fig. 6. For each tree $t \in T$ that has two sub-trees t_1 and t_2, we have two options in placing the nodes under it into a broadcast channel, i.e., either the nodes under t_1 are placed ahead of the nodes of t_2 (called 0-orientation), or the nodes under t_2 are placed ahead of the nodes under t_1 (called 1-orientation). The algorithm starts with the root of the BDT and computes the costs of the two possible orientations of its two sub-trees recursively. The orientation that has lower cost is kept while the one that has a higher cost is discarded. The resulted orientations at each intermediate node of the BDT form an orientation tree that has the same structure as the BDT. The orientation tree determines an ordering sequence of all the nodes in a graph.

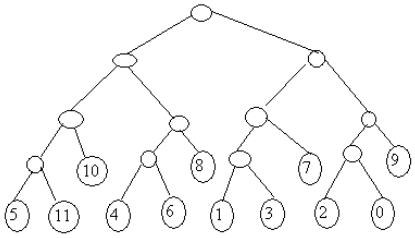

Fig . 6 A Binary Decomposition Tree

Since *t* has two orientations and the orientations of its two sub-tress, t_1 and t_2, are independent of each other, it is easy to prove that there are 2^{n-1} orderings for a full and balanced BDT. The efficiency of the algorithm is achieved by examining 2^{n-1} orderings in $O(n^2)$ time by computing the cost of *t* from the costs, the left outer cuts and the right outer cuts (see definitions below) of its two sub-trees in linear time. The left outer cuts and the outer right cuts of *t* again can be computed from the left outer cuts and the right outer cuts of its two sub-trees in linear time.

The Cost $_{L,V(t),R,\pi}$ with regards to a BDT sub-tree *t* under the ordering π is defined as the followings:

$$\text{Cost}_{[L,V(t),R,\pi]} = \sum_{(u,v)\in E} \begin{cases} w(u,v)*|\pi(u)-\pi(v)| & u\in V(t)\cap v\in V(t) \\ w(u,v)*\pi(u) & u\in V(t)\cap v\in L \\ w(u,v)*|V(t)-\pi(u)| & u\in V(t)\cap v\in R \\ 0 & otherwise \end{cases}$$

Where *V(t)* is the node set of *t*, *L* and *R* are the node sets that are to the left of *V(t)* and to the right of *V(t),* respectively. When *t* is the whole BDT, L=R=∅, $\text{Cost}_{L,V(t),R,\pi}$ is exactly the la(G). We next show how to compute the costs under the two orientations efficiently by recursive computation.

Let $\hat{t}$ be an orientation tree node corresponding to the ordered partition which consists of *L*, *V(t)* and *R* denoted as (L,V(t),R). We call the left child *left* and the right child *right* in both orientations of *t*. Suppose that each child of the BDT is assigned a cost for both 0-orientation (i.e., cost(left(0) and cost(right(0)) or 1-orientation (i.e., cost(left(1) and cost(right(1)). The cost of *t* under the two orientations can be computed as follow:

$$\begin{aligned} \text{cost}_0 &= \text{cost}(\text{left}(0))+\text{cost}(\text{right}(0)) + |V(t_2)|.\text{cost}(V(t_1),R) \\ &\quad + |V(t_1)|.\text{cost}(L,V(t_2)) \\ \text{cost}_1 &= \text{cost}(\text{left}(1))+\text{cost}(\text{right}(1)) + |V(t_1)|.\text{cost}(V(t_2),R) \\ &\quad + |V(t_2)|.\text{cost}(L,V(t_1)) \end{aligned} \quad (1)$$

$\text{Cost}(L,V(t_1))$ and $\text{cost}(L,V(t_2))$ are called left outer cuts and $\text{cost}(V(t_1),R)$ and $\text{cost}(V(t_2),R)$ are called right outer cuts and they can be computed recursively as follows. Let left_cut($\hat{t}$) and right_cut($\hat{t}$) be the left outer cut (i.e., $\text{cost}(L,V(t_1))$ or $\text{cost}(L,V(t_2))$) and the right outer cut (i.e., $\text{cost}(V(t_1),R)$ or $\text{cost}(V(t_2),R)$) of $\hat{t}$ respectively. Let in_cut be the total weight (cost) of edges whose beginning node and ending node have *t* as the Least Common Ancestor (LCA). When $\hat{t}$ is a leaf node, the values of the outer cuts are computed by considering the edges incident to *t*. When $\hat{t}$ is an intermediate node we have the followings:

$$\begin{aligned} &\text{cost}(\text{left\_cut}(\hat{t})) = \text{cost}(\text{left\_cut}(\text{left}(\hat{t}))) \\ &\qquad + \text{cost}(\text{left\_cut}(\text{right}(\hat{t}))) - \text{cost}(\text{in\_cut}(t)) \\ &\text{cost}(\text{right\_cut}(\hat{t})) = \text{cost}(\text{right\_cut}(\text{left}(\hat{t}))) \\ &\qquad + \text{cost}(\text{right\_cut}(\text{right}(\hat{t}))) - \text{cost}(\text{in\_cut}(t)) \end{aligned}$$

Thus formula (1) can be rewritten as:

$$\begin{aligned} \text{cost}_0 &= \text{cost}(\text{left}(0))+\text{cost}(\text{right}(0)) + |V(t_2)|.\text{cost}(\text{right\_cut}(\hat{t}_1)) \\ &\quad + |V(t_1)|.\text{cost}(\text{left\_cut}(\hat{t}_2)) \\ \text{cost}_1 &= \text{cost}(\text{left}(1))+\text{cost}(\text{right}(1)) + |V(t_1)|.\text{cost}(\text{right\_cut}(\hat{t}_2)) \\ &\quad + |V(t_2)|.\text{cost}(\text{left\_cut}(\hat{t}_1)) \end{aligned} \quad (2)$$

As discussed earlier, the cost of *t* is the lower of the two costs, cost_0 and cost_1.

4.3 An Example

We use the example data set shown in Fig. 7 to illustrate the optimization algorithm. In this example, the query window size is 10×10, thus all the four points (1, 2, 3 and 4) have extended areas (R_1, R_2, R_3 and R_4) of size 100, i.e. $A_1=A_2=A_3=A_4=100$. The intersection of R_1 and R_2 is R_{12} whose area is $A_{12}=36$. Similarly we have $A_{13}=16$, $A_{23}=56$, $A_{24}=7$, $A_{34}=12$, $A_{123}=14$ and $A_{234}=4$.

By using the Inclusion-Exclusion Theorem in set operations, we can compute $\widetilde{A}_1$, the area of the distribution region of the centers of the query windows that contain only the data point 1, as follows:

$A_1-(A_{12}+A_{13}-A_{123})=100-36-16+14=62$.

Similarly we can have $\widetilde{A}_2$=19, $\widetilde{A}_3$=34, $\widetilde{A}_4$=85, $\widetilde{A}_{12}$=22, $\widetilde{A}_{13}$=2, $\widetilde{A}_{23}$=38, $\widetilde{A}_{24}$=3, $\widetilde{A}_{34}$=8, $\widetilde{A}_{123}$=14 and $\widetilde{A}_{234}$=4. The hyper-graph to represent the spatial semantics among the points is shown in Fig. 7. Now we are ready to illustrate the optimization process.

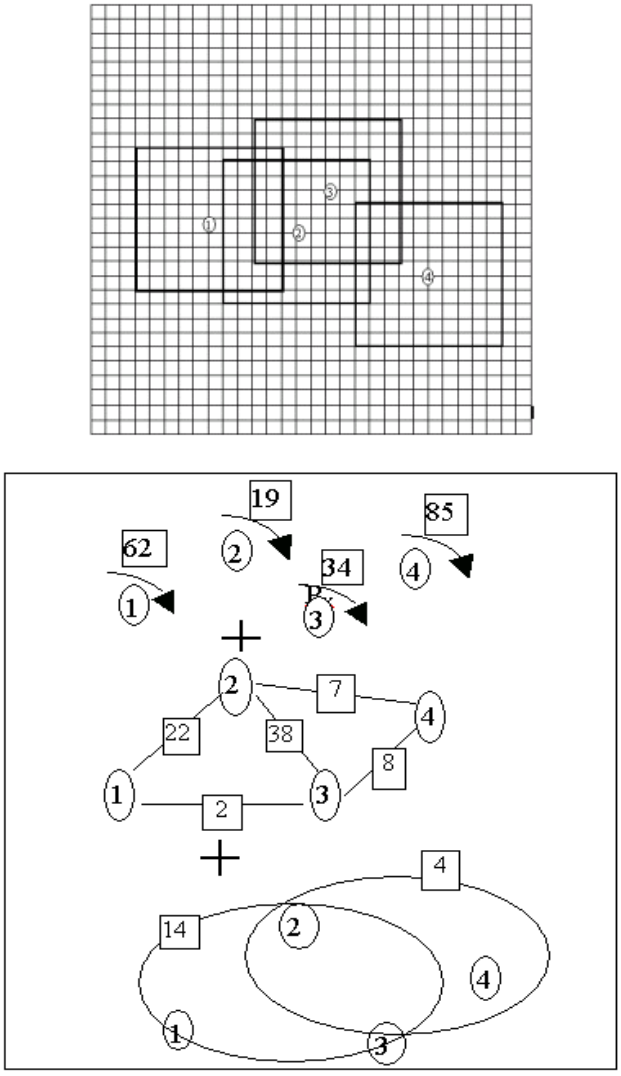

Fig 7. The Example Point Data Set (Top) and Its Hyper-graph Representation (Bottom)

We first remove the 4 hyper-edges that have only single node since their average access time cost is always the half of the broadcast cycle and do not contribute to the ordering. We then build a full and balanced binary tree for the four nodes and use it as our BDT (Fig. 8). Among the remaining 7 hyper-edges, the common ancestor of the nodes in edge {1,2} is rooted at T_{11} with in_cut of 22, the common ancestor of the nodes in edge {3,4} is rooted at T_{12} with in_cut of 8 and the common ancestors of the nodes in rest edges are rooted at T_0 with their total in_cut as the summation of the following values: 2 for edge {1,3}, 38 for edge {2,3}, 3 for edge {2,4}, 14 for edge {1,2,3} and 4 for edge {2,3,4}. Thus the total in_cut of T_0 is 61.

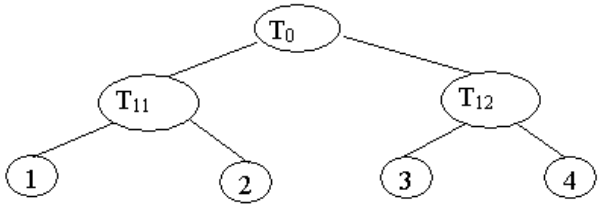

Fig. 8 The BDT of the Example

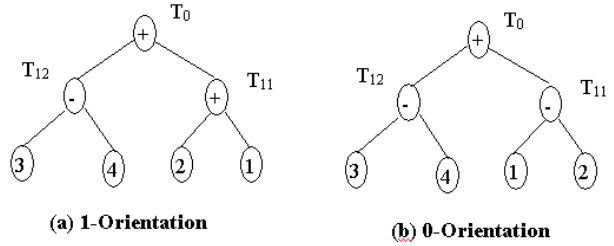

Fig. 9 The Orientation Trees of Two Possible Orientations of T_{11}

For illustration convenience, we also use "+" to denote the1-orientation and "-" to denote the 0-orientation. For the orientation tree in Fig. 9 (a), the ordering of the four nodes is {3,4,2,1}. Node 2 is the ending node of edges {2,3}, {2,4} and {2,3,4}, thus the left_cut of node 2 is 38+3+4=45. Node 1 is also the beginning node of edge {1,2} and thus its right_cut is 22. Similarly, the left_cut of node 1 is 2+14+22=38 and the right_cut of node 1 is 0. Since node 1 and 2 are leaf nodes, their costs are the same as their left_cuts which are 45 and 38, respectively. Thus the left_cut and the right_cut of their parents, T_{11}, are 45+38-22=61 and 22+0-22=0, respectively. The total cost of T_{11} under the current 1-orientation can be computed as 45+38+(22-22)*1+(38-22)*1=99. If the orientation of T_{11} is switched to the 0-orientation (Fig. 9(b)), we can get the left_cut of node 1 as 2, the right_cut of node 1 as 22, the left_cut of node 2 as 81 and the right_cut of node 2 as 0, thus the left_cut and and the right_cut of T_{11} under the current 0-orientation are 2+81-22=61 and 22+0-22=0, respectively. The total cost of T_{11} is 2+81+(22-22)*1+(81-22)*1=142. Since the 1-orientation of T_{11} has smaller cost (99) than the 0-orientation cost of T_{11} (142) we set 1-orientation to T_{11}. Similarly, we set 1-orientation to T_{12} since its 1-orientation cost (15) is smaller than its 0-orientation cost (66). The left_cut and the right_cut under the 1-orientation of T_{12} are 0 and 61, respectively. Thus the total cost of T_0 is 99+15+(61-61)*2+(61-61)*2=114 under the ordering of [4,3,2,1]. This is the global optimal cost of all possible orderings (4!=24).

To compare the goodness of the approximation, we enumerate all possible 4!=24 orderings and compute both the linear cost and quadratic cost of data access time as shown in Fig. 10. The *x*-axis represents all possible individual ordering of the set of the 4 data items. The *y*-axis represents the access time associated with each data ordering. It is easy to see that they have the same trend. The optimal order under the linear cost model is also the optimal order under the quadratic cost model. This confirms our theoretical results. The access time under the quadratic model is always larger than that under the linear model as expected since the former include both DPW and DBW while the latter includes only DBW.

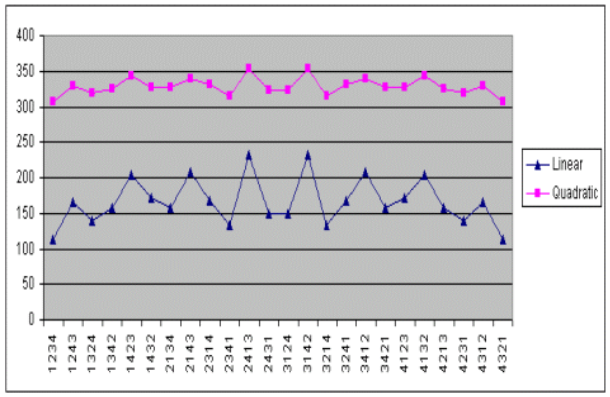

Fig. 10 Comparison of Access Time of Linear versus Quadratic Models

5 EXPERIMENTS

We perform experiments to compare three data ordering methods: 1) Random ordering without taking any data semantics into consideration, 2) Ordering using Heuristics, such as R-Tree traversal ordering and Hilbert ordering, 3) Ordering using the proposed optimization method. In this section, we first describe the data sets used in our experiments, then we present the comparison results.

5.1 Generating Data Sets

We use five synthetic data sets in our experiments with sizes of 100,200, 300, 400 and 500 points, respectively. They all have a data space of [0,1) ×[0,1) and we use a query window size of (0.1,0.1). The points in the data sets are generated randomly within the data space and with the following restrictions: First, the Extended Region of a point intersects with no more than N other Extended Regions. This is to ensure that the lengths of the hyper-edges are bounded by the constant N to be complied with the requirements of the optimization algorithm. We choose N to be 10 in the experiments. Second, The distances between a point and the points that fall into its extended region are no less than 1% of the radius of the query window (0.005 in our experiments). This is to prevent us from generating very tiny intersected regions to ensure that the weights of the hyper-edges is not too small to be meaningful for optimization. And third, we remove the points whose extended region does not intersect with any other extended regions since they do not contribute to ordering of the nodes as discussed at the beginning of Section 4.3. This might make the size of some data sets slightly less than their original size, e.g., data sets 1,3,4 and 5 listed in Table 1. Table 1 shows the parameters of the five data sets. It can be observed that as the number of the points increases, both the number of hyper-edges and the average nodes per hyper-edge increase monotonically.

Table 1. Parameters of the Data Sets

Data Set	# of Points	# of Hyper-edges	Total # of Nodes in All Hyper-edges	Average Nodes Per Hyper-edge
1	96	253	667	2.64
2	200	1054	3393	3.22
3	294	1796	6358	3.54
4	382	2111	8854	4.19
5	452	2147	10802	5.03

5.2 Comparison of Random Orderings

To form a basis for the comparisons of different orderings, we first generate 1000 random orderings (i.e., data items are sequenced randomly) and compute their access time. We report the minimum, the maximum and the average access time among the 1000 random orderings as shown in Table 2. Note that all the access times reported in this and the following sub-sections are the costs under the quadratic cost model.

Table 2. Results of 1000 Random Orderings

Data Set	Minimum AT (Rand_Min)	Maximum AT (Rand_Max)	Average AT (Rand_Ave)	Improvement $\frac{Rand\_\max - Rand\_Min}{Rand\_Ave}$
1	41.89	47.05	44.73	11.54%
2	218.79	236.23	228.69	7.63%
3	365.46	385.18	374.98	5.26%
4	357.93	376.95	369.09	5.15%
5	293.27	306.24	300.74	4.31%

From the results we can see that the improvement percentage (defined as the difference of the maximum and minimum divided by the average) decreases as the data set size increases. This is understandable since the possible number of orderings (n!) increases very fast as the data set size (n) increases, and hence, the portion of 1000 and n! decreases dramatically consequently. This means that the possibility of getting good ordering sequence decreases dramatically by only randomly examining a constant number of orderings, which again suggest ordering heuristics and low-cost optimization methods are desirable. We will examine the effectiveness of the two heuristics, R-Tree traversal ordering and Hilbert ordering, we are going to provide and the optimization method provided in Section 4 in the next two sub-sections.

5.3 Comparison of Two Heuristics

Space Filling Curves (SFC) (Gade, 1998), such as row-wise enumeration of the cells, Peano curve or Z-Ordering, Hilbert-Ordering and Gray-Ordering, which transforms multi-dimensional data into one-dimension can be used to generate the orderings by comparing the SFC code. Although spatial index trees such as the R-Tree family (Guttman, 1984; Sellis, 1987; Beckmann, 1990) are not originally designed to be aware of the order of data items, traversals of these trees do generate orderings that can be used to sequence the data items. Since spatial indexing methods usually maintain spatial adjacencies, the orderings generated by SFCs and spatial index tree traversals are good candidates. They have low computation costs since it takes linear time to traverse an R-Tree and $O(n*\log^n)$ to sort Hilbert SFC codes to generate an ordering. In this sub-section, we evaluate the access time under both Hilbert ordering and R-Tree traversal ordering. The results are listed in Table. 3.

From the results we can see that Hilbert ordering is only marginally better than the average of 1000 random orderings while R-Tree traversal ordering is more significantly better than the average. It suggests that R-Tree traversal ordering can be a very good ordering heuristic.

Table 3. Comparisons of Hilbert and R-Tree Traversal Ordering Access Time with 1000 Random Orderings Average

Data Set	Rand-Ave	Hilbert Ordering (HO)	R-Tree Ordering (RO)	Hilbert Ordering Improvement $\frac{Rand\_Ave}{HO}-1$	R-Tree Ordering Improvement $\frac{Rand\_Ave}{RO}-1$
1	47.05	45.28	40.06	3.91%	5.17%
2	236.23	228.95	201.68	3.18%	3.08%
3	385.18	373.18	318.81	3.22%	8.77%
4	376.95	363.46	313.4	3.71%	6.58%
5	306.24	297.77	254.73	2.84%	10.09%

5.4 Optimization of R-Tree Ordering

In this section we compare the optimized ordering with the R-Tree ordering and the average of 1000 random orderings. Since the computation time for optimization for all the five data sets are no more than 3 seconds on our Dell Dimension 4100 personal computer with 866MHZ processor and 512M memory, we will not include the computation cost at the server side in our discussion, rather we focus on the access time at the client side. The results for the five data sets are listed in Table 4.

Table 4. Comparison Optimized Ordering, R-Tree Ordering and 1000 Random Orderings Average

Data Set	Rand-Ave	R-Tree Ordering (RO)	Optimized R-Tree Ordering (OO)	Improv1 $\frac{RO}{OO}-1$	Improv2 $\frac{Rand\_Ave}{OO}-1$
1	47.05	40.06	38.09	17.45%	23.52%
2	236.23	201.68	195.65	17.13%	20.74%
3	385.18	318.81	293.1	20.82%	31.42%
4	376.95	313.4	294.04	20.28%	28.20%
5	306.24	254.73	231.38	20.22%	32.35%

From the results we can see that the access time of the optimized orderings are about 17% to 20% better than the heuristic R-Tree ordering and 21% to 32% better than the 1000 random orderings average. The improvements are thus significant.

6. CONCLUSIONS AND FUTURE WORK DIRECTIONS

In this paper, we propose to use the access time of DBW to approximate the access time of DPW+DBW and convert the optimization problem under the quadratic cost model into a MinLA optimization problem. The method is based on two observations. The first observation is the structural similarity between the quadratic cost model we previously developed and the MinLA problem. The second observation is the monotonic relationship between the cost in terms of DPW+DBW and the DBW for a single query. The experiment results using the five synthetic data sets based on optimization method showed that the optimized ordering is 21%-32% better than the 1000 random orderings average under our quadratic cost model. This confirms that both the approximation and the optimization are effective.

For future work, we plan to include access time to the index channel in our cost model and explore more ordering heuristics as well as exact and/or approximation optimization methods. Also we plan to do more experiments using both synthetic and real data sets with different sizes, distributions and densities to examine the effectiveness and scalabilities of the optimization methods.

References

[1]. Charles J. Alpert, Andrew B. Kahng: A general framework for vertex orderings with applications to circuit clustering. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 4(2): 240 –246(1996

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

[3]. Y. C. Chehadeh, A. R. Hurson, Mohsen Kavehrad: Object Organization on a Single Broadcast Channel in the Mobile Computing Environment. Multimedia Tools and Applications 9(1): 69-94 (1999)

[4]. Josep Daíz and Jordi Petit and María Serna: A Survey on Graph Layout Problems. *ACM* Computing Surveys, 34(3): 313-356 (2002)

[5]. Yon Dohn Chung, Myoung-Ho Kim: Effective Data Placement for Wireless Broadcast. Distributed and Parallel Databases 9(2): 133-150 (2001)

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

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

[8]. S. Hambrusch, C.-M. Liu, W. Aref, S. Prabhakar: Query Processing in Broadcasted Spatial Index Trees. SSTD,2001: 502-521

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

[10]. Qinglong Hu, Wang-Chien Lee, Dik Lun Lee: Indexing Techniques for Power Management in Multi-Attribute Data Broadcast. MONET 6(2): 185-197 (2001)

[11]. T. Imielinski, S. Viswanathan, B. R. Badrinath: Energy Efficient Indexing On Air. SIGMOD Conference, 1994:25-36

[12]. T. Imielinski, S. Viswanathan, B. Badrinath: Power Efficient Filtering of Data on Air. EDBT, 1994: 245-258

[13]. T. Imielinski, S. Viswanathan, B. R. Badrinath, Data on Air: Organization and Access. IEEE Transactions on Knowledge and Data Engineering, 9(3): 353-372 (1997)

[14]. Birgitta König-Ries, etc.: Report on the NSF Workshop on Building an Infrastructure for Mobile and Wireless Systems. SIGMOD Record 31(2): 73-79 (2002)

[15]. Y. Koren, D. Harel: A Multi-Scale Algorithm for the Linear Arrangement Problem, Lecture Notes in Computer Science, Vol. 2573, Springer Verlag, 2002:293-306

[16]. Guanling Lee, Shou-Chih Lo, Arbee L. P. Chen: Data Allocation on Wireless Broadcast Channels for Efficient Query Processing. IEEE Transactions on Computers 51(10): 1237-1252 (2002)

[17]. Wang-Chien Lee, Dik Lun Lee, Using Signature Techniques for Information Filtering in Wireless and Mobile Environments. Distributed and Parallel Databases, 4(3): 205-227 (1996)

[18]. Bernd-Uwe Pagel, Hans-Werner Six, Heinrich Toben, Peter Widmayer: Towards an Analysis of Range Query Performance in Spatial Data Structures. PODS 1993: 214-221

[19]. Philippe Rigaux, Michel O. Scholl, Agnes Voisard, Spatial Databases: With Application to GIS, San Diego, CA: Academic Press 2002

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

[21]. Ayse Y. Seydim, Margaret H. Dunham, Vijay Kumar: Location dependent query processing. MobiDE, 2001: 47-53

[22]. Yannis Theodoridis, Timos K. Sellis: A Model for the Prediction of R-tree Performance. PODS 1996: 161-171

[23]. Yannis Theodoridis, Emmanuel Stefanakis, Timos K. Sellis: Efficient Cost Models for Spatial Queries Using R-Trees. TKDE 12(1): 19-32 (2000)

[24]. Reuven Bar-Yehuda, Computing an optimal orientation of a balanced decomposition tree for linear arrangement problems. Journal of Graph Algorithms and Applications, 5(4): 1-27 (2001)

[25]. Jianting Zhang, Le Gruenwald, Prioritized Sequencing for Efficient Query on Broadcast Geographical Information in Mobile-Computing, ACM-GIS, 2002: 88-93

[26]. Jianting Zhang, Le Gruenwald, An Access Time Cost Model for Spatial Range Queries On Broadcast Geographical Data Over Air, to appear in DEXA 2003

[HREF 1] http://coecs.ou.edu/Jianting.Zhang/pdfs/proof.pdf