Search RNN on Broadcast Environment

Lien-Fa Lin*

Department of Computer Science and Information Engineering National Cheng-Kung University
E-mail: lienfa@cc.kyu.edu.tw

Abstract

While the RNN is well studied in the traditional wired, disk-based client-server environment, it has not been tackled in a wireless broadcasting environment. The linear property of wireless broadcast media and power conserving requirement of mobile devices make the problem particularly interesting and challenging. In this paper, the issues involved with organizing location dependent data and answering RNN queries on air are investigated. An efficient data organization, called Jump Rdnn-tree, and the corresponding search algorithms are proposed. Performance of the proposed Jump Rdnn-tree and other traditional indexes (enhanced for wireless broadcast) is evaluated using both uniform and skew data. The results show that Jump Rdnn-tree substantially outperforms the traditional indexes.

Keyword: location-dependent services, data broadcast, energy-conserving, mobile computing

1. Introduction

Owing to the popularity of personal digital devices and advances in wireless communication technologies, location-based services (LBSs) have received a lot of attention from both of the industrial and academic communities [2].

The RNN problem has been introduced in database setting by Korn and Muthukrishman [3] along with several applications. For example, the bank plans to establish a new branch. If customers always prefer the nearest branch, then the new branch should be established on the location where the distance to such location for the majority of customers is shorter than that to other banks. Another common example is how a taxi driver chooses customers. By using wireless devices, a taxi driver may know the location of a customer who is looking for a taxi. From the view of competition, RNN is more meaningful than NN. As shown in Figure 1, the nearest neighbor to Taxi A is Customer C, but that does not necessarily mean Taxi A is the most likely to get to Customer C because Taxi B is even closer to Customer C. On the contrary, Taxi A should head for Customer D because Taxi A is the nearest neighbor in relation to Customer D. That is, the RNN for Taxi A is Customer D, and Taxi A may get to Customer D faster than all other taxis.

![Figure 1: Example of RNN query.](image)

The limited battery capacity of mobile devices makes power become a critical issue in wireless environment. Therefore, there is much literature dedicated to general query processing on mobile devices with effective power management [8,9]. From these studies we have deduced some principles for designing a good on air index. We use these principles to design an on air index method that can process RNN query efficiently. In addition, simulation experiments proved that our method may significantly improve efficiency when compared to Rdnn-tree modified for broadcasting environment.

The rest of the paper is organized as follows. Section 2 is an overview of related work. In Section 3, we describe the effectiveness on air indexing design rules.

* Lien-Fa Lin is also a lecturer at the Department of Information Communication Kao Yuan University, No.1821, JhongShan Rd, Lujhu Township, Kaohsiung Country 821, Taiwan, R.O.C.
The details of Jump Rdnn-tree index structure are introduced in Section 4. Finally, we summarize the paper and describe our future work in Section 5.

2. Related work

In this section, we shall introduce RNN query, and research topics that relate to on air index and RNN query in broadcasting environment in the following subsections.

2.1 Reverse Nearest Neighbor Query

The RNN query application is quite widespread, including decision-making support system, biological information and so on. In [1] mention many about the RNN query application example.

A straightforward solution to computing reverse nearest neighbor (RNN) queries is to check for each point whether it has a given query point as its nearest neighbor. However, this approach is unacceptable when the number of points is large. The general method is use an specially R-tree (is called RNN tree [4]) to process query. Conjun Yang [1] proposed Rdnn-tree index structure to improve the method in [4]. This Rdnn-tree index structure can be applied to solve NN and RNN query problem simultaneously. The difference between Rdnn-tree and R-tree is that Rdnn-tree has recorded each object’s NN information which can be used to process RNN query effectively.

2.2 On Air Index

Because linear access is not considered in the design of traditional index structure, the algorithm that is currently adopted in disk-based spatial index can not satisfy the need of effective power conservation. Shown in Figure 2 is R-tree index; its broadcasting sequence is root, $R_1$, and $R_2$. The visit sequence for searching for NN with a given query point of $q_2$ is: Root is first visited because the distance between $q_2$ and $R_2$ is shorter than that to $R_1$. Therefore, $R_1$ is skipped and $R_2$ is visited first. However, the shortest object to $q_2$ is $O_4$ of $R_1$ in MBR, and therefore $R_1$ must be first visited. However, at this time $R_1$ has just been broadcasted and it can only be accessed in the next broadcasting cycle.

With the feature of linear access in broadcasting environment, if the broadcasting sequence differs from the sequence of query, then long access latency will occur. Therefore, branch-and-bound query method in broadcasting environment is not a very effective method in term of access latency. An alternative is direct access to MBRs sequentially. However, this method will cause unnecessary traversal of MBRs, and index search performance will not be optimized. For example, the search of NN for $q_1$, the real NN is $O_4$ of MBR $R_2$, and accessing to $R_1$ is obviously a waste of resource. Therefore, a new index method must be designed for wireless broadcasting environment to effectively adopt the feature of linear access in broadcasting environment and satisfy the need of power conservation for mobile devices.

3. Effective Broadcast Index Design

Access latency of accessing to data and tuning time that a mobile device requires in active mode are the two benchmarks for broadcast index efficiency measurements. Access latency is the time required for accessing to data from the moment a user gives the query command to the data that satisfies the query is accessed. Tuning time is the time required for users to receive requested data in active mode. Broadcast index is mixed with broadcast data and sent out together, and MU receives data in the following three steps [5]: (1) Initial probe: during any point in time of broadcasting, a user tunes into a broadcast channel and wait for the index data to be broadcasted. This period of time is called initial probe waiting. (2) Index search: When index data arrives, a user receives the index data, selectively accesses some index data according to his/her needs, and finds the location of the requested data. (3) Data retrieval: When the requested data arrives, a user downloads and accesses to the data. The time required for these three steps shall influence broadcast index efficiency. Therefore, a design of effective broadcast index must reduce the time required for these three steps.

Reduction of initial probe time: Initial probe waiting is the time that a user waits for index data. By duplicating multiple indexes in the entire broadcast cycle, the possibility of index appearing may increase, and the initial probe waiting time can be reduced. Imielsinski et al. [9] used interleaving method to duplicate m copies of index data in order to reduce initial probe waiting time.

Reduction of index data size: Index searching time is related to the size of index data; the smaller the
indexes data size is, the shorter the search time will be. Consequently, the entire broadcast cycle will be shorter, and the average access latency will be smaller. For example, Imielinski et al. [9] only duplicated \( k \) layers of index tree to reduce the size of index data. Hu et al. [8] used the signature capture technique to reduce index data size.

**Effective data placement:** Chen et al. [7] has proved that different broadcasting sequence of different data would affect average access latency of data retrieval, and proposed ORD algorithm to reduce average access latency by arrangement of the sequence of broadcast data according to retrieval frequency. Currently broadcast index studies focus on one single step to enhance efficiency without considering improving the efficiencies of the three steps. This Study has designed a new broadcast index to handle RNN query in broadcasting with considerations for the three steps.

4. A New Index Structure for RNN Query

A RNN query that searches for \( q \) returns a collection of objects of nearest neighbors in relation to \( q \). If we may know the distance between every object and its NN in advance, then all we have to do is to find out the distance between \( q \) and the objects which are closer than that between the objects and its NN, and then the objects are the results for the RNN query that searches for \( q \). The difference between Rdnn-tree and R-tree is that Rdnn-tree stores the information of every object (such as distance of neighbor, or DNN), and it may directly determine whether a leaf node is the result of the query, while R-tree cannot directly determine whether a leaf node is the result of the query and must use branch-and-bound technique, which may cause back-tracking problem. Therefore, we further improve Rdnn-tree with the principles for a better broadcast index that we have proposed to make it an index structure that can effectively support the RNN search in wireless broadcasting environment.

The design of a good broadcast index as mentioned in Section 3 includes three steps: reducing initial probe time, reducing size of index data, and effective placement of broadcast data. We shall explain how we improve Rdnn-tree with these three steps.

**Reduction of initial probe time:** The traditional approach is to increase the possibility of the appearance of index by duplicating index. However, this approach will cause longer broadcast cycle and longer average data access latency. Our approach is to build a Jump Rdnn-tree with our index structure for broadcast data. Data and index will be mixed together and broadcasted based on every sub-tree. After the index of such sub-tree has been broadcasted, the data under the sub-tree will be broadcasted in order to reduce the distance between data and index instead of broadcasting all data after the index broadcasting is completed. Taking Figure 3 as example, the broadcasting sequence is \( B, B1, b1, a, b, c, d, b2, e, f, g, B2, b3, h, i, b4, j, k, \) and \( l \). The cyclic index structure is also adopted. The location of each index node to be visited next is calculated with Depth First Search (DFS) traversal according to depth priority. This approach is also called jump pointer, as shown in Figure 4. Because every index node has a jump pointer, index trees are interlined through jump pointers. Therefore, index tree search may begin from any index node instead of root node. In the same time, index tree search follows pointer sequence, and there will be no back-tracking problem.

![Figure 3: Data structure of Rdnn-tree](image)

![Figure 4: Data Structure of Jump Rdnn-tree](image)

**Reduction of index data size:** As mentioned earlier, we adopt cyclic index structure, replication of index data for the entire broadcast cycle is not required and only one copy of index data is needed. Therefore, the size of index data is very small. Also, if the largest sub-tree at every layer of the structure of a traditional index tree has \( f \) fan-out that represents the index tree needs \( f \) pointers are needed to record the address of every sub-tree. Because of linear access feature of wireless broadcasting environment, MU can only access data in active mode, or skip the current data in doze mode and wait for the next data access in active mode. That is, only two behaviors are available: Next, which proceeds to next action, and Jump, which skips
data retrieving. If the criteria for Jump are not met, then Next takes place and Next behavior does not have to be recorded. Therefore, any search for non-leaf node of a tree requires only keeping one jump pointer, and other f-1 pointers for sub-trees can be eliminated, leading to reduction of index data size.

**Effective data placement:** Query point is produced based on the entire search space. The possibility of the occurrence in every area shall affect searching efficiency. If the corresponding index data for the area where the possibility of the occurrence of query point is higher are broadcasted earlier, then the index search efficiency will be better and if the location of the query point is in uniform distribution. Therefore, the broadcast sequence of a sub-tree is determined according to the area of MBR where the sub-tree is, because the larger the area of MBR is, the higher the possibility of the query occurrence will be.

After Rdnn-tree is improved by the above-mentioned approach, the problem of back-tracking is eliminated, and Rdnn index tree with cyclic structure fits with linear access feature of broadcast better. The detailed algorithm of RNN queries of cyclic broadcast is illustrated in Algorithm 1: RNN-Search-On-Air. $D(q, ptid)$ represents the distance between query point $q$ and objects $ptid$, while $D(q, rect)$ represents the distance between query point $q$ and rectangle $rect$.

**RNN-Search-On-Air (Node $n$, Point $q$)**

Case 1: If $n$ is leaf node then
For all (data-item, dnn) in $n$
If $D(q, ptid) < dnn$, then data-item is the RNN for $q$.

Case 2: If $n$ is non-leaf node, then
For all branch $B = (ptr, MBR, Maxdnn)$ in $n$
If $D(q, rect) < Maxdnn$, then call RNN-Search-On-Air ($B.ptr, q$).
Algorithm 1: RNN Search on Air

5. Conclusions and Future Works

In this Study we have discussed how to effectively organize and deploy data in wireless broadcasting environment and answered questions about RNN query. Based on previous studies, we have summarized the principles for designing broadcast index. Based on these principles, a new index structure, Jump-Rdnn tree that is idea for RNN query in broadcast environment, is designed. Because this Study focuses on location-dependent data access in broadcasting environment and is different from traditional on-demand access model, we only discuss issues concerning static RNN query. We shall further extend to more advanced and more dynamic location-dependent queries.

6. References