Outsourced Similarity Search on Metric Data Assets

Man Lung Yiu, Ira Assent, Christian S. Jensen, Fellow, IEEE, and Panos Kalnis

Abstract—This paper considers a cloud computing setting in which similarity querying of metric data is outsourced to a service provider. The data is to be revealed only to trusted users, not to the service provider or anyone else. Users query the server for the most similar data objects to a query example. Outsourcing offers the data owner scalability and a low-initial investment. The need for privacy may be due to the data being sensitive (e.g., in medicine), valuable (e.g., in astronomy), or otherwise confidential. Given this setting, the paper presents techniques that transform the data prior to supplying it to the service provider for similarity queries on the transformed data. Our techniques provide interesting trade-offs between query cost and accuracy. They are then further extended to offer an intuitive privacy guarantee. Empirical studies with real data demonstrate that the techniques are capable of offering privacy while enabling efficient and accurate processing of similarity queries.

Index Terms—Query processing, Security, integrity, and protection.

1 INTRODUCTION

Advances in digital measurement and engineering technologies enable the capture of massive amounts of data in fields such as astronomy, medicine, and seismology. The effort of data collection and processing, as well as its potential utility for research or business, create value for the data owner. He wishes to store them and allow access by himself, colleagues, and other (trusted) scientists or customers. This can be supported by outsourced servers that offer low storage costs for large databases. For instance, outsourcing based on cloud computing is becoming increasingly attractive, as it promises pay-as-you-go, low storage costs as well as easy data access. However, care needs to be taken to safeguard data that are valuable or sensitive against unauthorized access. In this context, we call any item in a data collection an object, individuals with authorized access query users, and the entity offering the storage service the service provider.

We illustrate the sensitivity issues with several scenarios. First, consider space programs such as the NASA Apollo program on the Earth’s Moon1 or the ESA Mars Express2 that collect scientifically valuable and rare data. The NASA

1. http://ti.arc.nasa.gov/project/planetary/moon/.
2. http://sci.esa.int/marsexpress/.

1041-4347/12/$31.00 © 2012 IEEE Published by the IEEE Computer Society
scenario, the data is costly and rare, making them very valuable to their owners.

Many applications in science and business rely on similarity search of metric data other than time series and vector data. Computer-aided gene sequencing uses the similarity between an unknown sequence from one species and a known sequence from a closely related species to predict the former’s function4 (see Fig. 1b). In drug design, pharmacists search for the most similar graph structures to their quest for a suitable molecule, which can be represented as a labeled graph,3 as shown in Fig. 1c.

In general, the above diverse scenarios have the following common characteristics: valuable data in a metric space are searched based on a similarity measure. When this data is outsourced, they must be secured against leaks or attacks.

1.1 Shortcomings of Existing Methods

In the literature, a number of concepts for securing databases have been studied. Private information retrieval techniques [15] hide the user’s query, e.g., the data item searched for, but not the data being queried. To outsource valuable data to an insecure server, such techniques are clearly not appropriate. Digital watermarking [2] establishes the data owner’s identity on the data. Additional information stored in the data helps prove ownership, but it cannot prevent an attacker from illegally copying the data set. Anonymization techniques [25] secure data by releasing only a generalized version. Aggregate statistical analysis is still possible on the generalized data, but the result of a specific query is not guaranteed to be accurate.

Traditional encryption methods are capable of protecting the confidentiality of the data. However, this also prevents users from querying the data on the untrusted server. Obviously, transferring all the encrypted data to the query user for searching takes outsourcing ad absurdum. Moreover, when services are made available to users on a pay-as-you-go basis, the service providers are not interested in such a brute force data transfer.

Note that certain applications may involve large numbers of query users from scientific institutions, hospitals, or branch offices globally. For example, consider a query load of 10,000 queries/s. If a brute force solution transfers 10 MBytes for a single query, the server needs a network bandwidth of 100,000 MBytes/s, which is far beyond the available bandwidth of Fast Ethernet (100 Mbits/s). Also, from the user point of view, it is better to have most of the processing done in the cloud. For example, a user with a smart phone may not have enough resources (bandwidth, CPU, battery power) to download and query a large data set locally. Therefore, the design of communication-efficient solutions is of critical importance to the success of cloud computing applications, allowing these to be operated at low cost.

Typically, cloud computing providers (e.g., Amazon, HP, and Microsoft) attempt to solve the problem by offering contractual agreement that promise not to release outsourced data to third parties. Nevertheless, even if the provider respects the contractual agreement, the data is not guaranteed to be safe. Unintended leaks of data is reported regularly, and hackers may still exploit vulnerabilities to gain access to data. Therefore, we believe that data owners will find it attractive to outsource encrypted rather than plain data.

Closest to our work are the recent outsourcing proposals [28], [31] on searching problems in the spatial domain and multidimensional space, respectively. Unfortunately, their techniques rely on specific properties of those spaces and they cannot be extended to solve our problem, which considers arbitrary metric data spaces (e.g., strings, graphs, time-series).

1.2 Objective

The goal of this research is to develop a transformation method \( t() \) for converting an original object \( p \) in a metric space into another metric space object \( p' = t(p) \). First, the data owner specifies a key value \( CK \) in order to define the instance of \( t() \) to be used. In a preprocessing phase, the data owner computes \( p' \) for each object \( p \) and uploads it to the server (i.e., service provider). At query time, the query user specifies his query object \( q \) and then submits the transformed query object \( q' \) to the server for similarity search. The transformation method must satisfy these requirements:

- Even in the worst case that the attacker knows the inverse of \( t() \), he can only estimate the original object \( p \) from the transformed object \( t(p) \) with bounded precision.
- It enables high-query accuracy.
- It enables efficient query processing in terms of communication cost.
- It supports insertion and deletion of objects.

Our contributions are as follows: We present three transformation techniques that satisfy the above requirements. They represent various trade-offs among data privacy and query cost and accuracy.

- In our first solution, we propose an encrypted index-based technique with perfect privacy, but multiple communication rounds. This technique flexibly reduces round trip latency at the expense of data transfer.
- For our second solution, our private anchor-based indexing guarantees the correct answer within only 2 rounds of communication. Retrieval is accelerated by bounding the range of potential nearest neighbors (NN) in the first phase.
- Our third solution limits communication to a single round, and also returns a constant-sized candidate set by computing a close approximation of the query result.

5. For example, molecule data available at http://dtp.nci.nih.gov.
We extend our solutions in order to meet an intuitive privacy guarantee requirement.

The rest of the paper is organized as follows: Section 2 reviews related work. We define the problem and introduce an intuitive privacy guarantee in Section 3, before presenting our solutions in Section 4. Thorous experimental evaluation is discussed in Section 5, before we conclude in Section 6.

2 RELATED WORK

We first introduce existing work on indexing and nearest neighbor search techniques for metric data. Then, we will cover work on privacy and security of outsourced data.

2.1 Indexing and NN Search in Metric Space

We review metric indexing because our proposed methods provide metric indexing on the server for efficient processing.

The R*-tree [7] and the X-tree [8] are well-known disk-based indexes for multidimensional objects, where each object is modeled as a vector of coordinate values. Complex data objects (e.g., DNA sequence, time series) cannot be effectively represented by coordinate values. Instead, we model them in metric space, where a (black-box) distance function \( \text{dist}(p_i, p_j) \) is used to compute the dissimilarity between objects \( p_i \) and \( p_j \). The distance function \( \text{dist}(\cdot) \) is said to be a metric if it satisfies symmetry, nonnegativity, and the triangle inequality. Interested readers are referred to two excellent surveys [10], [20] on metric space indexing. In this section, we only describe three representative indexing methods for a set \( \mathcal{P} \) of metric space objects. They are the vantage-point tree (VP-tree) [29], the multi-vantage-point (MVP-tree) [9], and the M-tree [11].

The VP-tree is a binary tree built on \( \mathcal{P} \) by utilizing the mutual distances among the objects in \( \mathcal{P} \) [29]. First, we choose an arbitrary object \( a \in \mathcal{P} \) as the root object, and then we determine the median distance \( r \) among the distances \( \text{dist}(a, p) \) from \( a \) to the objects \( p \in \mathcal{P} \). Each object \( p \in \mathcal{P} \) satisfying \( \text{dist}(a, p) \leq r \) is inserted into the left subtree of \( a \), whereas the others are inserted into the right subtree of \( a \). The tree is built in a top-down manner by applying the above construction procedure recursively to the subtrees of \( a \). The MVP-tree [9] is an extension of the VP-tree, so that each index node stores two anchor objects and has \( m^2 \) subtrees (\( m \) being a parameter). The VP-tree (and MVP-tree) supports insertion/deletion of objects at the risk of an unbalanced tree.

The most popular metric space index is the M-tree [11] (and its variant [22]) due to its efficient support of object insertion/deletion. Each index entry \( e \) stores a minimum bounding sphere (MBS) consisting of

1. an anchor object \( e.a \) as the sphere center,
2. a covering radius \( e.r = \max\{\text{dist}(e.a, p) \mid p \in \text{sub}(e)\} \) as the maximum distance from \( e.a \) to any object in the subtree of \( e \).

In addition, the entry stores a pointer to its child node, and a precomputed distance from \( e.a \) to the parent entry of \( e \).

In contrast, a leaf entry only stores the actual object \( p \), and its precomputed distance to its parent entry.

Given a query object \( q \) and a set \( \mathcal{P} \) of objects, the nearest neighbor query retrieves the object \( p \in \mathcal{P} \) such that \( \text{dist}(q, p) \) is minimized. The best-first paradigm [20], [26] is the state-of-the-art method for performing NN search on a hierarchical metric space index (e.g., the M-tree). Given a query object \( q \) and an index entry \( e \), the function \( \text{mindist}(q, e) \) is used to compute the (conservative) minimum distance between \( q \) and any object indexed by the subtree of \( e \). The best-first search employs a min-heap \( H \) for organizing its encountered entries in ascending order of \( \text{mindist}(q, e) \). Initially, the entries in the root node of the tree are inserted into \( H \). When an index entry \( e \) is deheap from \( H \), we access its child node and insert all entries of the node into \( H \). The first object \( p \) that is deheap from \( H \) is guaranteed to be the NN of \( q \).

Hashing techniques [6], [16] have also been proposed to answer the NN query efficiently. These techniques do not guarantee exact NN retrieval, but they return objects close enough to the NN in practice. The locality-sensitive hashing technique (LSH) [16] is specifically designed for the \( L_2 \) norm in the multidimensional space \( \mathbb{R}^d \); it is inapplicable to arbitrary metric spaces (e.g., edit distance over the domain of strings).

The distance-based hashing technique (DBH) [6] is an extension of LSH for metric spaces. It takes as input two parameters: 1) the number \( A \) of bits, and 2) the number \( C \) of hash tables. Let \( HT_j \) be the \( j \)th hash table, for indexing objects \( p \in \mathcal{P} \) based on their \( A \)-length bitmaps \( BM_j(p) \). To compute the \( j \)th bit of the bitmap \( BM_j(p) \), we pick two anchor objects \( a_j, b_j \in \mathcal{P} \), and define the projection function [14] as

\[
\mathcal{PF}_{a_j, b_j}(p) = \frac{\text{dist}^2(p, a_j) + \text{dist}^2(a_j, b_j) - \text{dist}^2(p, b_j)}{2 \cdot \text{dist}^2(a_j, b_j)}.
\]

Then, we determine the value \( r_j \) as the median value of \( \mathcal{PF}_{a_j, b_j}(p) \). The \( j \)th bit of \( BM_j(p) \) is set to 0 if \( \mathcal{PF}_{a_j, b_j}(p) \leq r_j \); otherwise, the bit is set to 1.

During the construction phase, we insert each object \( p \in \mathcal{P} \) into the hash table \( HT_j \) according to the bitmap \( BM_j(p) \). This step is repeated for all \( C \) hash tables. At query time, the user requests the hash table \( HT_j \) to return all objects having the same bitmap as the bitmap \( BM_j(q) \) of the query object \( q \). Again, this step is repeated for all hash tables and eventually the closest of them to \( q \) is reported as the result.

Nevertheless, DBH has two limitations. First, it is possible that no hash table \( HT_j \) contains any object with the same bitmap as the query bitmap \( BM_j(q) \), leading to an empty result. Second, once the DBH structure is built (i.e., values of \( A \) and \( C \) are fixed), its query accuracy cannot be dynamically optimized by the user. We will address the above problems by developing a flexible hashing technique in Section 4.3 that 1) prevents empty results, and 2) allows the user to boost the query accuracy online by trading-off communication cost.

2.2 Privacy and Security

The idea of outsourcing database services to a service provider was introduced by Hacigümüş et al. [18]. Since then, various techniques have been developed to maintain the confidentiality of outsourced data. Given a relational table, Hacigümüş et al. [17] map the tuples of the table into
buckets and then store the encrypted tuples of those buckets at the server. At query time, the user compares the query object against the description of those buckets, and then determines the necessary buckets that need to be retrieved from the server. In another proposal [12], the data owner applies the encryption function on each node separately and then stores all encrypted tuples at the server. The method of Agrawal et al. [3] employs an order-preserving function on 1D data values such that the distribution of output values is different from that of input values.

The two works most related to ours are proposed by Yiu et al. [31] and Wong et al. [28]. Yiu et al. [31] present several transformation-based techniques for outsourcing spatial data to the (untrusted) server, such that the server is able to perform spatial range search correctly for trusted users on those transformed points, without knowing their actual coordinates. They propose spatial transformations in 2D space based on scaling, shifting, and noise injection. Also, they develop a solution using an encrypted R-tree. Those solutions operate on explicit 2D coordinates, rendering them inapplicable in our setting, where the distance function is a generic distance metric.

Wong et al. [28] propose to outsource multidimensional points to the (untrusted) server, by using a secure scalar product encryption technique. Methods are then provided for kNN search at the server, without the server learning the distances among the points. However, the secure scalar product relies on specific properties of the euclidean distance in the multidimensional space. It is not applicable to other L_p norms, e.g., the L_1 norm (the Manhattan distance). Obviously, it also cannot be applied to our problem setting which considers arbitrary metric space objects (e.g., strings, graphs, time-series). Another drawback of this proposal is that no indexing scheme can be built on the encrypted tuples, forcing the server to perform a linear scan over the data set. This affects severely the scalability of the system.

In the field of privacy-preserving data mining, perturbation techniques [4], [23] have been developed for introducing noise into the data, before sending them to the service provider. However, such an approach does not guarantee the exact retrieval of results.

The k-anonymity model [27] has been applied extensively for the privacy-preserving publication of data sets. The idea is to generalize the tuples in a table such that each generalized representation is shared by at least k tuples. This way, each object cannot be distinguished from at least k - 1 other objects. It is often used to generalize the medical records of patients so that the adversary cannot link a specific patient to a medical record. Except for some person-related data like DNA data, most of the metric data that we consider (e.g., astronomy data, time series) is collected from nature rather than from persons. Therefore, k-anonymity is of relatively low relevance to the intended applications of the paper’s proposals. Nevertheless, we still view this k-anonymity approach as a competitor solution in Section 3.3.

3 Problem Setting

We start by introducing our scenario and our problem setting in Section 3.1. Then, we propose an intuitive privacy guarantee for metric data in Section 3.2. Next, we describe straightforward solutions to our problem and discuss their drawbacks in Section 3.3.

3.1 Problem Definition

We first discuss our scenario and then define our problem.

3.1.1 Scenario

Fig. 2 depicts our scenario for outsourcing data. It consists of three entities: a data owner, a trusted query user, and an untrusted server. On the one hand, the data owner wishes to upload his data to the server so that users are able to execute queries on those data. On the other hand, the data owner trusts only the users, and nobody else (including the server).

The data owner has a set P of (original) objects (e.g., actual time series, graphs, strings), and a key to be used for transformation. First, the data owner applies a transformation function (with a key) to convert P into a set P’ of transformed objects, and uploads the set P’ to the server (see step A1 in the figure). The server builds an index structure on the set P’ in order to facilitate efficient search. In addition, the data owner applies a standard encryption method (e.g., AES) on the set of original objects; the resulting encrypted objects (with their IDs) are uploaded to the server and stored in a relational table (or in the file system).

Next, the data owner informs every user of the transformation key (see step A2). In the future, the data owner is allowed to perform incremental insertion/deletion of objects (see step A3).

At query time, a trusted user applies the transformation function (with a key) to the query and then sends the transformed query to the server (see step B1). Then, the server processes the query (see step B2), and reports the results back to the user (see step B3). Eventually, the user decodes the retrieved results back into the actual results. Observe that these results contain only the IDs of the actual objects. The user may optionally request the server to return the actual objects that correspond to the above result set.

Table 1 summarizes the notations used throughout the paper.

3.1.2 Problem Definition

We will use the term object for the metric data of interest to the data owner. A transformed object then refers to an object obtained from a transformation.
Let \(\text{dist}(p_i, p_j)\) denote the distance between two objects \(p_i\) and \(p_j\). We focus on nearest neighbor queries, for simplicity. The extension to the case of \(k\) nearest neighbors is straightforward. We first give the definition of the nearest neighbor query as follows:

**Definition 1 (Nearest Neighbor Query).** Given a query object \(q\) and a set \(P\) of objects, the nearest neighbor query retrieves the object \(p_{nn} \in P\) such that \(\text{dist}(q, p_{nn}) \leq \text{dist}(q, p)\) for all \(p \in P\).

Recall from Fig. 2 that both steps A1 and B1 require the data owner and the user to apply a transformation function. Our research objective is to design a transformation method that meets the following requirements:

- Even in the worst case where the attacker knows the inverse of the transformation function, the attacker can only estimate the original object \(p\) from the transformed object \(p'\) with bounded precision.
- It enables high-query accuracy.
- It enables efficient query processing in terms of communication cost.
- It supports insertion and deletion of objects.

### 3.2 Privacy Guarantee

In this section, we employ an intuitive obfuscation-based privacy guarantee that can be adapted for metric data.

In the two-dimensional space, obfuscation [5] has been used to represent an object’s location by a superset region called the **obfuscated region**. An adversary without a priori knowledge is unable to distinguish the object’s actual location from other locations in the obfuscated region. The privacy value is typically expressed as the area of the obfuscated region in the two-dimensional space. However, for generic metric spaces, there is only the concept of distance but not area. Privacy thus means avoiding small distance between an object and its obfuscated representation. We propose to obfuscate an object \(p\) by using a ring \((a, \text{dist}(a, p))\) whose center is a reference object \(a\) and radius is \(\text{dist}(a, p)\). This way, the object cannot be distinguished from any other possible object (not necessarily from the data set) that have the same obfuscated ring representation. We formally define this privacy guarantee as:

**Definition 2 (δ-Gap Guarantee).** Let \(p\) be an object of the data set \(P\). The ring \((a, \text{dist}(a, p))\) satisfies the δ-gap guarantee if \(\text{dist}(a, p) \geq \delta\).

The data owner is able to tune the value of \(\delta\) such that it describes exactly the required degree of obfuscation. In the example of Fig. 3a, the object \(p\) is represented by the ring \((a, r)\) where \(a\) is a reference object and \(r = \text{dist}(a, p)\). Since \(r \geq \delta\), the ring satisfies the δ-gap guarantee. Observe that any possible object (e.g., object \(p'\)) having distance \(r\) from object \(a\) also has the same representation as \((a, r)\).

The reference object \(a\) could be either an object picked from the data set \(P\) or a randomly generated object that satisfies \(\text{dist}(a, p) = \delta\). Regarding the choice of \(\delta\), we suggest to set \(\delta\) to the average \(c\)th nearest neighbor distance, where \(c\) is a small constant. This value causes the objects to be displaced in the vicinity of their neighbors, without significantly affecting most of the distances between pairs of objects.

For certain data related to individuals (e.g., DNA data), a more appropriate privacy guarantee may be \(k\)-anonymity [27]. Its adaptation to this paper’s problem setting and the extensions to the paper’s proposals necessary to support it are available in a technical report [30].

### 3.3 Straightforward Solutions

We describe two straightforward solutions to our problem and discuss why they are inefficient.

#### 3.3.1 Brute-Force Secure Solution (BRUTE)

This brute-force solution is the one we mentioned in the Introduction. In the offline construction phase, the data owner applies conventional encryption (e.g., AES) on each object and then uploads those encrypted objects to the server. At query time, the user needs to download all encrypted objects from the server, decrypt them and then compute the actual result. As mentioned, it is perfectly secure, but its query and communication cost are both prohibitively high, and pay-as-you-go is not supported.

#### 3.3.2 Anonymization-Based Solution (ANONY)

This anonymization-based solution achieves data privacy by means of \(k\)-anonymity—the objects are generalized in such a way that each generalized object cannot be distinguished from \(k - 1\) other generalized objects. In the context of similarity search, it is able to confuse the ranking of transformed objects because \(k - 1\) of them have the same rank as the transformed object of the actual nearest neighbor. Thus, we still consider this solution as a competitor, even though \(k\)-anonymity is not a suitable privacy guarantee for our applications (see Section 2.2).

For multidimensional data, the anonymization technique of LeFevre et al. [25] can be applied.

In the offline construction phase, the data owner applies a \(K\)-D tree partitioning technique [25] on the data set to obtain disjoint buckets such that each bucket contains at
least \( k \) objects. For each bucket \( e \), the data owner uploads to the server: 1) \( e.\text{MBR} \), the minimum bounding rectangle (MBR) of all objects inside the bucket and 2) encrypted strings of the tuples assigned to that bucket.

Let \( \text{mindist}(q, e.\text{MBR}) \) and \( \maxdist(q, e.\text{MBR}) \) represent the minimum and maximum distance from the query object \( q \) to \( e.\text{MBR} \) (the MBR of the bucket \( e \)).

At query time, the query user first obtains the MBRs of all buckets from the server, then computes the maximum distance from \( q \) to each bucket, and determines the smallest maximum distance (say, \( \gamma = \min_{e} \maxdist(q, e) \)). Next, the query user requests from the server all encrypted tuples from buckets \( e \) that satisfy \( \text{mindist}(q, e) \leq \gamma \). Eventually, the query user decrypts those tuples in order to obtain the actual result.

For arbitrary metric space data, the clustering-based anonymization technique of Aggarwal et al. [1] can be applied. It represents each bucket as a minimum bounding sphere consisting of an anchor object and a covering radius, similar to M-tree index entries as described in Section 2.1.

The above anonymization-based solution has two limitations. First, the MBRs/MBSs of the buckets contain substantial empty space, due to the curse of dimensionality. Thus, the derived upper NN distance bound \( \gamma \) can be loose, triggering a large number of buckets to be retrieved. Second, the solution still allows the server to know the MBRs/MBSs of the buckets, which are located in the same space as the original objects. In contrast, our proposed methods in Sections 4.2 and 4.3 do not permit the server to know any information in the original space; the anchors are converted to IDs and distance information is transformed to numbers or bitmaps.

## 4 Proposed Solutions

In this section, we present the transformation methods, and the corresponding query processing techniques. We propose three transformation methods (EHI, MPT, FDH) in Sections 4.1, 4.2, and 4.3, respectively. They capture various trade-offs among data privacy and query cost and accuracy. We present extensions of MPT and FDH in Section 4.4, in order to comply with the \( \delta \)-gap guarantee. We emphasize that, all the above methods view the distance function \( \text{dist}() \) as a black-box; they only require the distance function to be a metric (i.e., with the triangle inequality).

### 4.1 Encrypted Hierarchical Index Search (EHI)

This section presents a client algorithm, called encrypted hierarchical index, for performing NN search on an encrypted hierarchical index stored at the server. This method offers perfect data privacy for the data owner, but it incurs multiple communication round trips during query processing.

In the literature, algorithms have been developed for processing range queries on encrypted \( B^+ \)-tree [12] and encrypted R-tree [31]; however, no solutions were proposed for the NN query on those encrypted indexes.

#### 4.1.1 Transformation Key

The transformation key of EHI is simply an encryption key \( \text{CK} \) for standard encryption algorithm (e.g., AES).

#### 4.1.2 Data Transformation

An index entry \( e \) consists of its anchor object \( e.a \) and its covering radius \( e.r \). Given a query object \( q \) and an index entry \( e \), their minimum distance and maximum distance are defined as \( \text{mindist}(q, e) = \min \{0, \text{dist}(q, e.a) - e.r\} \) and \( \maxdist(q, e) = \text{dist}(q, e.a) + e.r \), respectively [11]. In Fig. 4, the anchor object of each entry is shown as a gray dot. For the index entry \( e_0 \), we can compute \( \text{mindist}(q, e_0) = 6 - 1 = 5 \) and \( \maxdist(q, e_0) = 6 + 1 = 7 \).

Our EHI method supports any disk-based hierarchical index (e.g., the M-tree [11]), provided that they permit the computation of \( \text{mindist}(q, e) \) and \( \maxdist(q, e) \). Please refer to Section 2 for the discussion on the M-tree. To construct the EHI structure, we first build a disk-based tree index \( TR_p \) on the data set \( P \) (e.g., the example M-tree in Fig. 4). Then, for each tree node, we encrypt its content, and send the encrypted node with its disk block ID to the server. At the end, we send the disk block ID of the root node to the server.

#### 4.1.3 Query Processing

Since the tree index stored at the server is encrypted, the server cannot process the NN query by itself. An algorithm for communication between the client and the server needs to be developed in order to answer the NN query correctly.

The total response time of the algorithm consists of the round trip latency and the data transfer time. These two measures are analogous to the seek time and transfer time in hard disks. Traditional best-first NN search algorithms [20], [26] guarantee that the data transfer time is minimized. However, in the above context, they need to send a message to the server each time a node is requested. This would incur very high round trip latency.

This motivates us to enhance the best-first NN search algorithm in the context of a client-server architecture. Algorithm 1 is the pseudocode for the user to search the encrypted index stored at the server side. It has two new features: 1) a parameter \( \lambda \) is used to reduce the number of communication messages and round trip latency (see Lines 7-8), and 2) nonleaf entries are exploited to derive an upper bound of NN distance for pruning unqualified entries (see Lines 3 and 13).

**Algorithm 1.** EHI Searching Algorithm for Client.

**Algorithm** EHI-Search (Query object \( q \), Encryption Key \( \text{CK} \), Integer \( \lambda \ ))

1: request the server for the (encrypted) root node \( L_{\text{root}} \),
### 4.2 Metric Preserving Transformation (MPT)

In this section, we develop a method, called metric preserving transformation, for evaluating the NN query. Unlike the EHI method, MPT incurs only 2 rounds of communication during the query phase.

The basic idea behind MPT is to pick a small subset of the data set $P$ as the set of anchor objects [11] and then assign each object of $P$ to its nearest anchor. For each object $p$, we compute its distance $\text{dist}(a_i,p)$ from its anchor $a_i$ and then apply an order-preserving encryption function $\text{OPE}$ [3] on the distance value. These order-preserving encrypted distances will be stored in the server and utilized for processing NN queries.\(^6\)

A function $\text{OPE} : \mathbb{R} \to \mathbb{R}$ is said to be order preserving if it guarantees that $\text{OPE}(x) > \text{OPE}(x')$ for any values $x, x'$ that satisfy $x > x'$.

#### 4.2.1 Transformation Key

The transformation key consists of an encryption key $\text{CK}$, an integer $A$, and $A$ pairs of the form $(a_i, r_i)$, where $a_i$ is an (anchor) object and $r_i$ is a distance value. We will soon elaborate on how to generate the transformation key.

#### 4.2.2 Data Transformation

Algorithm 2 is the pseudocode for constructing MPT from the input data set $P$. Integer $A$ denotes the number of anchor objects. We apply a heuristic from the M-tree [11] to select a set of $A$ anchor objects from $P$. Such a heuristic aims at optimizing the “indexing quality” of the anchors, but we do not cover the details here. In Line 2, we compute the value $B$, i.e., the maximum number of objects that can be assigned to the same anchor. Next, we apply a heuristic [11] to assign each object to its anchor object. For example, each object is assigned to its nearest anchor object (having fewer than $B$ assigned objects).

**Algorithm 2. MPT Building Algorithm for Data Owner.**

**Algorithm** MPT-Build (Data Set $P$, Encryption Key $\text{CK}$, Integer $A$)

1: use a heuristic of Ref. [11] to select a set of $A$ anchor objects from $P$;
2: Integer $B := \lceil |P| / A \rceil$;
3: use a heuristic of Ref. [11] to assign each data object of $P$ to an anchor object, subject to the capacity constraint $B$;
4: for $i := 1$ to $A$ do
5: let $a_i$ be the $i$-th anchor object;
6: let $a_i.S$ be the set of objects assigned to the anchor $a_i$;
7: $r_i := \max_{p \in a_i.S} \text{dist}(a_i,p)$;  \(\triangleright\) compute covering radius
8: for each object $p \in a_i.S$ do
9: send the tuple $(p.id,\text{OPE}(|\text{dist}(a_i,p)|),\text{ECR}(p,\text{CK}))$ to the server;

Next, we examine each anchor object. Let $a_i$ denote the $i$-th anchor object and the set $a_i.S$ represent its assigned set of objects. We compute the anchor’s covering radius $r_i$, which denotes the maximum distance from $a_i$ to any object in its set $a_i.S$ [11]. The anchor distance plays an important role in

---

\(^{6}\) In fact, the transformation of MPT is different from iDistance [21] because iDistance is specifically designed for indexing points in a multidimensional space rather than in a generic metric space. Our MPT solution is applicable to arbitrary metric data (e.g., graphs, strings), but iDistance is not.

---

### Table 2

Requested Nodes in EHI for Communication Rounds

<table>
<thead>
<tr>
<th>Round</th>
<th>Requested Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>root node</td>
</tr>
<tr>
<td>2</td>
<td>$e_2$</td>
</tr>
<tr>
<td>3</td>
<td>$e_2, e_1$</td>
</tr>
<tr>
<td>4</td>
<td>$e_2, e_1$</td>
</tr>
<tr>
<td>5</td>
<td>$e_2, e_1$</td>
</tr>
</tbody>
</table>

2: $H := \text{new min-heap}; p_{\text{min}} := \text{NULL};$
3: $\gamma := \min_{e \in L_{\text{root}}} \max \text{dist}(q,e)$; \(\triangleright\) derive NN distance bound
4: for each entry $e \in L_{\text{root}}$ such that $\min \text{dist}(q,e) \leq \gamma$ do
5: insert the entry $(e, \min \text{dist}(q,e))$ into $H$;
6: while $H$ is not empty and its top entry’s key $\leq \gamma$ do
7: pop next $\lambda$ entries from $H$ and insert them into a set $S$;
8: request the server for each (encrypted) child node of $S$;
9: for each retrieved node $L_{\text{cur}}$ do
10: if $L_{\text{cur}}$ is a leaf node then \(\triangleright\) check for closer objects
11: update $\gamma$ and $p_{\text{min}}$ by using objects in $L_{\text{cur}}$;
12: else \(\triangleright\) expand the entries of $L_{\text{cur}}$
13: $\gamma := \min \{\gamma, \min_{e \in L_{\text{cur}}} \max \text{dist}(q,e)\}$;
14: for each $e \in L_{\text{cur}}$ such that $\min \text{dist}(q,e) \leq \gamma$ do
15: insert the entry $(e, \min \text{dist}(q,e))$ into $H$;
16: return $p_{\text{min}}$ as the result;

Table 2 illustrates our algorithm on the example in Fig. 4, for different values of $\lambda$. Interestingly, our algorithm represents a generalization of different search paradigms on a disk-based tree index. When $\lambda$ equals 1, our algorithm degenerates to the traditional best-first search. It has the optimal data transfer cost (5 nodes), but incurs a large number of round trips (5 rounds). When $\lambda$ is set to $\infty$, our algorithm behaves as breadth-first search. It has the optimal number of round trips (i.e., the tree height, 3), but it may lead to a high data transfer cost (7 nodes). Observe that after the first round, the distance $\max \text{dist}(q,e_2)$ is used as the upper bound of the NN distance so the entry $e_3$ is pruned before the second round.

When $\lambda$ is set to 2, the algorithm accesses only 5 nodes in a total of 3 rounds. Thus, it has the lowest response time when compared to other values of $\lambda$. In general, when the value of $\lambda$ is large, the number of round trips is small, but the data transfer cost may be high.

As a guideline, we recommend to set the value of $\lambda = \lceil \frac{\tau_{\text{node}}}{\tau_{\text{trip}}} \rceil$, where $\tau_{\text{node}}$ is the transfer time of a node, and $\tau_{\text{trip}}$ is the round trip latency. This scheme ensures that the data transfer time of $\lambda$ nodes equals the round trip latency.

#### 4.1.4 Incremental Data Update

The data owner is able to insert and delete objects from the encrypted tree, similar to the way of using the underlying index (e.g., M-tree).
query processing, as we will discuss shortly. For each object $p$ in the set $a_i S$, we compute its distance $\text{dist}(a_i, p)$ from its anchor, and then apply an order-preserving encryption function $\text{OPE}$ [3] on $\text{dist}(a_i, p)$. A tuple consisting of the object ID $p.id$, the order-preserving encrypted distance $\text{OPE}(\text{dist}(a_i, p))$, and the encrypted object $\text{ECR}(p, CK)$ will be sent to the server. The benefit of using $\text{OPE}$ is that it hides the original distance values and yet allows comparisons to be correctly evaluated at the server side.

Fig. 5a shows an example with eight objects. Suppose $\text{dist}(a_1, p_1), \text{dist}(a_1, p_2)$ are assigned to anchor $a_1$, whereas the others are assigned to anchor $a_2$. The transformed data set also stores the order-preserving encrypted anchor distance and encrypted object for each tuple. For instance, $p_2$ has distance $\text{dist}(a_1, p_2) = 0.32$ from the anchor $a_1$. To prevent the server from knowing such an actual distance value, the order-preserving function $\text{OPE}$ is applied on the distance and the distance value 0.8 is stored instead.

As a remark, the transformed data set (at the server side) can be indexed by an $R^*$-tree using a composite key on the anchor ID and the anchor distance.

### 4.2.3 Query Processing

The client has the transformation key, which contains an encryption key $CK$, the value of $A$, the set of anchor objects $\{a_i\}$, and the corresponding distance values $\{r_i\}$.

We first establish the foundation of NN search on MPT. Let $q$ be the query object and $\gamma$ be an upper bound of the NN distance (derived by the client). The client is able to compute the values $\text{dist}(q, a_i) - \gamma$ and $\text{dist}(q, a_i) + \gamma$ and their order preserving encrypted values using $\text{OPE}(\cdot)$. With these encrypted distances, the following lemma states when an object $p$ cannot become the NN of $q$. In order to guarantee exact NN retrieval, the client needs to issue a distance range request to the server so that any object $p$ satisfying $\text{OPE}(\text{dist}(a_i, p)) \in [\text{OPE}(\text{dist}(q, a_i) - \gamma), \text{OPE}(\text{dist}(q, a_i) + \gamma)]$ is retrieved.

**Algorithm 3. MPT Searching Algorithm for Client.**

**Algorithm MPT-Search (Query object $q$, Encryption Key $CK$, Integer $\theta$, Integer $A$, Pairs $\{(a_i, r_i)\}$)**

1. $\gamma := \min_{i \in [1, A]}(\text{dist}(q, a_i))$; \(\triangleright\) initial bound of NN distance
2. let $a_{\text{near}}$ be the anchor leading to the $\gamma$ value;
3. request the server for $\theta$ random tuples whose anchor ID equals to that of $a_{\text{near}}$;
4. let $S_{\text{samp}}$ be the set of decrypted objects from the received tuples;
5. for each $p \in S_{\text{samp}}$ do
6. $\gamma := \min\{\gamma, \text{dist}(q, p)\}$; \(\triangleright\) refined bound of NN distance
7. for $i := 1$ to $A$ do
8. if $\text{mindist}(q, (a_i, r_i)) \leq \gamma$ then
9. request the server for all tuples (with anchor ID as $a_i$) whose $\text{OPE}(\text{dist}(a_i, p))$ falls into the range $[\text{OPE}(\text{dist}(q, a_i) - \gamma), \text{OPE}(\text{dist}(q, a_i) + \gamma)]$;
10. let $S_{\text{and}}$ be the set of decrypted objects from the received tuples (of the above request);
11. return the object $p \in S_{\text{and}}$ with the minimum $\text{dist}(q, p)$ value as the final result;

The objective of the distance bounding phase (Lines 1-6) is to derive a tight upper bound $\gamma$ of the NN distance, in order to reduce the cost of the candidate retrieval phase substantially. In Line 1, the anchor objects are used to derive an upper bound of the NN distance $\gamma$. In Lines 2-5, the closest anchor $a_{\text{near}}$ to $q$ is selected and the client requests the server to return $\theta$ random objects associated with that anchor. These objects are then used to further tighten the NN distance bound $\gamma$.

The goal of the candidate retrieval phase (Lines 7-11) is to retrieve the set of potential candidates that can become the NN of $q$. The minimum distance between $q$ and any object indexed by anchor $a_i$ (with covering radius $r_i$) is given by Ciaccia et al. [11]

$$\text{mindist}(q, (a_i, r_i)) = \max\{\text{dist}(q, a_i) - r_i, 0\}.$$
Obviously, we only need to consider the anchors satisfying \( \text{mindist}(q, (a_i, r_i)) \leq \gamma \). In Line 9, we apply Lemma 1 and issue the corresponding distance range query to retrieve the potential candidates indexed by anchor \( a_i \). Eventually, we obtain the candidate set \( S_{\text{can}} \) and return the object \( p \in S_{\text{can}} \) with the minimum \( \text{dist}(q, p) \) value as the result.

Fig. 5b illustrates NN query processing on MPT. The query object \( q \) is shown as a white dot. Suppose that \( \text{dist}(q, a_1) = 0.13 \) and \( \text{dist}(q, a_2) = 0.34 \). First, the client uses the anchor objects to derive the upper distance bound \( \gamma = \text{dist}(q, a_1) = 0.13 \). For simplicity, we skip the steps of using \( \theta \) additional objects to tighten \( \gamma \) (see Lines 2-6 of the algorithm). We then proceed to the candidate retrieval phase. We derive the distance range query \([0.13 - 0.13, 0.13 + 0.13] = [0, 0.26]\) for anchor \( a_1 \), and the distance range query \([0.34 - 0.13, 0.34 + 0.13] = [0.21, 0.47]\) for anchor \( a_2 \). We then apply the \( \text{OPC} \) function on those distance ranges and send the transformed queries to the server. The search space is depicted by the gray region in Fig. 5b. Thus, the server searches the transformed data set of Fig. 5c and returns the encrypted tuples corresponding to objects \( p_1, p_2, p_3, p_4 \). Eventually, the client decrypts those objects and obtains object \( p_4 \) as its NN.

**4.2.4 Incremental Data Update**

The data owner inserts or deletes an object \( p \) for the corresponding bucket having the anchor object of \( p \).

If the data distribution in the data set changes gradually over time, the objects in a partition may be far away from their anchor at some point. To recover the indexing quality of MPT, the data owner can perform the following periodic (say, weekly) maintenance procedure: 1) split the largest partition into two equal-sized partitions (with new anchors), and 2) merge the smallest partition with its nearest partition. The maintenance overhead is small because it involves only three partitions.

**4.3 Flexible Distance-Based Hashing (FDH)**

In this section, we propose a hashing-based technique, called flexible distance-based hashing, for processing the NN query. The main advantage of this technique is that the server always returns a constant-sized candidate set (in one communication round). The client then refines the candidate set to obtain the final result. Even though FDH is not guaranteed to return the exact result, the final result is very close to the actual NN in practice.

During query processing, FDH allows the client to specify an integer parameter \( \theta \) for increasing the accuracy of a query result, without rebuilding the transformed data stored at the server. This is a significant enhancement over earlier work [6], in which query accuracy cannot be adjusted once the indexing structure is built.

In addition, our FDH method employs a novel technique for conceptually linking similar hash buckets, in order to maximize the utility of the transformed data for answering queries.

In the literature, the Hilbert curve transformation has been employed for approximate NN search in 2D space [24]. However, its performance degrades rapidly for high-dimensional space and it is inapplicable to the metric space.

In contrast, our FDH method can be applied for any black-box distance function \( \text{dist}() \).

**4.3.1 Transformation Key**

The transformation key consists of an encryption Key \( CK \), an integer \( A \), and \( A \) pairs of the form \( (a_i, r_i) \), where \( a_i \) is an object and \( r_i \) is a distance value. The generation of the transformation key will be discussed shortly.

**4.3.2 Data Transformation**

The data owner specifies a parameter \( A \) for the data transformation step. A typical value of \( A \) is in the range of tens to hundreds. This parameter provides a trade-off between the cost of transformation and the query accuracy, e.g., a higher value of \( A \) leads to better query accuracy, but also high-transformation cost.

Let \( P \) be the original data set of objects. In the data transformation phase, we choose \( A \) random objects from the set \( P \) as anchor objects. Let these anchor objects be \( a_1, a_2, \ldots, a_A \). For each anchor object \( a_i \), we need to pick a distance value \( r_i \) (which will be determined shortly).

Given an object \( p \in P \), we convert it into an \( A \)-length bitmap where the \( i \)-th bit of the bitmap is defined as

\[
B_M(p)[i] = \begin{cases} 
0 & \text{if } \text{dist}(a_i, p) \leq r_i, \\
1 & \text{otherwise.}
\end{cases}
\]

Our transformation is cheap to compute because the \( \text{dist}() \) function is called exactly once for computing a bit value. In contrast, the transformation function of Athitsos et al. [6] is more expensive to evaluate as it calls the \( \text{dist}() \) function three times (for computing a bit value).

Fig. 6a shows a set of objects \( P = \{p_1, p_2, p_3, p_4, p_5, p_6\} \). Suppose that \( p_1 \) is chosen as the anchor object \( a_1 \). We illustrate how to derive the distance value \( r_1 \) for the anchor \( a_1 \). Observe that each object within the circle (i.e., \( p_1, p_4, p_6 \)) has a bit value 0, whereas any other object (i.e., \( p_2, p_3, p_5 \)) has a bit value 1. If the distance value \( r_1 \) is too large, then most of the objects will be assigned the bit value 0, implying that we cannot distinguish pairs of objects (e.g., the pair \( (p_1, p_2) \)) that are far apart. A similar problem arises when \( r_2 \) is set to a small value. To balance the number of objects on both sides, we choose the distance value \( r_i \) as the median value among the distances \( \text{dist}(a_i, p) \) (of objects from the anchor \( a_i \)).

Fig. 6b shows an example with \( A = 3 \) anchors \( a_1, a_2, a_3 \) (which are \( p_1, p_2, p_3 \), respectively). The domain space is conceptually partitioned into eight regions, each of which can be represented by a bitmap as defined in (1). For example, the region “010” contains objects that are close to \( a_1 \) and \( a_2 \), but far from \( a_3 \).

It is worth noticing that the above regions may not be continuous for arbitrary anchor objects in arbitrary space. This subtle issue does not lead to any problem since we do not need to describe the extents of the regions explicitly. We only define those regions implicitly by using (1).

Algorithm 4 is the pseudocode of the construction algorithm for FDH. We first choose the anchors \( a_i \) and distance values \( r_i \) as described before. Then, for each object \( p \in P \), we apply (1) to obtain an \( A \)-length bitmap \( B_M(p) \), and we apply a standard encryption (e.g., AES) on \( p \) to obtain an encrypted object \( ECR(p, CK) \). A tuple consisting of
the object ID, transformed bitmap \(BM(p)\), and the encrypted object \(ECR(p, CK)\) is uploaded to the server.

**Algorithm 4.** FDH Building Algorithm for Data Owner.

**Algorithm** FDH-Build (Data Set \(P\), Encryption Key \(CK\), Integer \(A\))

1: for \(i := 1 \rightarrow A\) do ▷ key generation
2: choose an object randomly from \(P\) as an anchor object \(a_i\)
3: find the distance value \(r_i\) such that half of objects \(p \in P\) satisfy \(dist(a_i, p) \leq r_i\);
4: for each object \(p \in P\) do
5: compute the encryption \(ECR(p, CK)\);
6: compute \(BM(p)\);
7: send the tuple \((p.id, BM(p), ECR(p, CK))\) to the server;

Fig. 6c shows the content of the transformed data set (stored at the server), obtained from the configuration of Fig. 6b. We introduce an appropriate indexing method for the above bitmaps shortly.

### 4.3.3 Query Processing

Note that the client has the transformation key, which contains the value of \(A\), the set of anchor objects \(\{a_i\}\), and the corresponding distance values \(\{r_i\}\). At query time, the user specifies a query object \(q\) and then applies (1) to obtain the bitmap \(BM(q)\). The user then requests the server to return an encrypted object \(ENC(p, CK)\) such that its bitmap \(BM(p)\) is identical to \(BM(q)\).

A closer look reveals that the above query method has two potential problems. In Fig. 6b, different regions contain different numbers of objects. In case the query object \(q\) falls into the region “111,” none of the data objects can be retrieved. On the other hand, if the query bitmap is “101,” the corresponding region contains multiple objects, and the server cannot decide which of them is closer to \(q\) solely based on the bitmap.

These two important problems have not been considered in earlier work [6]. Intuitively, if the query bitmap falls in a region (say, “111”) without objects, we need to expand the search space to neighborhood regions (e.g., regions “110,” “100,” “101”). In case the query object \(q\) falls into a region (say, “101”) with multiple objects, we need to gather extra information to decide which of them is closer to \(q\).

It is well known that the Hamming distance is an intuitive function for capturing the distances among bitmaps. In our context, two regions close together are expected to have a low Hamming distance between their corresponding bitmaps.

At the server side, we employ a metric space index (e.g., the M-tree [11]) for indexing the bitmaps based on the Hamming distance function \(hamming()\). In other words, hash buckets (for the regions) are no longer explicitly maintained in the system. This eliminates the problems raised by empty hash buckets or buckets with many objects. As we will see later, the query processing strategy is to apply a similarity search on the above metric space index.

**Algorithm 5.** FDH Searching Algorithm for Client.

**Algorithm** FDH-Search (Query object \(q\), Encryption Key \(CK\), Integer \(\theta\), Integer \(A\), Pairs \(\{(a_i, r_i)\}\))

1: compute the query bitmap \(BM(q)\);
2: request the \(\theta\) tuples \((p.id, BM(p), ECR(p, CK))\) with the lowest Hamming distance from \(q\) from the server (ties are resolved arbitrarily);
3: let \(S\) be the set of decrypted objects from the received tuples;
4: return the object \(p \in S\) with the minimum \(dist(q, p)\) value as the final result;

The advantage of the above algorithm is that it provides the clients the flexibility of choosing different values of \(\theta\) at query time. In our experimental study, we will show that a small value of \(\theta\) is sufficient to achieve a reasonable accuracy for the reported result.

#### 4.3.4 Incremental Data Update

The data owner efficiently computes the bitmap of an object and then requests the server to insert or delete that object (with a particular object ID).

We need a mechanism to cope with the scenario that the data distribution in the data set changes gradually over time. We suggest that the data owner precomputes a bitmap (for each object) such that its length equals, e.g., 10\(|A|\). The data owner then performs a periodic (say, weekly) procedure as follows: 1) asks the server to find \(|A|\) bit positions (out of all 10\(|A|\) positions) such that the ratio of 0/1...
bits is the most balanced, and 2) publishes to the users a new key with those A bit positions only. This maintenance task is very efficient as the server needs not ship any bitmaps or encrypted data objects to the data owner.

4.4 Enforcing δ-Gap in Original Space

We now discuss how the δ-gap guarantee (in Definition 2) can be achieved by adapting our transformation methods MPT and FDH. Recall that this guarantee is used to define privacy in the original space, and it requires that: each original object \( p \in P \) must be represented by a tuple \((p, \text{dist}(a, p))\) such that \( a \) is a reference object and \( \text{dist}(a, p) \geq \delta \). This requirement has to be fulfilled in the original space before applying our transformation method. We call these extended solutions δ-MPT and δ-FDH, respectively. Only their construction algorithms are modified; the query algorithms of MPT and FDH can be directly reused for processing queries as the schema used for the transformed data remains unchanged.

4.4.1 δ-Gap Variant of MPT

In order to provide the δ-gap guarantee in MPT, we modify Line 3 of Algorithm 2 as follows, by restricting how a data object \( p \) can be assigned to an anchor object of the set \( A \). Specifically, we define the set \( A_p = \{ a \in A \mid \text{dist}(a, p) \geq \delta \} \) for object \( p \). Every anchor in \( A_p \) always satisfies the δ-gap guarantee with \( p \). Then, we assign \( p \) to an anchor from \( A_p \), by using a heuristic [11] (e.g., finding nearest anchor).

Suppose we now use the above variant to assign objects to anchors in the example of Fig. 7a, where objects \( p_1 \) and \( p_2 \) are anchors \( a_1 \) and \( a_2 \), respectively. Each dotted circle has its center at anchor \( a_i \) and its radius as \( \delta \). To meet the δ-gap requirement, an object located within a dotted circle cannot be assigned to the anchor of that circle. For instance, object \( p_1 \) can only be assigned to anchor \( a_2 \). Similarly, objects \( p_2 \) and \( p_6 \) are assigned to anchor \( a_1 \). Each remaining object can be simply assigned to its nearest anchor.

Observe that a small \( \delta \) value only forces a small number of objects to be assigned to farther anchors.

4.4.2 δ-Gap Variant of FDH

To offer the δ-gap guarantee in FDH, we displace each original object \( p \in P \) by the distance \( \delta \). This is performed before (1) is used to convert \( p \) into a bitmap in the transformed space.

Fig. 7b shows how this variant works. In this example, locations within the solid circle are mapped to the bit value 0, whereas other locations are mapped to the bit value 1.
distance-based hashing method (DBH) [6] was reviewed in Section 2.

For the construction phase and the query phase, we use the following default parameter values: \( A = 100, \theta = 500, \lambda = 10, \) and \( k = 32. \) Experiments on queries are run with a query workload of 100 query objects. We measure the communication cost (in KBytes), server CPU time (in s), the distance ratio (of the result NN distance to the actual NN distance), and the rank (of the result NN on the data set); and we report their average values for the query workload. We also measure the time taken in the construction phase.

5.2 Experimental Results

5.2.1 Experiment on Default Parameter Values

Before evaluating the effects of the parameters, we give an overview of the general performance of the studied algorithms.

Table 5 shows the query communication cost of the solutions on all the data sets. The general tendency is the same for all data sets. The straightforward approaches BRUTE and ANONY have very high-communication cost when compared to the others. EHI has a moderate communication cost because the client performs the actual search procedure. Observe that MPT and FDH greatly reduce the communication cost by outsourcing search functionality to the server. MPT obtains its reduction in the communication cost by providing the server with information on the relative distances of objects (i.e., order preserving encryption). The best performance in terms of communication cost is achieved by the FDH method.

Due to the high cost of ANONY, it is omitted from the subsequent experiments.

5.2.2 Comparison between DBH and FDH

To facilitate a fair comparison between DBH and FDH, we allocate the same amount of disk storage space to them. Both DBH and FDH use the same parameter value of \( A. \) However, for DBH, its number \( C \) of hash tables is fixed to 1, so that its storage space is comparable to that of FDH.

Table 7 shows the percentage of empty results, average NN distance, and average communication cost on the GFC data set when varying the parameter \( A. \) It is worth noticing that DBH may often return empty results when \( A \) is above 100 (see the highlighted cells). This is because each hash bucket contains very few objects in those cases. On the other hand, FDH does not return empty result in any case, thanks to its search strategy based on Hamming distances among the bitmaps. The NN distances returned by DBH and FDH are similar, but the communication cost of FDH is much lower than that of DBH.

Next, we study the rank of result NN for individual queries (from default query workload) on all real data sets, whereas other parameters are set to their default values. For an empty result (e.g., obtained in DBH), its rank is set to the cardinality of the data set. Fig. 8a shows the rank of result NN on the data sets YEAST and MUSH, and Fig. 8b shows the rank of the result NN on the data sets SHUTL and GFC. For ease of visualization, the ranks are plotted in descending order. Observe that the result rank of FDH outperforms that of DBH on the data sets YEAST, MUSH, and SHUTL. For the data set GFC, the 10 percent of the query instances with that largest ranks favor FDH much more than DBH, but the next 30 percent of the query instances favor DBH slightly more than FDH. Interestingly, a certain query object has a large rank (3,153) on the data set SHUTL. According to our log for FDH, the result NN distance of that query object is 0.014379, which is much higher than the average NN distance of the workload (0.000498). This query object is an outlier whose NN resides in a dense “cluster” located far away. Even with a small error in the result NN distance, the rank of the result NN can become large. A similar behavior is observed in DBH, and it performs worse than FDH.
inferior performance and result quality. Also, we omit DBH from further studies due to its query workload, rather than result measures of individual queries. As a result, we omit DBH from further studies due to its inferior performance and result quality.

5.2.3 Scalability Experiment

To analyze the scalability, we generate large synthetic data sets of cardinality \( N \) between 50,000 and 200,000. SYN consists of 32 dimensional feature vectors, distributed into 10 different clusters. As we can see in Fig. 9, the relative performance of all methods is stable across different data sizes. The communication cost of the BRUTE method corresponds to the size of the data set. EHI and MPT achieve the same reduction in communication cost. Unlike MPT, however, EHI cannot be further improved by tuning parameter values \((A, \theta)\). The best method in terms of communication cost is once again the FDH approach which scales extremely well. The distance ratio of FDH is insensitive to \( N \). Due to the fixed number of clusters, the number of objects within a cluster increases proportionally as \( N \), and thus the result rank of FDH increases.

5.2.4 Effect of the Selection Parameter \( \theta \)

We now turn to the parameter \( \theta \) in the MPT and FDH techniques to determine a good value that optimizes communication cost. Fig. 10 depicts the results: increasing \( \theta \) means that the client requests more tuples/bitmaps from the server. In MPT, \( \theta \) is used in the initial communication round. Hence, it influences only the quality of the initial approximation. The better this initial approximation, the lower the communication cost is in subsequent rounds. This is reflected in the experimental results where increasing \( \theta \) lowers the communication cost of MPT. Consequently, in FDH with only one round of communication, we observe the opposite effect. In this single round, the communication cost increases proportionally to the number of objects requested, and the approximation quality improves.

5.2.5 Effect of the Number of Anchors \( A \)

We then study the effect of varying the number of anchors \( A \) that are used in the MPT and FDH techniques. Fig. 11 exhibits the effects on both the communication cost and the nearest neighbor result. We observe a gradual improvement in communication cost with increasing numbers of anchors. This is due to the fact that more anchors provide more detailed information on the locations of objects. This is also reflected in the approximation quality of FDH, which improves with increasing number of anchors. With only 100 anchors, we achieve a low communication cost and a good approximation of the nearest neighbor.

5.2.6 Effect of the Order \( c \) on the \( \delta \)-Gap Solutions

We now examine the performance and accuracy of the extended solutions that satisfy the \( \delta \)-gap guarantee: \( \delta \)-MPT and \( \delta \)-FDH. Recall from Section 3.2 that we recommend setting \( \delta \) to the average \( c \)th nearest neighbor distance in the data set. Fig. 12 shows the communication cost and result measure with respect to \( c \). The cost of EHI is independent of \( c \) and it is shown as a reference for comparison. The communication cost of \( \delta \)-MPT rises with \( c \), and it can be higher than that of EHI at large values of \( c \). Regarding \( \delta \)-FDH, its communication cost is insensitive to \( c \) but its result distance ratio and rank increase with \( c \). Note that the performance and accuracy of MPT and FDH are close to \( \delta \)-MPT and \( \delta \)-FDH at a small \( c \) (e.g., 1).

In summary, \( \delta \)-FDH is preferred when the user has very limited communication bandwidth and \( \delta \)-MPT is preferred when the user requires exact answers. EHI is an appropriate solution when the data owner specifies a large \( \delta \) (and \( c \)) for privacy guarantee and the user requires exact answers.

6 Conclusions

We propose similarity search techniques for sensitive metric data, e.g., bioinformatics data, that enable outsourcing of such search. Existing solutions either offer query efficiency at no privacy, or they offer complete data privacy while sacrificing query efficiency. We introduce approaches that shift search functionality to the server. The proposed Metric Preserving Transformation stores relative distance information at the server with respect to a private set of anchor objects. This method guarantees correctness of the final search result, but at the cost of two rounds of communication. The proposed Flexible Distance-based Hashing methods finishes in just a single round of communication, but does not guarantee retrieval of the exact result. The
 experimental evaluation shows that the approximation is very close to the exact result. Both MPT and FDH are extended to satisfy the \( \delta \)-gap privacy guarantee. We demonstrate their efficiency and privacy on synthetic and real-world data.

**ACKNOWLEDGMENTS**

The research was conducted when I. Assent and C.S. Jensen were full-time employees at Aalborg University, Denmark.

**REFERENCES**


Man Lung Yiu received the bachelor’s degree in computer engineering and the PhD degree in computer science from the University of Hong Kong in 2002 and 2006, respectively. Prior to his current post, he worked at Aalborg University for three years starting in the Fall of 2006. He is now an assistant professor in the Department of Computing, Hong Kong Polytechnic University. His research focuses on the management of complex data, in particular query processing topics on spatiotemporal data, and multidimensional data.

Ira Assent received the master’s and PhD (with distinction) degrees in computer science from RWTH Aachen University, Germany, in 2003 and 2008, respectively. In 2009, she was awarded the BTW Dissertationspreis for her dissertation by the database section of the German computer society. She is currently an associate professor in the Department of Computer Science at Aarhus University, Denmark. Prior to joining Aarhus University, she was an assistant professor at Aalborg University, Denmark. Her research interests are in databases with a special focus on efficient algorithms for similarity search and data mining.

Christian S. Jensen received the PhD and DrTechn degrees from Aalborg University. He is a professor of Computer Science at Aarhus University and a part-time professor at Aalborg University. From September 2008 to August 2009, he was on sabbatical at Google, Inc., Mountain View. He has previously been on several sabbaticals at University of Arizona. His research concerns data management and spans semantics, modeling, indexing, and query and update processing. During the past decade, his focus has been on spatiotemporal data management. He has received several awards for his research. He is vice chair of ACM SIGMOD. He is an editor-in-chief of the VLDB Journal and has served on the editorial boards of ACM Transactions on Database Systems, IEEE Transactions on Knowledge and Data Engineering, and the IEEE Data Engineering Bulletin. He was PC chair or cochair for STDM 1999, SSTD 2001, EDBT 2002, VLDB 2005, MobiDE 2006, MDM 2007, TIME 2008, and DMSN 2008, ISA 2010, and ACM GIS 2011. He is an ACM fellow and an IEEE fellow. He is a member of Royal Danish Academy of Sciences and Letters, the Danish Academy of Technical Sciences and the EDBT Endowment, and he is a trustee emeritus of the VLDB Endowment.

Panos Kalnis received the Diploma in computer engineering from the Computer Engineering and Informatics Department, University of Patras, Greece, and the PhD degree from the Computer Science Department, Hong Kong UST. He is an associate professor in the Division of Mathematical and Computer Sciences and Engineering at the King Abdullah University of Science and Technology (KAUST). Prior to joining KAUST, he was assistant professor in the Department of Computer Science, National University of Singapore (NUS). In the past, he was involved in the designing and testing of VLSI chips at the Computer Technology Institute, Greece. He also worked in several companies on database designing, e-commerce projects, and web applications. In 2009, he was on sabbatical in Stanford University. His research interests include database outsourcing and cloud computing, mobile computing, peer-to-peer systems, OLAP, data warehouses, spatial-temporal and high-dimensional databases, GIS, security-privacy-anonymity.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.