Spatial Index on Air*

Baihua Zheng[†] Wang-Chien Lee[‡] Dik Lun Lee[†]

[†] Hong Kong University of Science and Technology Clear Water Bay, Hong Kong
{baihua,dlee}@cs.ust.hk
[‡] The Penn State University, University Park, PA 16802
wlee@cse.psu.edu

Abstract

With the advent of wireless networking and personal digital devices, the population of mobile users will increase significantly. Broadcasting is particularly suitable for environments having a large number of clients. In this paper, we study the query processing of some typical location-dependent queries, such as window queries and kNN queries, in a broadcast system. To reduce clients' power consumption and provide efficient services, a transformation of the objects is applied based on Hilbert Curve. Furthermore, a linear index structure is constructed and several algorithms are devised to answer spatial queries. Experiments are conducted to evaluate the performance of the proposed transformation and related algorithms. Results show that the proposed schemes outperform existing algorithms significantly.

Keywords: location-dependent queries, wireless broadcast, index structure, Hilbert Curve

1 Introduction

It is predicted that computing will enter our lives and become pervasive in the near future. A pervasive computing environment will serve a large population of mobile users[1]. Broadcasting is an effective way for disseminating information to a large number of users.

Pervasive computing must be context-aware in order to be intelligent and transparent to the user. Location information is an important class of context information. Since clients in a pervasive computing environment have unrestricted mobility, their locations become dynamic and affect the results of certain spatial queries. Consequently, traditional spatial query processing methods cannot guarantee the same performance that they can achieve in a traditional database environment. In this paper, queries whose answers are dependent on some location information (in most cases, it is the query issuer's position) are called *Location-dependent Queries (LDQs)*.

In a mobile environment, the number of clients is expected to be very large. Furthermore, LDQs are likely to exhibit some temporal and spatial locality. In other words, a user tends to ask the same query over a period of time (e.g., finding hotels as he is driving), and different users tend to query the same objects (e.g., traffic reports or local attractions in a city). Under this situation, it is desirable to disseminate location data on wireless broadcast channels and let users retrieve answers to their LDQs by listening to the channels. Example applications include city guide, search for nearest services, local traffic report and so on. In addition to the scalability advantage, broadcast systems allow clients who know their current positions to retrieve answers without submitting the location information to the server, thus reducing the high uplink cost.

As we illustrate in the later section, the specific characteristics of LDQs and broadcast systems introduce many new challenges that make it difficult for existing technologies to be applied to this new kind of environments. Motivated by this fact, we performed research related to the processing of LDQs in wireless broadcast environments, such as grid-partition index structure for nearest-neighbor queries [19] and d-tree for general LDQs [18]. In this paper, we concentrate on some typical LDQs with the objective to propose an indexing structure that can serve several different kinds of queries. Although much work on efficient index structures for different kinds of queries has been done in the spatial database area, it is the first time that this topic is addressed in a wireless broadcast environment. In this paper, we study the difference between the traditional databases and wireless broadcast systems. Based on the difference, a new index structure based on Hilbert Curve is constructed in order to satisfy the specific demands of broadcast systems.

*Research supported by Research Grants Council, Hong Kong SAR, China under grant number HKUST6225/02E and HKUST6079/01E.

[1]A "user" could be a human user operating a PDA or an information seeking device embedded in the environment.

The rest of this paper is organized as follows. A brief review is provided in Section 2. In Section 3, we explain the particular requirements of the broadcast system in detail and define the main objective of our work. According to the requirements, a new index structure based on the Hilbert Curve, a space filling curve, is proposed in Section 4, along with related algorithms for answering different kinds of queries. The simulation model is constructed and the results are depicted in Section 5. Finally, we conclude this paper in Section 6.

2 Related Work

Focusing on answering LDQs via broadcast channels, we first briefly survey the existing work related to these two different areas. To the best of our knowledge, this is the first work combining the broadcast environment and processing of spatial queries together.

2.1 Location-Dependent Queries

In this paper, we concentrate on two common classes of LDQs, namely, *window query* and *k-nearest-neighbor (kNN)* search. Window query is to find the objects that are within a given window, which is a rectangle in a 2-dimensional space. *k-Nearest Neighbor (kNN)* search is to find k objects among the whole set that contains n ($n >= k$) objects according to a given query q, such that the distance between query point q and any object in the answer set is no longer than the distance between q and any other object in the whole set.

R-tree [7] index and its variants provide a good solution to window query. Based on some heuristic optimization, it groups the objects close to each other into a node, and a window query only visits the nodes that overlap with the query window.

kNN search originally is a very natural problem in computational geometry and was first formulated by Minsky and Papert in 1969. In the 1990's, researchers in spatial database began to become interested in this problem [15, 17]. Currently, the problem is extended to a high dimensional space, such as image similarity comparison and content based retrieval in multimedia applications [2, 8].

According to the number of times that an algorithm scans the whole dataset, the existing algorithms can be divided into two categories: single-step search and multi-step search.

Single-Step Search This kind of algorithms searches kNN based on the suitable index structure, scanning the dataset only once. There are several approaches available from the literature. Branch-and-bound algorithms use heuristic distance information to choose the next node for visiting and prune some impossible branches. Various algorithms differ in the searching order and the metrics used to prune the branches [15, 4, 10]. Incremental algorithms report the objects one by one to allow the algorithm be employed in a pipelined fashion, especially for complex queries involving proximity [9]. Some approaches directly use the voronoi-diagrams, which provide solution-space for the kNN queries for a fixed k [1].

Multi-Step Search The methods in this category scan the dataset multiple times until the proper answers are obtained. Korn, et. al. proposed an adapted algorithm [12]. First, a set of k primary candidates was selected based on stored statistics to obtain the upper bound d_{max} which can guarantee that there are at least k objects within the distance d_{max} from the query point q. Next, a range query was executed on dataset to retrieve the final candidates. An extended version of this algorithm was proposed in [17], in which d_{max} was adapted every time a candidate object was checked.

2.2 Wireless Broadcasting

In mobile computing environments, there are two general approaches to disseminating information to mobile clients:

On-Demand Access: A mobile client submits a request to the server. The server locates the appropriate data and returns it to the mobile client.

Broadcast: Data are broadcast on a wireless channel open to the public. After a mobile client receives a query from its user, it tunes into the broadcast channel and filters out the data according to its real situation.

Compared to on-demand access, broadcast has the advantage of scaling up to service a huge number of clients without any additional cost at the server site. Therefore, it is a promising and desirable dissemination method for the future pervasive computing environment whose client population is expected to be huge.

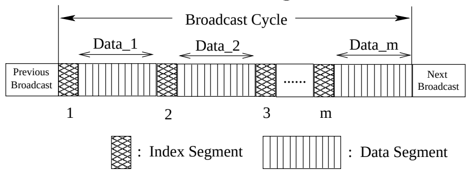


Figure 1: Data and Index Organization on the Broadcast Channel Using the (1, m) Interleaving Technique

Index information can be interleaved with the data to facilitate the search process and save clients' limited battery power [11]. By looking up the index, the mobile client is able to predict the arrival time of the desired data and only needs to tune into the broadcast channel when the requested data arrives. Index organization refers to the interleaving method of the index and the data on the broadcast channel. A well-known organization is the $(1, m)$ interleaving technique [11] (see Figure 1). That is, the whole index is broadcast preceding every $\frac{1}{m}$ fraction of the broadcast cycle.

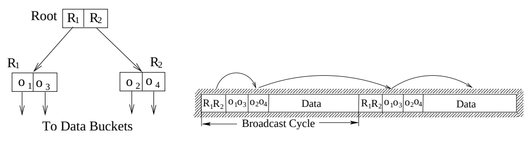
In wireless communications, a bit stream is normally delivered in the unit of *packet* (or *frame*), for the purposes such as error-detecting, error-correction, and synchronization [11]. For example, in the GPRS network a packet can contain the data of up to 1600 bytes [3]. As a result, data are accessed by clients also in the unit of packet, similar to the page concept in traditional databases. In the following description, we use page, rather than packet (frame), for its generosity.

2.3 Discussion

There are some factors making information access in the wireless broadcast environment different from that in the traditional database environment. These differences introduce new research challenges and motivate our work, which can be summarized in three aspects: mobile devices' resource constraints, clients' unlimited mobility, and on-air index for wireless data broadcast.

Devices' Resource Constraints The portability of mobile devices results in various resource constraints, such as small storage spaces, and limited battery power. Among these, power supply is particularly important, and algorithms designed to run on mobile devices should take client's resources into consideration.

Clients' Mobility The unlimited mobility of clients makes location-dependent information access a new and challenging topic. Existing query processing strategies in traditional database systems have not taken into consideration this mobility and changing location issues. Therefore, new information access and dissemination schemes need to be devised.



(a) R-tree Index (b) Branch-and-Bound Search

Figure 2: Linear Access on Wireless Broadcast Channel

On Air Index An important characteristics of wireless data broadcast is that the index information is on air. It uses air as a dissemination medium. The index information is available to the clients only when it is broadcast. Hence, when an algorithm traverses the index nodes in an order different from their broadcast sequence, it has to wait for the next time they are broadcast. In contrast, the index for traditional databases is stored in the resident storage, such as memory and disk space. Consequently, it's available anytime. Since nearly all the existing index structures and algorithms devised for traditional databases do not consider the time-series characteristics of the air index, they cannot be easily deployed in wireless broadcast environments. An example of the well known R-Tree index is given in Figure 2. Assuming that the query processing algorithm first visits the node R_2 and then R_1, and that the server first broadcasts node R_1 then R_2, the access latency is significantly extended since the node R_1 is not available until the next cycle.

As a conclusion, a good index structure and associated algorithm serving spatial queries in a wireless broadcast environment should incur small space cost, take linear broadcast order into account, and perform the search efficiently.

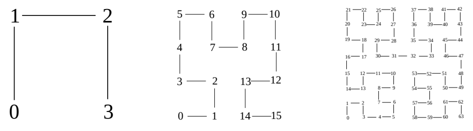
Regarding the performance metrics, common criteria *tuning time* and *access latency* are employed in this study. The former is the time spent by a client listening to the broadcast channels, logically representing the client's power consumption. The latter is the time duration from the point that the client requests some data to the point that the desired data is received. Both the tuning time and the access latency are measured in terms of number of page accesses [11].

3 Hilbert-Curve Index Structure

In response to the linear characteristics of wireless broadcast, we propose an index structure to facilitate the processing of LDQs by linear scanning of the dataset, rather than random traversal of the index nodes based on some heuristics. A space-filling curve, a continuous path visiting every point in a k-dimensional grid exactly once without crossing itself, can serve as an index for LDQs in wireless data broadcast. Well-known space filling curves, including z-curve, Gray-coded curve and Peano curve, are different in the order that the points in the grid space are visited. *Hilbert Curve*, due to its optimal locality [6], is chosen in this paper to build an index for LDQs. In this section, we first explain the basic idea about Hilbert Curve. Then, related algorithms for answering LDQs are illustrated.

Like many other space-filling curves, *Hilbert Curve* maps points from a multi-dimensional space to a 1-dimensional space. *Locality* is an important metric for choosing space filling curves. A mapping from n dimensions to m dimensions (where $m < n$) is considered to have good locality if points that are close to each other in an n-dimensional space are also close to each other after being mapped into an m-dimensional space. Considering the nearest neighbors along the grid axes only, each grid point has $2n$ nearest neighbors in the original space. However, it will only have $2m$ nearest neighbors after being mapped into an m-dimensional space. Therefore, for a mapping from n dimensions to one dimension, such as Hilbert Curve, the best that we can hope for is to have two of the $2n$ nearest neighbors remain to be nearest neighbors in the 1-dimensional space. This subjective criteria is met by Hilbert Curve, which guarantees a good locality feature.

Figure 3(a) gives the basic Hilbert Curve of order 1. To derive a curve of order i, each vertex of the basic curve is replaced by the curve of order $i-1$, which may be strategically rotated and/or reflected to fit the new curve. The Hilbert Curves of order 2 and 3 are depicted in Figure 3(b) and 3(c), respectively, in which the number represents the index value of different points in the Hilbert Curve. For example, point $(1, 1)$ has the index value 2 in H_2.



(a) H_1 (b) H_2 (c) H_3

Figure 3: Hilbert Curves of order 1, 2 and 3

Considering the representative size, the Hilbert Curve assigns enough number of bits to represent the index value in order to guarantee that each point in the original space has distinct value. Given c_i the number of bits for a coordinate in ith dimension, $\sum_{i=1}^{m} c_i$ bits are allocated to represent an index value for an m-dimensional space. Therefore, the Hilbert Curve is sure to visit each point in the space.

Given the mapping function of Hilbert Curve, it is easy for a client to perform conversion between coordinates and Hilbert-Curve index values. Let n be the number of bits assigned to represent a coordinate, the expected time for the conversion is $O(n^2)$. Since n is a preset system constant, the conversion can be done in a constant time. The detailed conversion algorithm is available in [14], so we skip the explanation here.

The Hilbert Curve realizes the linear scan of the objects, the problem left is that whether it can be employed to answer LDQs. In the rest of this section, our algorithms based on Hilbert Curves for window queries and kNN queries, two important classes of LDQs, will be explained.

3.1 Window Queries

To process a window query based on Hilbert-Curve index, a basic idea is to decide a candidate set of points along the Hilbert Curve which includes all the points falling within the query window. These points are retrieved to filter out those falling outside the window. For a query window, an existing algorithm can return the Hilbert Curve values of the first and last points on the bounding box of the query window with a time complexity O(n^2). In the following, we prove that the largest and the smallest Hilbert Curve values on the query window's bounding box provide a sufficient range to contain all the points satisfying the query.

Claim 1: For a given window, the point p within the query window that has the largest Hilbert-Curve index value must be lying on the bounding box.

Proof: Assume there exists another point p' inside the query window which has a larger index value than that of p. Since Hilbert Curve is a continuous path to visit every point in the search space, there must be a point p'' outside of the query window, with a larger index value than p'. Otherwise, p' is the last point of the Hilbert Curve and must be the vertex of the original search space according to the definition of the Hilbert Curve. If we draw a line to connect p' and p'', there must be an intersection point q on the bounding box. Since the index values of the points on the Hilbert Curve are monotonously increasing, the index value of q which is between p' and p'' must be larger than that of p'. Hence, the previous assumption fails. Thus, the point p has the largest value and is on the bounding box. Therefore, our claim is proven. □

Similarly, the point within the query window that has the smallest index value is guaranteed to be on the bounding box. In order to facilitate our discussion, a 2-dimension space is assumed and it is easy to be extended to a high-dimensional space. For a given window, the client computes the first and last points on the box which define the range of the index values. All the objects whose index values are within this range constitute the candidates set. Finally, a filtering mechanism is employed to find out the objects that are really located in the window.

3.2 kNN Queries

A general strategy employed by any kNN algorithm is to determine a search space that is not too small to lose any correct answer and not too big to perform any unnecessary search. It is ideal to know the exact distance between the query point q and the k-th nearest neighbor o_k, which is difficult to be obtained. An alternative is to estimate the distance values. As described in Section 2, some optimistic algorithms use tight prediction to avoid any unnecessary search and some pessimistic ones use loose estimation to avoid any loss of the requested objects. In our algorithm, a range estimation algorithm is proposed based on the locality property of the Hilbert Curve. The pseudo-code is given in Algorithm 1.

As the first step, the kNN objects to the query point along the Hilbert Curve are found and a minimal circle centered at query point is constructed to contain all those k objects. The minimal-bounding rectangle of that circle, containing at least k objects and definitely not causing any loss, serves as the search range. The question left is whether this range is too loose. Based on the fact that Hilbert Curve is close to optimal locality, the kNN objects should lie near the query point along the Hilbert Curve. Consequently, we assume that the bounding rectangle only introduces limited extra search. Later simulation results show the correctness of this assumption.

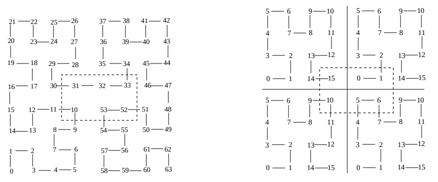
Algorithm 1 kNN Search

```
Input: query point $q$, sorted objects' indexes;
Output: k-nearest neighbors;
Procedure:
1: $index_q = coor\_to\_index(q)$;
2: locate the $i$th object $o_i$ who has the nearest index value
   ($index_i$) to $index_q$;
3: $begin$ = MAX(0, $i$-$k$/2);
4: for j = $begin$, $rad$ = 0; j =< ($begin + k$; j++ do
5:     $p$=$index\_to\_coor(index_j)$; $rad$ = max($rad$, distance($p$, $q$));
6: end for
7: let r be the bounding square centered at $q$ and having $2rad$ as
   side length;
8: $result\_set$ = window_query(r); $answer\_set = \varnothing$; $max\_dis = 0$;
9: for each object $o_i$ in the $result\_set$ do
10:    $dis$ = distance($o_i$, $q$);
11:    if ($answer\_set$ is not full) then
12:        $answer\_set = answer\_set \cup \{o_i\}$;
13:        $max\_dis$ = max($max\_dis$, $dis$);
14:    else
15:        if $dis < max\_dis$ then
16:            replace the farthest object in the $answer\_set$ with $o_i$
               and update $max\_dis$ correspondingly;
17:        end if
18:    end if
19: end for
20: return $answer\_set$;
```

3.3 Search Improvement

The locality of Hilbert Curve is a major factor that impacts the performance of our algorithms. If the nearby points in the original search space have big difference among their index values, the window query will have to check much more points than necessary. A motivating example is depicted in Figure 4(a) in which the dashed rectangle represents a query window. Employing our original algorithm, all the points whose index values between 9 and 54 should be checked. Obviously, this range actually contains many points outside the window.

It can be observed from Hilbert Curve that the order i curve is derived from order $(i-1)$ curve. If a query window crosses several order $(i-1)$ curves, it has a higher probability to contain many more points than necessary due to the low locality of the points near the boundary of the $(i-1)$ curves. Therefore, one solution is to partition the whole space into several disjoint grids and the objects in each grid have their own Hilbert Curve. For a 4-grid partition, each grid only needs an order $(i-1)$ curve, rather than a part of order i curve for the whole search space. Figure 4(b) shows that, after partitioning, the candidate set contains much fewer objects. We only need to check the points whose index values are between 0 and 2 for the grid in quadrant 1, and points whose values are between 14 and 15 for the grid in quadrant 2 and so on.



(a) Before Partition (b) After Partition

Figure 4: Improvement Introduced by Search Space Partition

Besides this advantage, the partition can also reduce the representation size. If the original space needs n bits to represent the index value of an object, it only requires $(n-2)$ bits for 4-grid partition in a two-dimensional space. In order to make a full use of this kind of reduction, we partition the space into the number that is powered by 2.

For a window query, the original query window should be partitioned into several disjoint sub-rectangles. For each grid, the overlapping between the query window and the grid produces the sub-rectangle.

We can observe that the larger the number of partitions, the fewer the number of objects checked, since the partition itself preserves the objects' locality. However, as we mentioned before, objects in broadcast environments are accessed in the unit of page, rather than by objects themselves. Hence, a smaller number of objects searched does not necessary result in a smaller number of page accessed, which depends on the paging strategy employed to organize the index information. The other cost is that the client has to perform multiple window searches in the partitioned space for a query. The design of good guiding rules for partitioning is our next work objective.

4 Performance Evaluation

This section evaluates the performance of the proposed Hilbert-Curve index by comparing it to the traditional indexes for spatial queries. Two datasets are used in the evaluation. In the first dataset (UNIFORM), 10,000 points are uniformly generated in a square Euclidean space. The second dataset (REAL) contains 5848 cities and villages of Greece, which is extracted from the point dataset available from [5]. The discrete-time simulation package *CSIM* [16] is used to implement the model.

For the existing index, we evaluate the search algorithm based on R-tree and original search algorithm is modified to meet the linear visiting request of broadcast environments. No matter where the query is, the MBRs are accessed sequentially while impossible branches are pruned according to well-defined heuristics (see [15] for details). Since the objects are available a priori, the STR packing scheme is employed to build the R-tree (denoted as *STR* R-tree in the later presentation) in order to provide a fair performance comparison [13]. The R-tree is broadcast in a depth-first order to facilitate rollback operations. A packing algorithm using *Hilbert-Curve* for R-tree is available. It is also implemented in our simulation in order to evaluate the impact introduced by the Hilbert-Curve on R-tree which is denoted as *Hilbert-Curve* R-tree.

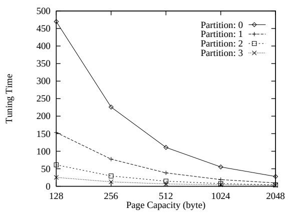
As we explained previously, all the information transferred in the wireless channels should be in the unit of page and the detailed paging scheme employed is explained as follows. Since Hilbert-Curve index belongs to linear information, the well-known B-tree is used. The fan-out of a node can be decided according to the page capacity. At the leaf level, each object is represented by its Hilbert Curve value and a pointer pointing to the data page containing the real data.

The system model in the simulation consists of a base station, lots of clients, and a public channel for broadcast. *WinSideRatio* defines the ratio of the query window's side length to the side length of the whole search space and has the default value of 0.1. The available bandwidth is set to 1000 kbps, and page capacity changes from 2^7 to 2^{11} bytes. Two floating-point numbers are used to represent a two-dimensional coordinate, each one assigned 4 bytes. The same amount of bits are for an index value of Hilbert Curve.

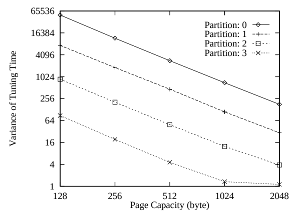
4.1 Partitioning Space

As mentioned above, the search performance is dependent a lot on the locality of the Hilbert Curve. From the observation, partitioning the search space into smaller grids can reduce the probability that two nearby points have a large difference between their index values. The performance improvement obtained by partitioning on the UNIFORM dataset is depicted in Figure 5. As we mention before, one advantage of partition is to reduce the representation size. Consequently, we partition the space into p_i sub-partitions along ith dimension, given p_i equals to 2^{q_i}. Hence, the represent size for the index value of Hilbert Curve can be decreased by the sum of q_i, i.e., $\sum_{i=1}^{m} q_i$, for an m-dimensional space. The number following the word **Partition** is q_i. In current implementation, all the dimensions are partitioned into the same number of sub-partitions, therefore q_i is same for all different i. For instance, *Partition:1* means that the original two-dimensional space is divided into $2^1 \times 2^1$ grids. Without explicit specification, Partition 2 services for the default setting in later simulations.

From the result, we can observe that the performance increases a lot due to the partition, in terms of both tuning time and its variance. However, the computational overhead on clients are increased since it has to determine the boundary index values for multiple sub-query windows, rather than for a single query window in the original situation.



(a) Tuning Time

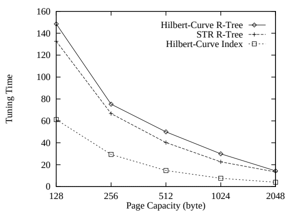


(b) Variance of Tuning Time

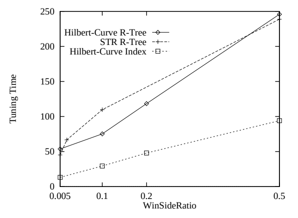
Figure 5: Tuning Time and its Variance (UNIFORM)

4.2 Window Query

As described before, *window query* is a common and important query type in spatial databases, which is frequently used to evaluate the performance of spatial indexing structures. In this subsection, the performance of the newly proposed Hilbert-curve index, compared with various traditional methods, is presented. Figure 6 shows the comparisons for UNIFORM dataset under fixed window size with various page capacity and various window size with a fixed page capacity (256 bytes).



(a) Fixed Size



(b) Various Size

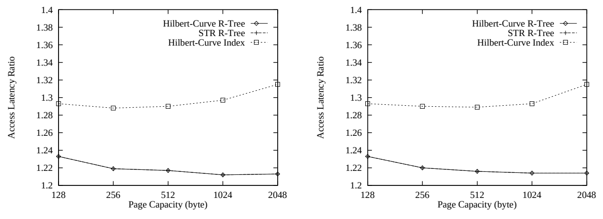
Figure 6: Tuning Time of Window Query (UNIFORM)

The most obvious observation obtained is that Hilbert-Curve index with Partition 2 has the best performance, and it outperforms the other two significantly. For UNIFORM dataset, the improvement in terms of tuning time is about 61.9% and 67.2% over STR R-tree and Hilbert-Curve tree, respectively. For REAL dataset whose result is omitted due to space limitation, the advantage of Hilbert-Curve index is also dramatic. Compared to STR R-tree and Hilbert-Curve tree, the improvement is 53.4% and 55.6% in terms of the tuning time.

In order to provide a comprehensive evaluation, the size of the query window is also changed. The value of parameter *WinSideRatio* is changed from 0.05, to 0.1, to 0.2, and finally to 0.5. As we expected, the Hilbert-Curve index with Partition 2 has the best performance and its gain becomes more obvious as the size of the query window increases.

4.3 kNN Queries

K-Nearest-Neighbor query is one of the most representative spatial queries. It returns k objects that are nearest to the query point. When k is set to 1, it is the famous Nearest-Neighbor search. In this section, the performance of different indexing structures for kNN search is compared. First, we evaluate their performance for traditional NN problem, then for fixed k, and finally for various settings of k.



(a) UNIFORM Dataset (b) REAL Dataset

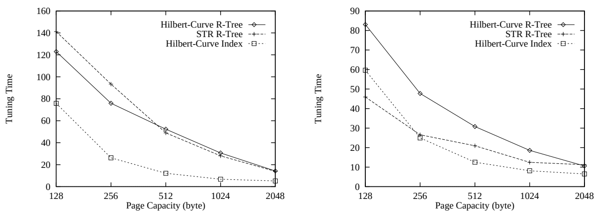
Figure 7: Access Latency of Different Indexing Structures for KNN Queries

The space cost is not a big issue in the traditional disk-index environment, with the enlargement of the disk capacity and reduction of its price. However, in the broadcast environment, all the index information has to occupy some bandwidth for transformation and somehow affects the access latency of the clients. Therefore, the index size is preferred to be small. Since our algorithm devised to solve $k$nn problem should scan the index twice, with the first time to decide the necessary search boundary and the second time to obtain the answer objects, its index size will be larger than that of R-tree. Hence, the expected access time of clients achieved by (1, m) index organization algorithm [11] is compared first to guarantee that the new index does not introduce too long latency and its performance is depicted in Figure 7. Assuming the access latency of the scheme having no index information as 1, R-tree introduces the access latency about 1.23 and Hilbert-Curve based index incurs the latency about 1.30 for the UNIFORM dataset. For the REAL dataset, the result is nearly the same. Hence, we can make the conclusion that the index overhead caused by Hilbert-Curve index is acceptable and similar as it of other existing ones.

4.3.1 Nearest-Neighbor Queries (k=1)

Nearest-Neighbor search is a special case of kNN queries. It has been frequently used as a test case to evaluate the performance of index structures proposed for kNN problem. Figure 8 depicts its performance obtained.

Obviously, Hilbert-Curve index can provide a better performance when partitioned into several sub-grids. It outperforms STR R-tree and Hilbert-Curve R-tree for about 66.4% and 64.5%, respectively, on UNIFORM dataset. While the outperformance for REAL dataset is not so significant, only 18.7% and 46.0% respectively.

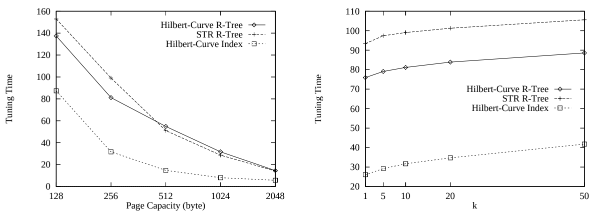


(a) UNIFORM Dataset (b) REAL Dataset

Figure 8: Tuning Time of Nearest Neighbor Query

4.3.2 K-Nearest Neighbor Queries

For the general kNN search, Figure 9 shows the performance of various indexes for UNIFORM Dataset. For various setting of k, the page capacity is set to 256 bytes.



(a) $k = 10$ (b) Various k

Figure 9: Tuning Time of kNN Query (UNIFORM)

Considering the kNN search problem, with a fixed value of k or varying values of k, the Hilbert-Curve index with partition performs the best in nearly all the cases. The first observation is that Hilbert-Curve index with partition always achieves the best performance for UNIFORM dataset, while it somehow performs a little bit worse than STR R-tree in the REAL dataset whose result is omitted due to the space limitation, especially for the small page capacity or large number of the request k. The second observation is that for the comparison between STR R-tree and Hilbert-Curve R-tree, the former works better for the REAL dataset and the latter achieves some gain of the performance for the UNIFORM dataset. The reason causing these two behaviors is that Hilbert Curve has a better location locality for the dataset that uniformly distributed, compared to the skew distributed dataset. From the previous simulation result, Hilbert-Curve based index always achieves a much better performance for the window query for both dataset. Therefore, the only reason that it works worse than STR R-tree for kNN search is that the approximated range window is larger than necessary which results in a worse performance due to that superfluous search.

5 Conclusion

With the advent of wireless networks and popularity of portable digital devices, the pervasive computing era will soon arrive. Wireless data broadcast, which allows simultaneous access by an arbitrary number of clients, is a very efficient and scalable information dissemination method. This paper addresses the problem of answering location-dependent spatial queries via broadcast channels. We first discuss the specific characteristics of broadcast environments and conclude that existing indexing structures are not suitable for this new environment. Then a new index structure based on a space-filling curve, Hilbert Curve, is proposed to enable linear broadcast of a multi-dimensional space, along with several search algorithms for window queries and kNN queries. A simulation model is implemented and the result shows that the proposed structure outperforms the other methods significantly in nearly all situations, for both the synthetical dataset and the real dataset.

Currently, the whole index information should be scanned twice in order to answer kNN queries. As such, the index overhead is increased. In our future research, we are investigating algorithms that can achieve the same exact answers using a single scan. Also, other kinds of LDQs, such as continuous nearest-neighbor query, will be studied.

References

[1] P. K. Agarwal, M. Berg, J. Matousek, and O. Schwarzkopf. Constructing levels in arrangements and higher order voronoi diagrams. *SIAM Journal on Computing*, 27(3):654–667, June 1998.

[2] S. Berchtold, B. Ertl, D. A. Keim, H. P. Kriegel, and T. Seidl. Fast nearest neighbor search in high-dimensional space. In *Proceedings of the Fourteenth International Conference on Data Engineering (ICDE'98)*, pages 209–218, February 1998.

[3] J. Cai and D. J. Goodman. General packet radio service in GSM. *IEEE Communications Magazine*, 35(10):122–131, October 1997.

[4] K. L. Cheung and W.-C. Fu. Enhanced nearest neighbour search on the r-tree. *SIGMOD Record*, 27(3):16–21, 1998.

[5] Spatial Datasets. Website at http://dias.cti.gr/~ytheod/research/datasets/spatial.html.

[6] C. Gotsman and M. Lindenbaum. On the metric properties of discrete space-filling curves. *IEEE Transactions on Image Processing*, 5(5):794–797, May 1996.

[7] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 47–54, 1984.

[8] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. What is the nearest neighbor in high dimensional spaces? In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB'00)*, September 2000.

[9] G. R. Hjaltason and H. Samet. Ranking in spatial databases. In *Proceedings of the 4th International Symposium on Advances in Spatial Databases (SSD'95)*, pages 83–95, 1995.

[10] Gí. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM Transactions on Database Systems*, 24(2):265–318, 1999.

[11] T. Imielinski, S. Viswanathan, and B. R. Badrinath. Data on air - organization and access. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 9(3), May-June 1997.

[12] F. Korn, N. Sidiropoulos, C. Faloutsos, E. Siegel, and Z. Protopapas. Fast nearest neighbor search in medical image databases. In *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB'96)*, pages 215–226, 1996.

[13] S. T. Leutenegger, J. M. Edgington, and M. A. Lopez. Str: A simple and efficient algorithm for r-tree packing. In *Proceedings of the 13th International Conference on Data Engineering (ICDE'97)*, pages 497–506, Birmingham, UK, April 1997.

[14] D. Moore. Hilbert curve. URL at http://www.caam.rice.edu/ dougm/twiddle/Hilbert.

[15] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data (Sigmod'95)*, pages 71–79, May 1995.

[16] H. Schwetman. *CSIM user's guide (version 18)*. Mesquite Software, Inc, http://www.mesquite.com, 1998.

[17] T. Seidl and H. Kriegel. Optimal multi-step k-nearest neighbor search. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (Sigmod'98)*, pages 154–165, July 1998.

[18] J. Xu, B. Zheng, W.-C. Lee, and D. L. Lee. Energy efficient index for querying location-dependent data in mobile broadcast environments. In *Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE'03)*, Bangalore, India, March 2003.

[19] B. Zheng, J. Xu, W. C. Lee, and D. L. Lee. An all-around grid-partition index for nearest-neighbor queries. Technical report, Dept. of Computer Science, Hong Kong Univ. of Science and Technology, Feb. 2002.