HORB-tree: AN EFFICIENT PRIVACY PROTECTION SOLUTION FOR QUERYING SPATIAL DATA IN LOCATION-BASED SERVICES

Nguyen Van Nghiem¹, Vu Dinh Long², Dang Le Nam³

¹Faculty of Information Technology, Vocational College Nº8 Dong Nai, Vietnam  
²Faculty of Information Technology, University of Transport Ho Chi Minh City, Vietnam  
³Faculty of Information Technology, University of Food Industry Ho Chi Minh City, Vietnam  
¹nvannghiem@gmail.com, ²long_cn@hcmutrans.edu.vn, ³namdl@cntp.edu.vn

ABSTRACT: Location-Based Services (LBS) have been providing convenient services and daily related-work information for users based on their locations. However, location information, user identities, and their paths are sensitive information which needs to be protected. In this paper, we propose a HORB-tree that combines aRB-tree index, Memorization algorithm, k-anonymity and obfuscation-based technique to enhance the performance of basic operations and increase the privacy protection.

Keywords: Spatial indexing, GIS, HORB-tree, R-tree, B-tree, aRB-tree.

I. INTRODUCTION

Nowadays, the mobile devices are very popular. It is important that users use location-based services (LBS) in daily work. The locations of an object or a user’s location on the earth is a reference point. This information was specified by using latitude/longitude/proof, or paths, etc. Location-based services are increasingly becoming available that return results relative to the locations of their users [11].

However, all users’ information such as their locations, their identifications, and their paths are very sensitive which need to be protected. Users will be more comfortable and more confident in using these services if they ensure that their location information is safe. To resolve this privacy-preserving problem, a variety of solutions have been suggested. The crucial objective is to hide the users’ personal information but still allow them to use services with acceptable quality.

The main idea of these solutions is to obfuscate users’ location [10] or to anonymize location information [8]. Of those solutions, these algorithms are separated from the database layer. This makes these algorithms go through two phases: retrieving the exact location of the user from the database (phase 1), and then obfuscating this information in the algorithm (phase 2). The two-phase processing consumes long time due to the number of disk access required to retrieve the exact location. In the previous works of [2, 4, 6], the author’s proposed solutions are integrated privacy protection algorithms into index structure. These solutions allow the queries which are processed in only one phase. These solutions could decrease the query processing time. However, users can not prevent sophisticated types of attacks (e.g. Query Sampling Attack) because the obfuscated areas returned by these do not satisfy the memorization. On the other hand, in [3], the author proposed solution of preserving query privacy with a query-based Memorizing algorithm.

Motivated by these solutions, in this paper, we propose a new method that combines the Memorization algorithm and the obfuscation-based technique. Furthermore, we integrate those in the indexed spatial data to gain performance boost. With the proposed solution, the obfuscated areas returned by HORB-tree are satisfying the memorization. This combined solution can prevent sophisticated types of attacks and therefore protects user privacy better than other known solutions. On the other hand, because using pointer is among leaf nodes, parent nodes and children nodes and so decreasing query time and update performance.

The rest of this paper is organized as follows. Section II briefly reviews existing techniques for location privacy protection. Section III presents our solution that is the HORB-tree. Section IV demonstrates the performance testing. Section V gives concluding remarks and introduces future works.

II. RELATED WORKS

2.1. The aRB-tree

In the previous research, there was some spatial data index methods proposed. One of these methods is the R-tree. It is a dynamic index structure for spatial searching. R-tree is used in the spatial database as a spatial access method. R-tree is used with the region and point data. These are tree data structures used for spatial attribute access methods, i.e., for indexing multi-dimensional information such as geographical coordinates, rectangles or polygons. B-tree is tree data structures used for non-spatial attribute access methods.
In [15], the authors proposed the aRB-tree index combining R-Tree and B-Tree to enhance spatial queries processing. The aRB-tree is more efficient and accurate when using individual index structures.

Figure 1. aRB-tree index structure [15]

Figure 1 illustrates the structure of the aRB-tree. In the aRB-tree, the extents of all regions (in this case r1, r2… r4) are stored in an R-tree. Each entry of the R-tree is associated with a pointer to a B-tree that stores historical aggregate data about the entry. The B-tree stores detailed information related to objects in MBRs.

2.2. The Hilbert R-tree

In the spatial database, there was some spatial data index structures proposed. One of these index structures is the R-tree. In the R-tree index, the minimum bounding rectangles (MBRs) can overlap, and it increases as the number of data increases. So, the R-tree index structure decrease query performance. Furthermore, it is imperative to conduct searches at all levels of the R-tree, when no (few) data objects satisfy the requirements.

On the other hand, the R-tree index structure also has the disadvantage of splitting when it inserts new entries. To solve the above problem, in [14], the authors proposed Hilbert R-tree with using space-filling curves (or fractals), and specifically, the Hilbert curve to impose a linear ordering on the data rectangles. Figure 2 và Figure 3 illustrate the structure of the Hilbert R-tree.

Figure 2. Data rectangles organized in a Hilbert R-tree [14]

The Hilbert R-tree contains entries of the form \([LHV, R, Ptr]\). R is the MBR that encloses all nodes’ children, Ptr is a pointer to the children node, and LHV is the largest Hilbert value among the data rectangles enclosed by R.

Following [14] the Hilbert R-tree has two basic advantages. The first, the indexing objects in spatial data based on the Hilbert values are simple processing. The second, Hilbert R-tree decreases node splitting, which insert new entries more than R-tree.
2.3. Location Obfuscation

In [10], the authors propose Obfuscation-Based techniques by enlarging the area containing users’ exact locations (as shown in Fig. 4). The main purpose of this method is to blur the user location information (this location information is associated with the user identities) to protect their privacy.

After that, users’ locations and service requests are sent to the Service Provider. The process of obfuscating user location can be done at devices of users or provided by a third party. With Obfuscation-Based techniques, an attacker can hardly infer users’ exact locations.

![Figure 4. An example of obfuscated location](image)

2.4. K-anonymity Algorithm

In [8], the authors propose K-anonymity algorithm to anonymizing users’ exact locations. A release of data is said to have the k-anonymity property if the information for each person contained in the release cannot be distinguished from at least k-1 individuals whose information also appear in the release. This method must be provided based on a trusted third party. Figure 5 illustrates the centralized model for privacy in LBSs.

![Figure 5. The Centralized model for privacy in LBSs](image)

2.5. Memorization Algorithm

One of the problems needs solving protection users’ query privacy against the overlapped area attack in LBSs. As shown in figure 6, the user queries multiple times at t1, t2, t3, and return results are R1, R2, and R3. The attacker will be able to infer user’s location based on three overlapping areas. So, the possibility of containing the user in the intersection of these areas is very high.

In [3, 10], the author proposes solution to preserving query privacy with a query-based Memorizing algorithm. The anonymization area yielded by the algorithm will satisfy the user’s privacy requirement and also solve the overlapped-area problem. The main idea of memorizing algorithm is to creating new obfuscation regions depends on the obfuscation regions previously. So, obfuscation regions will be contained in spatial databases and it was used to the next query processing time. Specifically, if users require a service in the first time, the obfuscation regions random (COR-candidate obfuscation region) will be returned based on CORs list. Otherwise, it returns the maximum similarity value with the matched obfuscation regions in RecId (set as the similarity value of CORs). According to Memorizing algorithm, the values of similarity are determined by the following formula:

\[
\text{Similarity} = \frac{\text{Matched CORs}}{\text{Total CORs}}
\]
III. THE HORB-TREE

To reduce query processing time and protect users’ locations privacy, the authors in [2, 4, 6] proposed a solution that needs only a one-phase processing by integrating some algorithms into the database level. The authors had proposed new trees: As Bob-tree, Bob+-tree and BobICO-tree. However, in these solutions, the overlapped area problem is not guaranteed so that users can be attacked location privacy.

Meanwhile, in [15], the authors proposed the aRB-tree (aggregate R-tree and B-tree) for the efficient processing of spatio-temporal count queries in LBSs. In the R-tree index, the minimum bounding rectangles (MBRs) can overlap, and it increases as the number of data increases. So, in [14] the authors proposed the Hilbert R-tree and presented algorithms for searching, insertion, deletion, and overflow handling with using space-filling curves in order to avoid this issue. To protect location privacy against such attacks, in this paper, we propose a HORB-tree that combines aRB-tree index (but replace R-tree to Hilbert R-tree), Memoziration algorithm, k-anonymity and obfuscation-based technique to enhance the performance of basic operations and increase the privacy protection.
Figure 8 illustrates the structure of the HORB-tree (with clustering of MBRs in HORB-tree as shown in figure 7). MHV are Hilbert values of HR regions (the largest Hilbert value of children nodes is reviewed). HR are region containing users’ exact locations. LeafList is a pointer to the leaf nodes, parent nodes and children nodes have two-way Pointers.

Figure 9 illustrates an example of HORB-tree. As shown in figure 9, hr1 of the B-tree contains four leaf entries. With the first root of entry hr1<1, 445> of the B-tree, the sum of objects in hr1 (corresponding to [1, 3]) is 445. The value of first time is 150, and second time is 150 (the sum of objects no change, 150 is not node duplicating). Thus, 445 is the sum of 150, 150 and 145. The second of entry hr1<4, 265> of the B-tree, the sum of objects in hr1 (corresponding to [4, 5]) is 265. The first root of entry HR1<1, 225> of the B-tree, the sum of objects in hr1 and hr2 at time 1 is 225. HR1<1,685> of the B-tree, the sum of objects in HR1 (corresponding to [1, 3]) is 685. The first root of entry hr4<1, 259> of the B-tree, the sum of objects in hr4 (corresponding to [1, 2]) is 259. The sum of objects (corresponding to [1, 3]) is 1069 (1069 is the sum of 685, 259 and 125). Our search algorithm will be based on k-Anonymity, the obfuscated region and Memorization algorithm. The following outlines the procedure to search for a record in the HORB-tree:

**HORB-tree Search Algorithm**

**Require:** \( Q \equiv \{id, k, s, l, c\} \)

**Ensure:** Obfuscated region (OR) with at least \( k \) users

If \( Rec_{id} \) is empty then

- Search \( hr \), that contains \( id \) on LeafList list in HORB-tree
- If not found then finish
- While (the number of users in \( hr \) is not satisfied k-anonymity and \( hr \) less than \( s \) do
  - \( hr \) = parent node of \( hr \)
- End while
- \( OR = hr \)

Else

- Foreach \( COR \), in \( Rec \),
  - \( OR = Select \ COR \), such that \( Sim(COR, s) \) is max.

End if
If user need other details then
Call the B-tree search
End if
Store record $Q'' = \{id, id', k, l, OR, c, ID, t\}$ into the database.

The details of the algorithm is described as follows: the first Search algorithm inputs: a $Q(id, k, s, l, c)$ queries with id user’s identification, l users’ exact locations, area of an obfuscated region s, and a k-anonymity value k, c the query contents. Then, it is depending on $Rec_{id}$ records in the database corresponding to id, the search algorithm has two cases: in the first case, if $Rec_{id}$ is empty, the search processing is based on LeafList list. Until regions containing at least k-1 other users are found. Otherwise, the search algorithm finds in parent nodes. In the second case, its search $Rec_{id}$ records in the database corresponding to id. So, COR-candidate obfuscation region has maximum similarity value; the values of similarity are determined by the following formula (2).

Finally, $Q''$ is stored in its database $Rec_{id}$ and forwards $Q''$ to the Location Based Services Provider. $Q''$ record (new user’s identification -id’; k; l; c; obfuscation region – OR; other user’s identification sets –ID; t represents the maximum time period save in record –t). The insert, delete and update operations of the HORB-tree are similar to those of the Hilbert R-tree.

$Sim(HR1, HR2) = \begin{cases} 
\frac{ID_{min} \cap ID_{max}}{k}, & \text{if } (ID_{min} \cap ID_{max}) < k \\
1, & \text{if } (ID_{min} \cap ID_{max}) \geq k
\end{cases}$

(2)

In Equation (2) above, the maximum value of sim ($HR1, HR2$) is 1, when the users in two anonymity sets corresponding to two regions have at least k users. Otherwise, the similarity value will be decided by the number of intersected users in anonymity sets corresponding to $k$.

IV. PERFORMANCE EXPERIMENTS

In this section, we experimentally compare the HORB-tree to the Hilbert R-tree and the aRB-tree in terms of the number of disk accesses. To conduct the experiments, the HORB-tree, the Hilbert R-tree and the aRB-tree are all implemented in C++, and all experiments are performed on a machine with Inter(R) Core(TM)i5-6200U CPU @ 2.30 GHz, ~2.4 GHz, RAM 4 GB, 500 GB HDD, install Windows 7. For all experiments, we use uniform datasets, where objects’ locations are randomly generated. We show the experimental results of not only the cost of querying, update cost and index size but also the degree of the location privacy protection. To create the dataset of paths, we use the Thomas Brinkhoff’s framework for generating network-based moving objects [16]. The network used in the experiment is Oldenburg which is about 102.96 km² [17].

Firstly, we define the grid corresponding with the Oldenburg city. The grid’s size is 60x60 cells. The area of each cell reflects the privacy level. This means that the larger the cell is the more privacy, but the service quality is the less. Secondly, we generate from 100 to 10,000 objects with random coordinate corresponding to the paths of Oldenburg city. To represent the moving of objects, their locations are changed after each interval of five minutes. Finally, we run datasets for 10 times and the results are described as below:
Figure 11. Update cost with varying query set

Figure 10 shows the query performance while varying the query size. As the result, the query cost of the HORB-tree is the lowest in index trees. The query cost of the Hilbert R-tree is higher than HORB-tree due to it can chooses candidate obfuscation region with maximum similarity value in database (the parameters s and k are satisfied user’s request). The query cost of the aRB-tree is higher than HORB-tree due to aRB-tree has the size of the keys relatively large.

Figure 11 shows the update cost with varying query set. As the result, the update cost of the HORB-tree is similar to the Hilbert R-tree due to the same update processing. Otherwise, the update cost of the aRB-tree is higher than HORB-tree due to splitting nodes. This characteristic is also similar to the index size evaluating, as shown in Figure 12.

Figure 12. Evaluating of the index size

Figure 13 illustrates the degree of privacy protection. As a result, HORB-tree always returns the obfuscated region that satisfies $k$-anonymity. However, the service quality is poor if the density of users is low (when that the obfuscated region is large). This disadvantage can be enhanced by using Dummy technique [11].
V. CONCLUSION AND FUTURE WORK

A variety of techniques are applied to privacy protection in location-based services depend on the different requirements. Therefore, it is hard to have a common solution for all. In this paper we proposed an integrated solution for data indexing by using HORB-tree which provides a better privacy degree.

On the other hand, because of including the Memorization characteristic of the Memorization algorithm, the solution can protect users’ locations from Query Sampling Attacks. The new blurred region will still attain the minimum area and k-anonymity that the users define in his privacy profile. In addition, HORB-tree considerably enhances the performance of querying.

In the future, we will conduct further study to integrate other algorithms into the spatio-temporal data index structure for user location privacy protection. We will also focus on the problem of quality of services with respect to the HORB-tree and other access methods.

VI. REFERENCES


HORB-tree: MỘT GIÁI PHÁP HIỆU QUẢ BẢO VỆ TÍNH RIÊNG TỰ KHI TRUY VẤN DỮ LIỆU KHÔNG GIẢN TRONG LBS

Nguyễn Văn Nghịêm, Vũ Đình Long, Đặng Lê Nam

TÓM TẮT: Dịch vụ dựa trên vị trí, viết tắt là LBS (Location Based Service), đã cung cấp các dịch vụ tiện ích và thông tin liên quan đến công việc hàng ngày của người dùng dựa trên vị trí của họ. Tuy nhiên, các thông tin vị trí, các thông tin danh và đường đi của người dùng là những thông tin nhạy cảm cần được bảo vệ. Trong bài báo này, chúng tôi đề xuất HORB-tree kết hợp giữa cấu trúc chỉ mục aRB-tree, thuật toán Memoization, k-anonymity và kỹ thuật Obfuscation để bảo vệ tính riêng tư khi truy vấn dữ liệu không gian trong LBS. Hơn nữa, HORB-tree giúp cải thiện tốc độ truy vấn dữ liệu và các thao tác cơ bản khác.

Từ khóa: Spatial indexing, GIS, HORB-tree, R-tree, B-tree, aRB-tree.