# Bichromatic Reverse Nearest Neighbor Query without Information Leakage

Lu Wang[1], Xiaofeng Meng[1](✉), Haibo Hu[2], and Jianliang Xu[2]

[1] School of Information, Renmin University of China, Beijing, China
{luwang,xfmeng}@ruc.edu.cn

[2] Department of Computer Science, Hong Kong Baptist University,
Hong Kong, China
{haibo,xujl}@comp.hkbu.edu.hk

**Abstract.** Bichromatic Reverse Nearest Neighbor (BRNN) Query is an important query type in location-based services (LBS) and has many real life applications, such as site selection and resource allocation. However, such query requires the client to disclose sensitive location information to the LBS. The only existing method for privacy-preserving BRNN query adopts the cloaking-region paradigm, which blurs the location into a spatial region. However, the LBS can still deduce some information (albeit not exact) about the location. In this paper, we aim at strong privacy wherein the LBS learns nothing about the query location. To this end, we employ private information retrieval (PIR) technique, which accesses data pages anonymously from a database. Based on PIR, we propose a secure query processing framework together with various indexing and optimization techniques. To the best knowledge, this is the first research that preserves strong location privacy in BRNN query. Extensive experiments under real world and synthetic datasets demonstrate the practicality of our approach.

**Keywords:** Privacy preservation · Location privacy · Private information retrieval · Bichromatic RNN

## 1 Introduction

Given two point sets $S$ (the servers) and $R$ (the objects), and a server $q \in S$, a bichromatic reverse nearest neighbor (BRNN) query finds the set of objects whose nearest server is $q$. BRNN has been receiving increasing attention since the boom of mobile computing and location-based services (LBS). It has numerous

X. Meng – This research was partially supported by the grants from the Natural Science Foundation of China (No. 61379050, 91224008); the National 863 High-tech Program (No. 2013AA013204); Specialized Research Fund for the Doctoral Program of Higher Education(No. 20130004130001); the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University(No. 11XNL010);HK-RGC GRF grants HKBU12200914 and HKBU 210612;HK-RGC Grants HKBU211512 and HKBU12200114.

M. Renz et al. (Eds.): DASFAA 2015, Part I, LNCS 9049, pp. 609–624, 2015.
DOI: 10.1007/978-3-319-18120-2_35

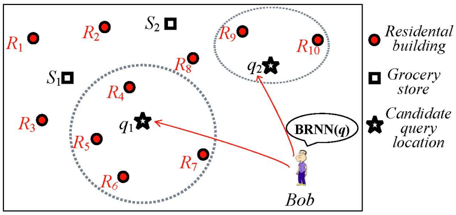


**Fig. 1.** Example of BRNN query

applications in map search, resource allocation, emergency service dispatching, military planning, and mobile reality games [3]. Figure 1 illustrates two sets of points of interest (POIs) from an online map service, where red circles are residences $R_i$ and black squares are grocery stores $S_i$. Bob has a few candidate locations $q_i$ (the black star) to open up a new store, so he wants to know which location can attract the most residences from existing stores based on distance. By issuing a BRNN query at each candidate location, he is able to tell $q_1$ is the best location to open a new grocery store, as it leads to the largest BRNN results — four residences $R_4$, $R_5$, $R_6$ and $R_7$.

However, the query location as well as Bob's business intention has been disclosed to the server during this process. Such privacy disclosure also occurs in other BRNN application scenarios. For example, in taxi dispatching, a taxi driver has to report the cab's current location in order to know the customers to whom he/she is the nearest to serve. However, such location can reveal sensitive information about the passenger, such as his/her medical or financial condition, as well as political affiliations [2]. Therefore, protecting the query input of a BRNN query against the LBS is indispensable. In the literature, the only existing approach for privacy-preserving BRNN query adopts the cloaking-region paradigm [3], which sends to the LBS a spatial region that contains the query point. Based on this region, the LBS returns a superset of the genuine reverse nearest neighbors, from which the client user will refine the true result. Unfortunately, this approach still reveals to the LBS substantial information about the location.

To guarantee strong location privacy, a promising cryptography tool is private information retrieval (PIR) [16]. PIR allows a data item (e.g., a disk page) to be retrieved from a server without leaving any clue of the item being retrieved. PIR was considered to be resource-intensive, but thanks to the recent progress in cryptography, practical software or hardware PIR solutions have been proposed [4]. Since then it has been successfully applied to database problems, such as $k$NN and shortest path search [6] [8].

In this paper, we investigate privacy-preserving BRNN query without the LBS deducing any information about the query point. To this end, we adopt practical PIR techniques that retrieve a single data page as the building block. The challenges of a PIR-based BRNN solution lies in the following aspects: (1) although PIR guarantees secure access of a single page from the server, the variation of the number of page accesses from different queries may reveal information about the query point, and (2) as the database contains voluminous points, directly applying PIR for the BRNN query is inefficient, thus calling for an integration with spatial index, such as KD-tree. To address these challenges, we first propose a PIR-based BRNN query processing framework that guarantees strong privacy. We then apply to this framework two indexing schemes, whose performance varies with the data distribution. An orthogonal optimization technique is also proposed to further enhance the performance. To summarize, our main contributions are:

(1) To the best knowledge, this is the first work on BRNN query processing with no information leakage.

(2) We propose a framework for PIR-based BRNN query and prove its security.

(3) We design two indexing schemes for different data distributions, and propose an optimization to further bring down the transformation cost.

(4) We conduct extensive experiments under real-world and synthetic datasets, which shows our proposed approach is practical.

The rest of the paper is organized as follows. Section 2 reviews the related work. In Section 3, we formalize the system model and problem definition. In Section 4, we present the framework for the PIR-based BRNN query processing, followed by two indexing schemes based basic methods, namely KD-tree based method and Adaptive grid based method in Section 5. We then propose an optimization in Section 6. The solutions are evaluated by experiments in Section 7.

## 2 Related Work

In this section, we review existing literature on bichromatic reverse nearest neighbor query and private information retrieval.

### 2.1 Bichromatic Reverse Nearest Neighbor

In light of its critical applications ranging from social life domain such as location selection to military activities such as the placement of food [9], bichromatic reverse nearest neighbor query (BRNN query) attracts considerable attention since its first seminal work [12]. To efficiently find the BRNNs for a query point, Voronoi polygon is widely used under various circumstances such as static or continuous query processing [14] [13]. In these works, the Voronoi polygon determines candidate or accurate region for the query point, within which object points are the query point's BRNN result. However, all these queries have not considered the privacy issue of disclosing the plaintext query location to the LBS. There is only one recent work addressing this issue [3]. In this work, the client issues a query region instead of a point to the server, and the server returns object points that are BRNNs to every point in the region. The client then refines the actual BRNNs based on his/her actual location point. While this solution still exposes a query region, our work supersedes it by revealing no location information of the query point.

### 2.2 Private Information Retrieval

Prior to PIR-based methods, data transformation based methods are considered to provide strong location privacy. [11] presents a model that adopts Hilbert mapping to transform the location data. Such transformation encrypts coordinates in a way that preserves distance proximity and thus can be applied for approximate nearest neighbor query or range query. [10] proposes a secure transformation to guarantee the approximate distances of POIs to the query point and answers $k$NN query. However, these methods are vulnerable to exposing relative distance [1] or access pattern attack[8].

Thanks to the advances of modern hardware and distributed/cloud computing, PIR has become a viable solution to oblivious data page access in malicious server [5]. However, it is not trivial to apply it to privacy-preserving location queries, because the processing for different queries incurs different numbers of PIR access, which may be exploited by adversaries to induce the query location [7]. Existing PIR-based methods includes PIR-based NN query[7], $k$NN query[6] and shortest path computation[8]. To guarantee equal number of PIR access for any query point, all these methods imposes the maximum number of PIR access on the dataset. Their main objective, therefore, is to design elaborate data structures (e.g., grid file or KD-tree) that decompose the space to reduce this number. Nonetheless, no existing work has been on applying PIR-based method to BRNN query.

## 3 Problem Definition

In this section we present the system model and formally define the problem as well as the security model.

### 3.1 System Model

Figure 2 illustrates the system model in this paper. The server (LBS) owns two POI datasets, namely, the server points $S$ and object points $R$. The client issues a bichromatic reverse nearest neighbor (BRNN) query $q \in S$, for which the server returns the set of objects whose nearest server is $q$. Formally, $BRNN(q) = \{r \in R | \forall s \in S : dist(r, q) \leq dist(r, s)\}$. As for the privacy requirement, the server should not learn any information about $q$.

To enable privacy protection, a naive solution is to ship both $S$ and $R$ datasets to the client for processing. However, due to their large volume and dynamic nature, this solution cannot scale well. Thanks to the recent advances in private information retrieval (PIR), we adopt the state-of-the-art hardware-based PIR as follows. The server installs a secure co-processor ($SCOP$), which offers unobservable and unmolested computation inside an untrusted hosting device. The $SCOP$ performs a hardware-based PIR protocol with the client, and offers the latter oblivious access of a data page [4]. With the $SCOP$, we propose a general secure processing framework for spatial queries, which is composed of

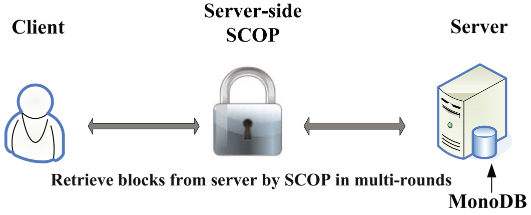


**Fig. 2.** System architecture

multi-round PIR access to a database $MonoDB$. It is a monolithic database that integrates both the datasets and indexes. In each round, the client retrieves a specific page of $MonoDB$ through $SCOP$. The fetched data helps the client determine the next page to retrieve, and the procedure repeats until the BRNN query is answered. Therefore, the secure query processing problem is reduced to the efficient design of the $MonoDB$ and the associated retrieval plan.

### 3.2 Adversary and Security Model

**Adversary.** The adversary in our problem is the LBS server. As a common assumption in private information retrieval, the computational power of the adversary is polynomially bounded.

**Security Model.** Our objective is to develop practical protocol for processing BRNN query without the LBS deducing any location information about the queries. Similar to [6], we assert that every BRNN query follows the same retrieval plan, which is necessary in order to achieve our privacy goal. Specifically, we ensure that every query (1) executes in the same number of rounds in the same order and (2) in each round it retrieves the same number of data pages. The retrieval plan is determined by the processing protocol and is publicly available. For example, if the protocol states that in the second round, 5 pages are fetched from the database, then every query must fetch 5 pages from database in the second round. If a query needs fewer than 5 pages, the protocol pads with dummy page requests. Since each invocation of PIR is secure, we can naturally reach the following theorem regarding the security of our proposed framework.

**Theorem 1.** *The BRNN query processing framework leaks no information to the adversary about query location. Equivalently, from the adversary's perspective, every query is indistinguishable from any other.*

## 4 Private BRNN Processing Framework on MonoDB

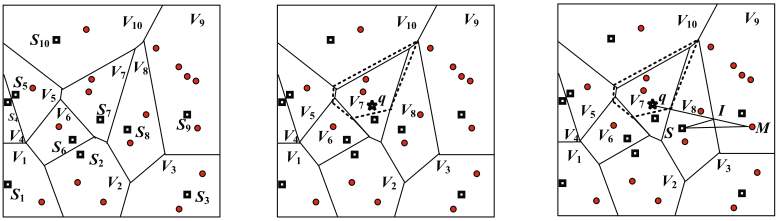
In this section, we overview private BRNN query processing in the proposed $MonoDB$ framework. Recall that in this framework, any query processing is equivalent to a multi-round retrieval of data pages of the $MonoDB$. The $MonoDB$ can be logically split into $n$ databases $DB_1, DB_2, ..., DB_n$, where $DB_i$ $(1 \leq i \leq n-1)$ are indexes and $DB_n$ is the object database $R$. The retrieval sequence of the query can also be split accordingly as $[c_1, c_2, ..., c_n]$, where the client fetches a set of pages $c_i$ from $DB_i$ $(1 \leq i \leq n)$. In this section, we first present a baseline BRNN processing algorithm, based on a key observation that reduces the number of servers and objects to retrieve for the query evaluation. Based on this algorithm, we describe the detailed $MonoDB$ design and retrieval plan.

### 4.1 Baseline BRNN Processing

There are several existing BRNN query processing methods in the literature. In the Voronoi Diagram-based method, a Voronoi Diagram is constructed for all server points and the query point $q$, and the object points that are in $q$'s Voronoi cell are the BRNN results. However, this method cannot be directly applied in our framework as the query point $q$ is dynamic and cannot be learnt by the LBS. Nonetheless, we observe that the Voronoi Diagram of the server points gives a nice bound of the result objects.

Figure 3(a) illustrates the Voronoi Diagram for all server points (they are called "seeds" in some literatures). For each seed, any object point in its corresponding Voronoi cell is closer to it than to any other seed. When a query $q$ is issued, $q$ is added to the set of seeds, and the Voronoi Diagram is updated as in Figure 3(b). Compared with these two diagrams, we observe that changes only occur in Voronoi cells $v_5$,$v_6$,$v_7$,$v_8$, and $v_{10}$, where $q$ is in $v_7$. In other words, the Voronoi cell for the query point $q$ only depends on the seed $v_7$ and its neighboring seeds in the original Voronoi diagram. As a result, only object points in these Voronoi cells can reside in the new Voronoi cell of $q$. Therefore, the BRNN results can be bounded by those object points in the Voronoi cells that contains $q$ and its neighboring cells. This observation can be formally stated as below.

**Theorem 2.** *Given query point $q$ and the Voronoi diagram of all server points, any object point $M$ outside of the Voronoi cell that contains $q$ and its neighboring cells cannot be $q$'s BRNN.*



(a) Voronoi Diagram by server points

(b) Voronoi Diagram by server points plus query point

(c) Proof for the observation

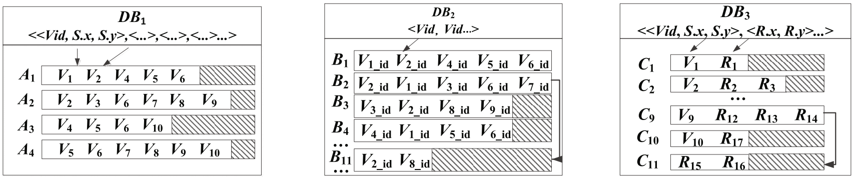
**Fig. 3.** Voronoi Diagram for server points and query point

*Proof.* As Figure 3(c) illustrates, the segment between $q$ and $M$ must cross some Voronoi cell that neighbors with $q$ at point $I$. Suppose that the seed of this cell is $S$. The distance between $q$ and $M$ equals to the segment length between $q$ and $M$, which is equal to $dist(M, I) + dist(I, q)$. Since $I$ is $S$'s BRNN, it must hold that $dist(I, S) < dist(I, q)$. Further, according to the triangle inequality, $dist(M, S) < dist(M, I) + dist(I, S)$. Therefore, $dist(q, M) = dist(M, I) + dist(I, q) > dist(M, I) + dist(I, S) > dist(M, S)$. As such, the distance between $q$ and $M$ is larger than the distance between $M$ and $S$, which means $M$ cannot be $q$'s BRNN.

By Theorem 2, the BRNN processing protocol between the client and LBS is as follows. The LBS first computes the Voronoi Diagram of all server points offline. When query $q$ arrives, it sends to the client (1) the servers (i.e., seeds) whose Voronoi cell contains $q$ or is a neighbor of it; and (2) all object points in the corresponding cells of these seeds. The client then refines the BRNN results by verifying among these seeds if $q$ is the nearest neighbor to each object point.

### 4.2 MonoDB Design and Retrieval Plan

**Three Databases.** Based on the above baseline BRNN query processing algorithm, the $MonoDB$ can be split into three logical databases as illustrated in Figure 4: $DB_1$ stores all the Voronoi cells, $DB_2$ records the Voronoi neighbors of each cell, and $DB_3$ stores object points of each cell. Note that $DB_1$ implies a space partition, from which only the relevant Voronoi cells need to be retrieved. The partition is non-overlapping so that only one record in $DB_1$ will be retrieved for any query $q$. The detailed partition algorithms are discussed in the next section.


DB1
<<Vid, S.x, S.y>,<...>,<...>,<...>...>
A1 V1 V2 V4 V5 V6
A2 V2 V3 V6 V7 V8 V9
A3 V4 V5 V6 V10
A4 V5 V6 V7 V8 V9 V10
DB2
<Vid, Vid...>
B1 V1_id V2_id V4_id V5_id V6_id
B2 V2_id V1_id V3_id V6_id V7_id
B3 V3_id V2_id V8_id V9_id
B4 V4_id V1_id V5_id V6_id
...
B11 V2_id V8_id
...
DB3
<<Vid, S.x, S.y>, <R.x, R.y>...>
C1 V1 R1
C2 V2 R2 R3
...
C9 V9 R12 R13 R14
C10 V10 R17
C11 R15 R16

(a) Example for $DB_1$ (b) Example for $DB_2$ (c) Example for $DB_3$

**Fig. 4.** Examples for the three index structures

**Retrieval Plan.** Given $MonoDB$ and a BRNN query $q$, the PIR retrieval plan is as follows. The client first accesses $DB_1$ for the record of the partition where $q$ is located. This record stores the coordinates of all seeds in this partition, so that the client can compute their distances to $q$ and finds the seed $i$ that is the closest to $q$. Then the client accesses $DB_2$ for the record of $i$, which stores $i$'s Voronoi neighbors. According to Theorem 2, the Voronoi cell of $q$ can be derived from $i$ and its Voronoi neighbors. So the client accesses $DB_3$ for the records of $q$ and $q$'s Voronoi neighbors. These records store all object points that are in these Voronoi cells. A final refinement step is needed to remove from the results those objects outside of the Voronoi cell of $q$.

Figure 4 illustrates the $MonoDB$ for the whole space and the server points of Figure 3(c). If the query $q$ is issued at the star point in Figure 3(c), it will be located in record $A_4$ in $DB_1$, where we can find $v_7$ is the closest to $q$. So we obtain $v_7$'s Voronoi neighbors from record $B_7$ of $DB_2$, i.e., server points $v_2, v_5, v_6, v_8, v_{10}$. We then access records $C_2, C_5, C_6, C_7, C_8, C_{10}$ from $DB_3$. These records give us the candidate result objects such as $R_2, R_3$ that are further refined by the client.

**Rationale of Three Databases.** Splitting the $MonoDB$ into three logical databases has a variety of benefits: (1) it decouples the server and object points so that the update in one dataset will not significantly change the $MonoDB$; (2) it removes redundancy information and thus enhances the PIR performance. For example, if $DB_1$ and $DB_2$ were merged into $DB_1'$, there would be a lot of common neighboring seeds in different records of $DB_1'$.

**Overflow and Underflow Handling.** Normally a record spans a single page of a database. If it is not full, it will be padded with dummy data. On the other hand, if a record overflows in any database $DB_i$, the LBS creates extra pages and appends them at the end of $DB_i$. These pages are chained up by the overflow pointer at the end of each page, e.g., $B_2$ with $B_{11}$ in $DB_2$ and $C_9$ with $C_{11}$ in $DB_3$. In what follows, we use $cnt_i$ to denote the maximum number of pages for a single record in $DB_i$.

## 5 Spatial Partition

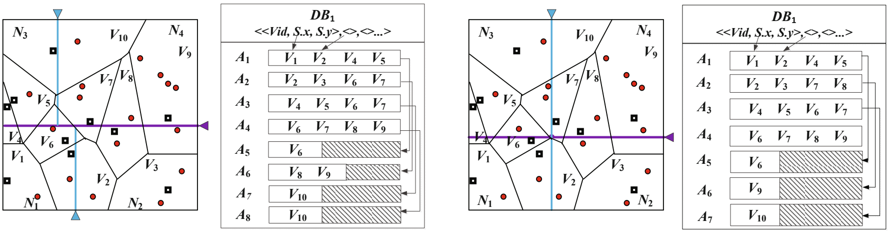
In our $MonoDB$ framework, while $DB_2$ and $DB_3$ depend only on the two datasets, $DB_1$ also depends on the space partition scheme. In this section, we present two space partition algorithms, which leads to two different indexes for $DB_1$.

### 5.1 KD-tree Partition

KD-tree is a widely adopted method for space partition due to its at least 50% space utilization. In what follows, we show how to construct the $DB_1$ based on a KD-tree.

Figure 5(a) illustrates that a KD-tree partition of the space over server points, which produces four node: $N_1, N_2, N_3$ and $N_4$. Each node contains a minimum of 2 and a maximum of 2*2-1=3 seeds. As such, $DB_1$ has four records $N_1$ through $N_4$, each of which stores the seeds whose Voronoi cell overlaps with this node. Note that we set each data page can hold 4 seeds, so each record spans two pages in Figure 5(a).

Algorithm 1 illustrates the BRNN query evaluation routine. First, the partition that covers the query point is obtained (Line 2) and the corresponding



(a) KD-tree based partitioning (b) Adaptive grid based partitioning

**Fig. 5.** The two partition schemes

**Algorithm 1** KD-Tree Based Method

```
Require: Three databases $DB_1$, $DB_2$ and $DB_3$, the index of KD-tree, query point $q$, $QP$
Ensure: Reverse nearest neighboring object points of $q$
 1: $result = \emptyset$
 2: $leafnode = kd.search(q)$
 3: Fetch the seeds $Cs$ of record $leafnode$ from $DB_1$ by $cnt_1$ PIR accesses
 4: $c = argmin_{i \in Cs} dist(i, q)$
 5: Fetch all neighboring Voronoi cells $Nc$ of record $c$ from $DB_2$ by $cnt_2$ PIR accesses
 6: Construct Voronoi Diagram $VD$ according to $Nc \cup q \cup c$
 7: $region = VD.q$
 8: for each Voronoi Cell $i \in VD$ do
 9:     Fetch all object points $O$ belonging to the record $i$ from $DB_3$
10:     for each object point $o \in O$ do
11:         if $o$ is contained in $region$ then
12:             $result = result \cup o$
13: return $result$
```

record in $DB_1$ is fetched (Line 3). The client then finds out the server point $c$ whose Voronoi cell contains the query point (Line 4). Next, we compute the Voronoi cell of $q$ (Line 5-7). Finally, those candidate object points that fall in the Voronoi cell of $q$ are the BRNN results (Line 8-13).

### 5.2 Adaptive Grid Partition

The disadvantage of KD-tree partition for $DB_1$ is the non-uniform distribution of record size. Although the number of server points in each partition is almost uniform, the number of points whose Voronoi cells overlap each partition is not. As such, a record in $DB_1$ may span too many pages and thus degrades the PIR retrieval performance. In what follows, we present an adaptive-grid based method that addresses this issue.

The motivation is to have fine-granularity partition over regions with dense Voronoi cells and coarse-granularity partition over regions with sparse Voronoi cells. In this way, it can avoid a single record in $DB_1$ that hold too many seeds.

Specifically, this method partitions the whole space into an $n \times n$ grid in an adaptive manner as follows. It first finds $n-1$ vertical lines one by one that partition the space into $n$ grid cells. Each time, a vertical line is found to minimize the difference of number of Voronoi cells overlapping with both cells. This is achieved by using the standard plane sweep algorithm. Then, it similarly finds $n-1$ horizontal lines that further partition each one of the $n$ cells into $n$ sub-cells. In Figure 5(b), there are only $2 \times 2$ grid cells, so only one grid line is needed for each dimension.

## 6 Optimization

In this section, we present an orthogonal general optimization to the two basic indexing methods by packing small records in $DB_3$ into one page. Note that the default placement for records in $DB_3$ assigns every record a different page, which suffers from low utilization and leads to inefficient PIR access that only fetches very few useful data from a page. Therefore, we propose to pack those records of $DB_3$ if they correspond to the same record in $DB_2$, as these records are always retrieved altogether. As a result, the PIR access of both $DB_2$ and $DB_3$ will be more efficient. Figure 6(a) illustrates the default placement that requires 2 PIR accesses to fetch object points for any query, whereas Figures 6(b) and 6(c) illustrate two example packing results. In Figure 6(b), only 1 PIR access is needed to fetch object points of $DB_3$ for any query. By contrast, 2 PIR accesses are still needed for any query if the records are packed as in Figure 6(c).

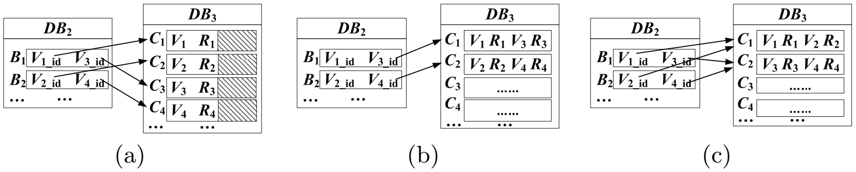


**Fig. 6.** Example for packing records of $DB_3$

Let $N_{DB_2}$, $N_{DB_3}$ denote the number of records in $DB_2$ and $DB_3$, respectively. Let $e_i^{DB_2}$ denote the $i$-th record in $DB_2$, and $\{e_1^{DB_3}, e_2^{DB_3}, ..., e_m^{DB_3}\}$ the $m$ records that corresponds to $e_i^{DB_2}$. Further, let $B_m$ denote the size of $e_m^{DB_3}$; since it might span multiple pages, and only the last page requires packing, the actual packing size $b_m$ is the fraction part of $B_m$, i.e., $b_m = B_m \% Page\_Size$. Then, the problem of record packing can be formalized as follows:

**Definition 1.** ***Record Packing Problem.*** *To pack records in $DB_3$ into data pages, so that $max_i \sum_{j=1}^{m} B_j$ for $\forall e_i^{DB_2}$ is minimized.*

The following theorem shows that this problem is NP-hard.

**Theorem 3.** *Record packing problem is NP-hard.*

*Proof.* This problem can be reduced from the "bin packing" problem. The aim of the latter is to find the fewest number of pages to accommodate a total of $m$ items, each of which is smaller than a page. We reduce a bin packing problem to our problem as follows. We create a single record in $DB_2$ which contains all server points. Then each item in the bin packing problem is mapped to a server point and the size of the item equals to the number of object points for this server point. It is obvious that this straightforward mapping is polynomial, thus completing the proof.

To design an approximation algorithm, in what follows we first present an integer programming solution to the problem, and then relax it to a linear programming problem.

Let variable $y_{m,j} \in \{0,1\}$ denote whether record $e_m^{DB_3}$ is stored in page $j$ of $DB_3$, and $x_{i,j} \in \{0,1\}$ denote whether any record $e_m^{DB_3} \in e_i^{DB_2}$ is stored in page $j$. Formally, we have $\forall e_m^{DB_3} \in e_i^{DB_2}$, $x_{i,j} \geq y_{m,j}$. And $\sum_{j=1}^{P} y_{m,j} = 1$, where $P$ is the number of data pages in the default placement for $DB_3$.

With these variables defined, the number of PIR accesses for object points for a record in $DB_2$ is the number of full data pages of corresponding object points in $DB_3$ plus the packed size for this record. That is,

$$e_i^{DB_2} = \sum_{e_m^{DB_3} \in e_i^{DB_2}} \lfloor B_m / Page\_Size \rfloor + \sum_{j=1}^{P} x_{i,j}$$

Finally, the total number of object points in a page should not exceed the page capacity. That is, $\sum_{m=1}^{N_{DB_3}} b_m y_{m,j} \leq Page\_Size$. Let $K$ be the maximum number of PIR accesses for any record in $DB_2$. Therefore, we reach the following integer programming problem for $K$ as follows:

$$\begin{array}{lll}
\text{minimize} & K & \\
\text{subject to} & \sum_{e_m^{DB_3} \in e_i^{DB_2}} \lfloor \frac{B_m}{Page\_Size} \rfloor + \sum_{j=1}^{P} x_{i,j} \leq K, & \forall 1 \leq i \leq N_{DB_2} \\
 & \sum_{m=1}^{N_{DB_3}} b_m y_{m,j} \leq Page\_Size, & \forall 1 \leq j \leq P \\
 & x_{i,j} \geq y_{m,j}, \forall 1 \leq i \leq N_{DB_2}, & \forall e_m^{DB_3} \in e_i^{DB_2} \\
 & \sum_{j=1}^{P} y_{m,j} = 1, & \forall 1 \leq m \leq N_{DB_3} \\
 & & x_{i,j}, y_{m,j} \in \{0,1\}
\end{array}$$

The above integer programming problem can be approximately solved in polynomial time in two steps. First, one can solve a linear relaxation of the problem, where $x_{i,j}$ and $y_{m,j}$ is a fraction in $[0,1]$. In this regard, $y_{m,j}$ serves as the probability of placing record $e_m^{DB_3}$ into data page $j$, and $x_{i,j}$ serves as the probability of records corresponding to $e_i^{DB_2}$ being placed into data page $j$. As the second step, we adopt the randomized rounding strategy to obtain a feasible solution as follows. We assign object points in the $m$-th record of $DB_3$ to the $j$-th page with probability $y_{m,j}$. If the page overflows, we will assign new empty pages until all object points in this record can be accommodated.

## 7 Experimental Evaluation

In this section, we conduct experiments under real world and synthetic datasets to demonstrate the effectiveness of our PIR-based BRNN algorithm. We also compare the performance with a weaker location privacy preservation approach — the cloaking-based PARNN method [3] and show our algorithm is of great practical value. We also carry out experiments to analyze the effect of our optimization approach.

### 7.1 Experiment Settings

The real world dataset is collected from Open Street Map[1], with location data from Boston and New York, respectively. Both datasets have relatively uniform distribution, while there are more points in New York than in Boston.

As for the synthetic dataset, we vary the number of server points from $10^5$ to $10^6$, and object points are 10 times those of server points. To emulate a skewed distribution, these points are generated by a widely adopted benchmark defined by Chen et al. [15]. In this benchmark, a portion $f \in (0, 1]$ of points are generated in a skewed way to capture object clusters while the rest $1 - f$ portion of points are uniformly generated. Specifically, the portion $f$ of the points are controlled by another skewed parameter $s$ and are generate not far from one of the $s$ randomly selected server points. Table 1 summarizes the detailed parameters of the datasets.

The two indexing methods are implemented with the optimization in Section 6 in place. All codes are written in C# and run on a machine with an Intel Core2 Quad CPU 2.53Ghz and 4 GByte of RAM. We also adopt the open source GNU Linear Programming Kit[2] as the solver the record packing problem in Section 6. As with previous hardware-based PIR methods, we assume the IBM 4764 PCI-X Cryptographc Coprocessor as the SCOP and strictly simulate its performance. The client communicates with the LBS using a link with round trip time of 700ms and bandwidth 384 Kbit/s, which emulates a moving client connected via a 3G network.

**Table 1.** Summary of Experimental Settings

| Dataset | The number of server points | The number of object points |
|---|---|---|
| Boston | 8381 | 146207 |
| New York | 126900 | 1462057 |
| Synthetic | 10000-1000000 | 100000-10000000 |

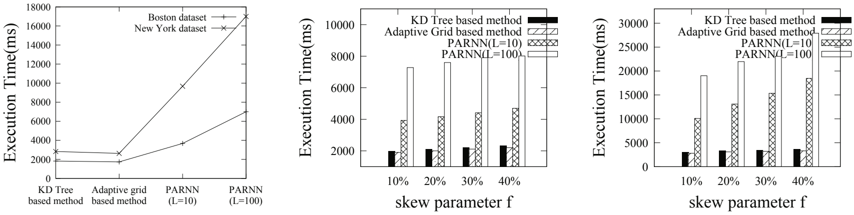
[1] www.openstreetmap.org

[2] https://www.gnu.org/software/glpk/

### 7.2 Performance Comparison

In this section, we compare the performance of our PIR-BRNN method with the PARNN method under both real world and synthetic datasets. The latter method fetches all Voronoi cells that overlap with the client-issued cloaking region, and then returns to the client all object points that are covered by these cells. Note that the performance of PARNN is plotted only for reference, as it still discloses a cloaking region to the LBS.

Figure 7(a) illustrates that PIR-BRNN method outperforms PARNN method with different cloaking region size by a factor of $2-4$ in terms of execution time. We can see that when the cloaking region size shrinks from $100 \times 100m^2$ to $10 \times 10m^2$ (in practice, from a plaza to a road crossing), the execution time for PARNN can improve by about 50%, because fewer server points and object points will be accessed by PARNN. Nonetheless, it still takes more than 2 times the execution time than our proposed PIR-BRNN method.



(a) Real World Datesets (b) $N_s = 10^5, s = 10$ (c) $N_s = 10^6, s = 10$

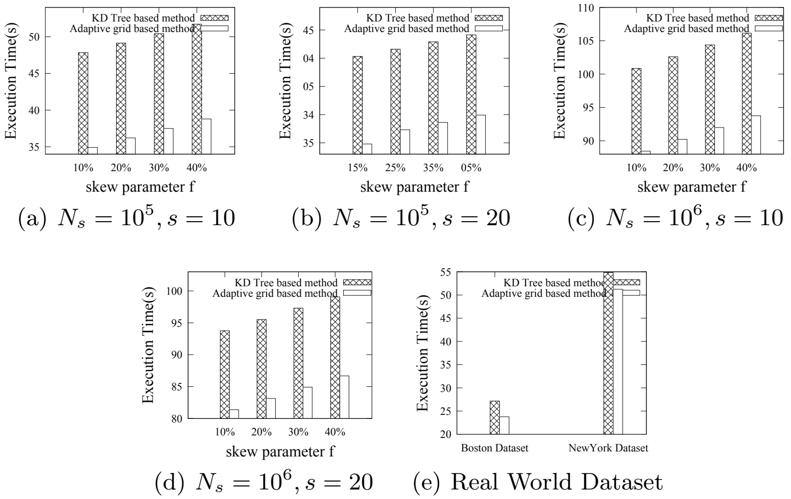
**Fig. 7.** Performance Comparison between PIR-BRNN and PARNN

In synthetic datasets, the performance of PIR-BRNN approach deteriorates as more dummy PIR accesses need to be carried out due to the skewed data distribution. However, Figures 7(b) and 7(c) illustrate that the performance of PIR-BRNN approach is still better than PARNN. The experimental results show that our PIR-BRNN approach is superior to the PARNN method by providing stronger privacy guarantee as well as faster query result time.

### 7.3 Effect of Space Partitions

In this experiment, we evaluate the effect of partition scheme without any optimization[3]. Figure 8(e) illustrates the performance under the real world dataset, where two methods have similar performance. However, as the point distribution becomes more and more skewed in Figures 8(a)-(d), the adaptive grid based method significantly outperforms KD-tree based method. This result coincides with our analysis in Section 5.2 that while the KD-tree keeps each partition

[3] The real SCOP only has 32MB of main memory and can only support up to 2.5GB addressable space. To enable the experiment in this subsection, however, we simply assume there are enough memory buffer in the SCOP emulator.



(a) $N_s = 10^5, s = 10$ (b) $N_s = 10^5, s = 20$ (c) $N_s = 10^6, s = 10$

(d) $N_s = 10^6, s = 20$ (e) Real World Dataset

**Fig. 8.** Performance under two basic methods

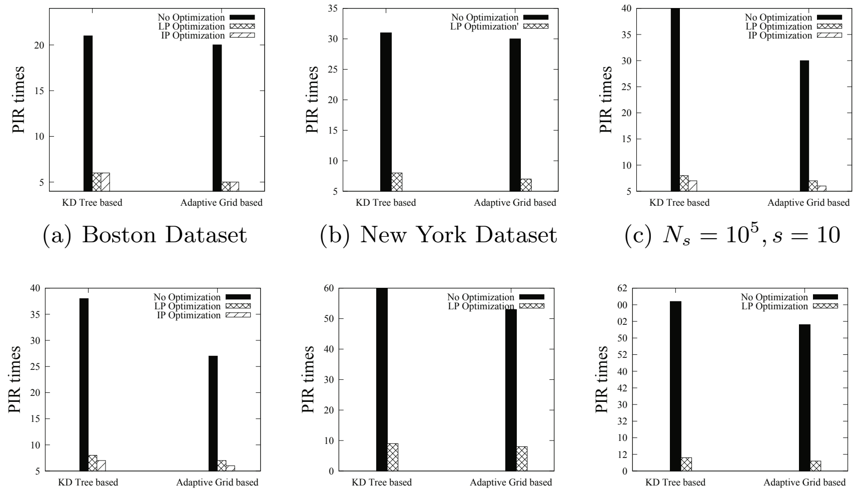
approximately equal number of points, it fails to keep each partition approximately equal number of overlapped Voronoi cells. Therefore, as the server points become skew, it suffers from more overflow pages for dense records in $DB_1$, and thus incurs unnecessary PIR accesses for these extra data pages.

### 7.4 Effectiveness of Optimizations

In this subsection, we evaluate the effect of the optimization proposed in Section 6. First, we show that the number of PIR accesses in both indexing methods is reduced significantly by the optimization. Then we show that although the linear programming (LP) based optimization does not yield the optimal packing of records, it runs much faster than the integer programming (IP) based optimization while still leads to reasonable performance.

Figure 9 illustrates that under real world dataset, both IP and LP based optimization reduce the number of PIR accesses by more than 70% on average. In particular, the effect of our optimization is most significant for skewed data distribution. This is because many server points have a lot of corresponding records in $DB_3$ with only very few object points; as they are packed together, the maximum number of PIR accesses is greatly reduced.

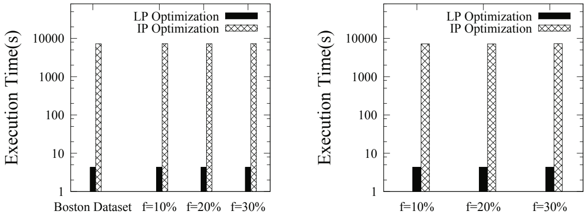
It is worth noting that although the LP based optimization dose not yield the optimal packing of records, it achieves comparable performance as the optimal IP-based optimization. On the other hand, Figure 10 illustrates that the running time of the former is much faster and is thus more practical than the latter. In fact, in our experiment we cannot complete the IP based optimization on the New York dataset or any synthetic dataset with more than $10^6$ server points.



(a) Boston Dataset (b) New York Dataset (c) $N_s = 10^5, s = 10$

(d) $N_s = 10^5, s = 20$ (e) $N_s = 10^6, s = 10$ (f) $N_s = 10^6, s = 20$

**Fig. 9.** Effect of Optimization on Various Datasets



(a) skew papameter $s = 10$ (b) skew parameter $s = 20$

**Fig. 10.** Performance Comparison between LP and IP Optimization

## 8 Conclusion

In this paper we introduce the novel problem of BRNN query with strong privacy guarantee, where an adversary cannot distinguish a query point from any other point in the space. This is the first work that applies PIR to BRNN query. Further, we show that it is NP-hard to minimize the number of PIR accesses given any partition scheme over the whole space, and therefore propose a linear programming approximation to the optimal packing problem in our proposed *MonoDB*. Finally, we evaluate our methods on real world dataset and synthetic dataset. Extensive experiments demonstrate the practicality of our method.

## References

1. Tian, F., Gui, X., Yang, P., Zhang, X., Yang, J.: Security analysis for hilbert curve based spatial data privacy-preserving method. In: 10th IEEE International Conference on High Performance Computing and Communications and 2013 IEEE International Conference on Embedded and Ubiquitous Computing, pp. 929–934, Zhangjiajie, China (2013)
2. Wicker, S.B.: The loss of location privacy in the cellular age. Commun. ACM **55**(8), 60–68 (2012)
3. Du, Y.: Privacy-aware RNN query processing on location-based services. In: 8th International Conference on Mobile Data Management, pp. 253–257, Mannheim, Germany (2007)
4. Williams, P., Sion, R.: Usable PIR. In: NDSS, San Diego, California, USA (2008)
5. Kushilevitz, E., Ostrovsky, R.: Replication is NOT needed: SINGLE database, computationally-private information retrieval. In: FOCS, pp. 364–373, Florida, USA (1997)
6. Papadopoulos, S., Bakiras, S., Papadias, D.: Nearest neighbor search with strong location privacy. In: VLDB, pp. 619–629 (2010)
7. Ghinita, G., Kalnis, P., Khoshgozaran, A., Shahabi, C., Tan, K.L.: Private queries in location based services: anonymizers are not necessary. In: SIGMOD, pp. 121–132, Vancouver, BC, Canada (2008)
8. Mouratidis, K., Yiu, M.L.: Shortest path computation with no information leakage. In: VLDB, pp. 692–703, Istanbul, Turkey (2012)
9. Stanoi, I., Riedewald, M., Agrawal, D. and Abbadi, A.E.: Discovery of influence sets in frequently updated databases. In: VLDB, pp. 99–108, Roma, Italy (2001)
10. Wong, W.K., Cheung, D.W.L., Kao, B., Mamoulis, N.: Secure kNN computation on encrypted databases. In: SIGMOD, pp. 139–152, Rhode Island, USA (2009)
11. Khoshgozaran, A., Shahabi, C.: Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In: Papadias, D., Zhang, D., Kollios, G. (eds.) SSTD 2007. LNCS, vol. 4605, pp. 239–257. Springer, Heidelberg (2007)
12. Korn, F., Muthukrishnan, S.: Influence sets based on reverse nearest neighbor queries. In: SIGMOD, pp. 201–212, Dallas, Texas, USA (2000)
13. Tran, Q.T., Taniar, D., Safar, M.: Bichromatic Reverse Nearest-Neighbor Search in Mobile Systems. J. IEEE SYSTEMS **4**(2), 230–242 (2010)
14. Kang, J.M., Mokbel, M.F., Shekhar, S., Xia, T. and Zhang, D.: Continuous evaluation of monochromatic and bichromatic reverse nearest neighbors. In: ICDE, pp. 806–815, Istanbul, Turkey (2007)
15. Chen, S., Jensen, C.S., Lin, D.: A benchmark for evaluating moving object indexes. In: VLDB, pp. 1574–1585, New Zealand (2008)
16. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM **45**(6), 965–981 (1998)