International Journal of Organizational and Collective Intelligence
Volume 6 • Issue 3 • July-September 2016

RSSMSO Rapid Similarity Search on Metric Space Object Stored in Cloud Environment

Raghavendra S., University Visvesvaraya College of Engineering, Bangalore, India
Nithyashree K., University Visvesvaraya College of Engineering, Bangalore, India
Geeta C.M., University Visvesvaraya College of Engineering, Bangalore, India
Rajkumar Buyya, University of Melbourne, Melbourne, Australia
Venugopal K. R., University Visvesvaraya College of Engineering, Bangalore, India
S. S. Iyengar, Florida International University, Miami, FL, USA
L. M. Patnaik, National Institute of Advanced Studies, Bangalore, India

ABSTRACT

This paper involves a cloud computing environment in which the dataowner outsource the similarity search service to a third party service provider. Privacy of the outsourced data is important because they may be confidential data. The data should be made available to the authorized client groups, but not to be revealed to the service provider in which the data is stored. Given this scenario, the paper presents a technique called RSSMSO which has build phase, query phase, data transformation and search phase. The build phase and the query phase are about uploading the data and querying the data respectively; the data transformation phase transforms the data before submitting it to the service provider for similarity queries on the transformed data; search phase involves searching similar object with respect to query object. The RSSMSO technique provides enhanced query accuracy with low communication cost. Experiments have been carried out on real data sets which exhibits that the proposed work is capable of providing privacy and achieving accuracy at a low cost in comparison with FDH

KEYWORDS

Cloud Computing, Data Transformation, Query Processing, RSSMSO, Similarity Search

1. INTRODUCTION

There is a rapid growth of the volume and diversity of digital data produced by all kinds of commercial, scientific and leisure-time applications; to search for a desired data in such voluminous data set is a tedious task. The complex data types, such as various sensor data, time series data, gene sequence data introduces a natural requirement for search. It is difficult to search such multimedia data using typical keyword search techniques; hence the similarity search (Zezula et al., 2006) comes into picture. With the growing popularity of cloud services, the natural approach is to outsource this task to the cloud environment. Service outsourcing means that the data is provided to third party repositories that are not controlled by the data owner. The outsourced data may be sensitive and confidential, (e.g. medicine data) or valuable (e.g. collected from a scientific research (Cheng and Church, 2000, Hubble et al., 2009)) and thus the privacy of the data is given more importance.

The concept of similarity search (Zezula et al., 2006) (Raghavendra et al., 2015) is applicable to a wide range of data and infinite number of various similarity functions. The time series pattern which

DOI: 10.4018/IJOCI.2016070103

has been collected in hourly or weekly basis can be searched by the scientist for similar patterns to indicate an interesting phenomenon. The similarity search can be used for analysis of DNA patterns for understanding gene or gene groups. Similarity search is most prominently used in the field of health care. Content-based retrieval (Pepsi and Mala, 2013) using similarity search is helpful in healthcare data like X-rays, MRT out-puts, various complex electric signals. New similarity search applications are constantly being developed, ranging from language translation systems to intellectual property protection.

The standard search techniques lie in the core of the similarity search and there are infinite number of (dis)similarity functions that can be used with a wide variety of data types. When searching, the similarity query typically contains a query object and the search should return the data objects that are the *most similar* to the query according to the specified function.

In our work we mainly focus on the similarity search based on the metric space model. The metric space is an ordered pair $M = (M, d)$, where M is a domain of data objects and d is a total distance function $d : M \times M \rightarrow R$ satisfying metric postulates of non-negativity, identity, symmetry, and triangle inequality. The set of indexed objects $X \subseteq M$ is typically searched by the query-by-example paradigm, for instance by the range query

$$Range(q, r) = o \in X \mid d(q, o) \leq r, q \in M$$

or by the nearest neighbours query $k - NN(q)$ covering k objects from X with the smallest distances to given $q \in M$ (Kozak, Novak and Zezula, 2012).

Motivation: Existing solutions offer any one of the following, its either query efficiency and no privacy, or complete data privacy and less query efficiency. Metric Preserving Transformation(MPT) and Flexible Distance-based Hashing(FDH) are existing methods which shifts search functionality to the server. The MPT stores relative distance information at the server with respect to a private set of anchor objects and guarantees to fetch exact results, but it needs two rounds of communication. The FDH method takes a single round of communication, but does not guarantee to retrieve the exact result. Hence our objective is to retrieve the exact result in just a single round of communication.

Contribution: In this paper, we describe a new technique for similarity search on metric data named as Rapid Similarity Search on Metric Space Object (RSSMSO). RSSMSO supports for fast retrieval of resultant object with accuracy and it provides privacy for objects by using data transformation steps before uploading to the cloud server. We suggest new technique to overcome the drawbacks of the outsourced similarity search on metric data assets (Yiu et al., 2012).

The implication of the contributions are:

1. RSSMSO method is developed to retrive Fast similarity search on metric space data. It helps to reduces communication cost over huge cloud data.
2. RSSMSO algorithm reduces the communication cost and increases accuracy.
3. Flexible Distance-based hashing methods allow the client to specify the *theta (θ)* value for increasing the accuracy of the result. Theta value would change depending on the size of the data set. In RSSMSO algorithm, θ value would always be 1, even when data set size vary. Hence we are able to retrieve the exact result with a very low theta value in a single communication round.
4. The experiment is demonstrated on real time data set gene expression matrix data (eg: YEAST (Cheng and Church, 2000)).

Organisation: The rest of the paper is organized in the following manner; We describe the Related work in Section 2 which gives the pros and cons of similarity search. Background is described in Section 3 which lists some of the existing method for deriving the results from the server. Problem

statement and System model describes the working of the system and gives the details about the design goals these are discussed in Section 4. Proposed work describing the implementation of data transformation, query and the search phase is in Section 5.Performance evaluation results are listed in Section 6. Conclusions are presented in Section 7.

2. RELATED WORKS

We have listed out various work related to similarity search on cloud environment, along with their advantages and diadvantages.

The techniques present in (Yiu et al., 2012),(Pepsi and Mala, 2013),(Kuzu, Islam and Kantarcioglu, 2012),(Sun et al., 2014) helps in accomplishing the similarity search over encrypted data and helps in search process to happen while preserving the data privacy. FDH (Flexlible Distance Based Hashing) (Yiu et al., 2012) shift the search functionality to the server with a single round of communication, but do not assure of providing accurate result. Kuzu et al., (Kuzu, Islam and Kantarcioglu, 2012) uses LSH for fast similarity search which is tolerant to typographical errors . (Sun et al., 2014) also uses the LSH to do similarity search on images. (Pepsi and Mala, 2013) does a dynamic similarity search using content based retrieval. This ability to search dynamically was helpful in medical industry to retrieve lung images.

Consider various parameters to do similarity search (Xia et al., 2014),(Hjaltason and Samet, 2003),(Amato and Savino, 2008) various parameters to do similarity search over encrypted cloud, similarity search based on distances and search based on metric spaces respectively. (Xia et al., 2014) returns files which are semantically related to the keyword but need to protect semantic information from the files. (Hjaltason and Samet, 2003) use M-tree to have a fast similarity search, but suitable only when we have a large amount of data. (Amato and Savino, 2008) use inverted files to obtain similarity search. This can be applied to any application which works on any such paradigm and can be modelled using metric space.

Important concepts in (Ciaccia, Patella and Zezula, 1997),(Bozkaya and Ozsoyoglu, 1999),(Agrawal et al., 2004) help in carrying out similarity search on metric space. (Ciaccia, Patella and Zezula, 1997) proposed M-tree to organize and search large data sets from generic metric space. They perform well in high dimensional space and are more efficient than R* tree. (Bozkaya and Ozsoyoglu, 1999) uses MVP tree which is created in atop down fashion on a given set of data points and hence guarantees a balance tree. (Jang, Yoon and Chang, 2013) offer a spatial data base encryption scheme that produce a transformed data base by using network distance among POIs (point of interest). Hence reduces the search range.

Indexing concept mentioned in (Kozak and Zezula, 2013),(Hong Lu et al., 2006),(Bin Cui et al., 2005),(Gil-Costa and Marin, 2012) play an important role in performing similarity search. (Kozak and Zezula, 2013) propose two new similarity indexes EM-Index and DSH Index that are apt for search systems outsourced in a cloud and also guarantee data privacy. EM-Index proves profitable by supporting precise evaluation of the range queries and efficient update operations while DSH guarantee higher privacy level. (Hong Lu et al., 2006) propose an efficient solution, called the Ordered VA-File (OVA-File) which addresses the problem of content-based video indexing. A high query result is obtained in the proposed method when compared to other two existing methods VA-file based method and iDistance. (Bin Cui et al., 2005) uses an indexing structure called Δ-tree. This indexing method helps in lowering the cost of computing and cache misses because the search process can reduce the space to be searched efficiently. (Gil-Costa and Marin, 2012) works on indexed metric space where query processing can happen in parallel manner. Scheduling algorithm is applied onto a global index which gets evenly distributed on the processors and hence helps to achieve good performance.

Concepts and the ideology of K-NN (Nearest Neighbour) are discussed in (Khoshgozaran and Shahabi, 2007), (Connor and Kumar, 2010),(Wong et al., 2009),(Hajebi et al., 2011). (Khoshgozaran

and Shahabi, 2007) realizes K-NN query by mapping the static and dynamic objects after applying one way transformation to another space and the query can be resolved blindly in the transformed space(Hilbert space). (Connor and Kumar, 2010) fostered K-NN graph construction using Morton ordering. Linear list of numbers is achieved as a result of applying Morton ordering on to N-Dimensional space. The algorithm favours faster construction of K-NN graphs and uses less space. (Wong et al., 2009) formulate a new Asymmetric Scalar-Product-Preserving Encryption (ASPE) aims to support K-NN computation on encrypted data by constructing secure schemes. APSE profits by giving a very low cost and resist different overhead cost at various level of practical attacks by considering different background knowledge. (Hajebi et al., 2011) introduce a new algorithm that helps in resolving the nearest neighbor search problem by performing hill-climbing on a K-NN graph.

Similarity search concept is considered in the following references (Xia et al., 2013),(Tsymbal et al., 2014),(Popivanov and Miller, 2002). (Xia et al., 2013) performs similarity search on encrypted images based on a secure transformation method. The transformation used does not mortify the result accuracy and also keeps the confidentiality of the data intact. (Tsymbal et al., 2014) share their experience gained by translation of a similarity search-based clinical decision support system called "Case Reasoner". They help in advanced similarity search and case retrieval-based solutions with lower computational complexity. (Popivanov and Miller, 2002) permit proficient similarity search over time-series where data is of high-dimensions and considers the use of wavelet transformations for reducing the dimensions.

There are many third party security issues related to the clouds; bussiness clouds are one among them, since it cater to many people and store huge amount of confidential data. (Chang, Kuo and Ramachandran, 2016) introduce a frame work designed for business clouds called CCAF multi-layered security. It can detect and block various types of viruses and trojans . This security model would help businesses to run smoothly by proctecting their data and assets using (Chang and Ramachandran, 2015).

3. BACKGROUND

Two basic solutions exist to derive the results from the third party server while preserving the privacy:

3.1. Brute Force Secure Solution

The objects will be uploaded to the server only after the data owner encrypts them by applying a symmetric key. Actual result is calculated after the client places a query at the query time and the encrypted objects are downloaded from the server. The method is absolutely secure due to the use of encryption, but there is an increase in the communication cost due to the downloading of all the objects, even the data objects not concerned with the query. Hence the method is not suitable for the present day needs.

3.2. Anonymization-based Solution

Data privacy for the anonymization-based solution can be achieved by the *k*-anonymity and not by the encryption. Here we assume that there exists k number of objects and the generalization happens in such a way that every object that is generalized cannot be discriminated from other *k-1* objects which are generalized. Hence by following generalization scheme the transformed objects ranking can be confused. The confusion created help to represent that the *k-1* objects has the same rank as the transformed object of the actual nearest neighbor. The clustering-based anonymization technique of (Aggarwal et al., 2006) can be applied for arbitrary metric space data. Each bucket is represented by Minimum Bounding Sphere (MBS).

The anonymization-based solution has a limitation; the MBRs/MBSs (Maximum/Minimum bounding Rectangle/Sphere) may contain lot of empty space as they are dealing with multidimensional data, causing them to retrieve large number of buckets. On contrary in the proposed method the anchors

are changed to IDs and distance information is changed to numbers and only what is concerned with the query are retrieved, not less nor more.

3.3. Indexing and NN Search on Metric Space

R*-Tree: It is capable of accessing multidimensional points and spatial data. Queries and operations, such as map overlay, rectangles and multidimensional points can be easily dealt using R*-tree. Data containing many dimensions can be stored by using X-tree whose indexing structure is based on R-tree. R tree and X tree are renowned disk based indexes for multi-dimensional objects. Data objects which are complex ex: time series, cannot properly match up to by the co-ordinate values.

VP Tree: Data in the metric space is isolated by choosing a position in the space, this is called Vantage Point (Vp). The data points are divide into two partitions, those which are at close proximity to the server and those which are farther away from the Vp. Feature vector of fixed dimensions can easily be represented by index objects.

MVP: Objects can be indexed using an abstract data structure called Multi vantage point (Mvp). Similarity query can be applied on large metric spaces using distance based index which is constructed using Mvp-tree. Similarity partitioning strategy are made use in the Mvp-tree and Vp-tree where they do not use pre computed distances and use only one Vantage point.

Mtree: Mtree is a dynamic distance-based index structure for metric domains. M-tree is more competent than R*-tree in terms of input /output cost and distance optimization. They perform well in high dimensional space. The disadvantage is that the objects are entered randomly, the parent node is located by travelling from the root of the tree until the node itself is found.

3.4. Encrypted Hierarchical Index

It is a hierarchical indexing structure which is built on the Metric Space(MS) object data set; these nodes are encrypted using a symmetric key algorithm and address of the root node is made public. *Mindistance* and *maxdistance* functions are used for the data transformation which makes the algorithm secure. The search service is at the client side. The client request for nodes, decrypts them and applies the search function on these nodes; a new set of nodes are requested again from the server until the required result is found. There exists a considerable traffic between the server and the client due to multiple communication round trips; a reason for increased communication cost. There is an additional overhead on the client due to the search procedure which takes place on its side. The method suffers from relatively very low search efficiency.

3.5. Metric Preserving Transformation (MPT)

Metric Preserving Transformation makes use of an order-preserving encryption (OPE). The function $f : R \rightarrow R$ is said to be order-preserving *f(x) > f(y)* iff *x > y*. A set of anchor objects A are selected from the data set P;where each object is assigned an anchor object A_i. The distance is computed for objects with respect to anchor objects. An OPE is applied on these distances and then uploaded to the server. The application of OPE serves as the transformation function. MPT needs two round trip communications during the query phase. The method is sufficiently secure but not well suited for dynamically changing data.

3.6. Flexible Distance-based Hashing

In the build phase, a set of n anchor objects is chosen and each anchor a_i is assigned a range r_i. Then, for every object o a bitmap of length *n* is created; the *i* th bit of this bitmap equals to zero if $d(a_i, o) \leq r_i$, otherwise it equals to 1. The objects are encrypted and are stored on the server along with their bitmap representation. Here bitmap is used as the data transformation function. Here the

client is given the liberty to select and vary the θ value; the server returns θ number of objects that are closest to the bit map representation of the query. This approach accomplishes the best communication cost because of the very compact bit map representation; only drawback being it innately supports only approximate search queries.

4. PROBLEM STATEMENT AND SYSTEM MODEL

Nearest neighbour search and Range queries are key subclasses of similarity search. Similarity Search is mostly performed on the complex objects like the metric space objects. The previous methods used multiple communication rounds to get the result, hence we work on reducing the communication round to as low as 1.

Objectives:

1. The data owner allows only authorized clients to run search queries on a third party cloud server and get results from it.
2. To avoid the overhead on the client side, search should be performed on the server side as much as possible.
3. Communication cost should be as minimal as possible between the client and the server.
4. Data on the server should be stored in a secure way.

4.1. System Model

1. **Data Owner:** Owns the data and allows to outsource the search service.
2. **Server:** Server(s) is a third party similarity cloud used to store the data. The data owner does not trust the server (server can be attacked and data from it leaks to an attacker).
3. **Authorized Client:** Access the data by using the secret key and retrieves the data needed by using the search service.

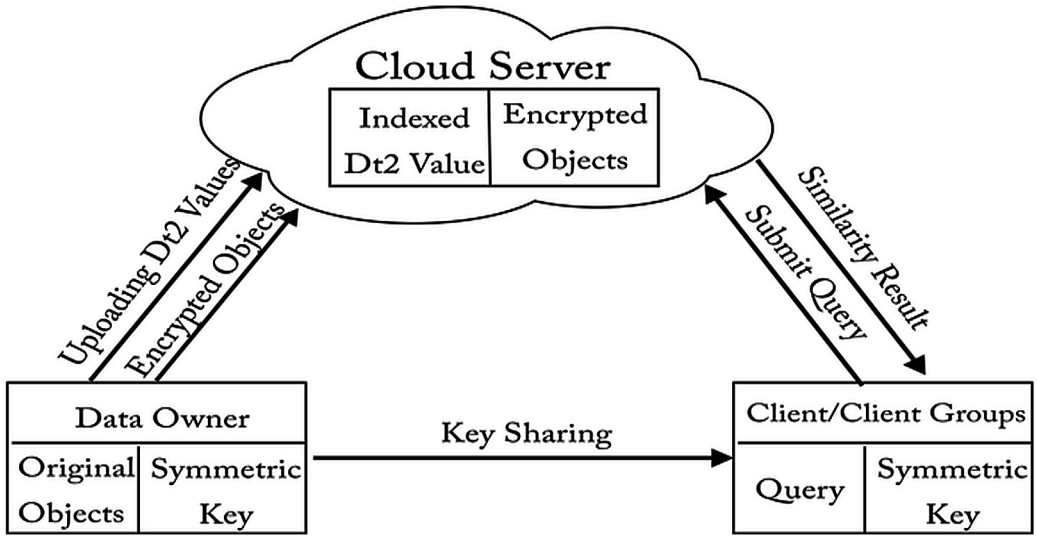
The system model has 3 elements as shown in Figure 1: *data owner*, *client* and *server*. Data owner desires that the data be made available to the authorized clients, but to do so he has to host his data on the server which he does not trust. Hence the data owner encrypts the data and uploads to the server. He applies a standard encryption method (symmetric key encryption algorithm e.g., AES) on the data set of original objects; this results in encrypted objects. These encrypted objects along with their *Ids* are uploaded to the server and stored in a relational table (or in the file system). The original objects are subjected to two steps of data transformation DT1 and DT2; the values obtained during the second step of transformation are sent to the server to be indexed (DT1 and DT2 are explained in section 5.1). This step is necessary to maintain privacy of the objects uploaded. The original data objects can be anything such as time series, graphs, strings, medical data, and scientific data. Search service is outsourced by the data owner to the server. Data owner shares a secret key with his clients. The clients having the secret key are authorized to use the search service. The client issues a query and must have the key as a proof of its authorization to the server. The server processes the query and returns the similarity result to the client.

4.2. Design Goals

4.2.1. Use of Relational Database

A relational database allows you to easily find specific information. The sorting is based on any field and generates reports which contain only particular fields from each record. A relational database

Figure 1. System Architecture



makes use of tables to store information. In a table, rows and columns correspond to field and records. Information can be quickly compared because the data is arranged in columns. The relational database is advantageous because of its uniformity, hence helpful to build completely new tables out of required information from tables which are already existing. A small table is created with the locations that can then be used for various purposes by other tables in the database. A large database, like the one a big Web site, such as Amazon and contains hundreds or thousands of tables all are used together to speedily find the exact information needed at any given time.

4.2.2. Use of Distance Object Tree Structure to Fetch the Exact Object Queried:

In a M-tree, objects are entered randomly, hence searching for a particular object takes more time. In our approach we keep the objects in a sorted way this would help in fetching the object faster.

5. PROPOSED WORK

There are four phases in RSSMSO technique:

5.1. Data Transformation

The original objects are transformed and then uploaded to the server. Transformation is important in the aspect of security; objects which are transformed and indexed on the server are having less chances of being understood by the third party like the server or the attackers. The prime concern of the data owner is the privacy of the data kept on these servers and also being able to cater to his authorized clients which lead to the need of data to be transformed. It consists of two stages for Data Transformation(DT):

DT1: The distance function $dist(a_x, b_y)$ is said to be a metric if it satisfies symmetry, nonnegativity, and triangle inequality. This value obtained from the function $dist(a_x, b_y)$ is used to compute the dissimilarity between objects a_x and b_y.

The proposed algorithm uses the Euclidean distance function. It is a straight line distance between two points in space. This distance in space are Metric Space.

Table 1. Notations

Symbols	Definition
P	Data set (set of original objects).
P'	Set of transformed objects.
p_i	Any object in the data set.
p_i |	Transformed object.
p.id	*id* of the object *p*.
CK	Encryption key.
ECR(x,CK)	Encrypting x using CK.
DCR(y,CK)	Decrypting y using CK.
$dist_{e(a,b)}$	Euclidean distance.
$dist_{o(a,b)}$	Object distance.
θ	Integer value used in query phase.
do_i	i^{th} distance object, $i = 1$ to n.
$dist_i$	i^{th} distance from the anchor object, where $i = 1$ to n.
Eo_i	i^{th} encrypted object, $i = 1$ to n.
qdist	distance of the query object from the anchor object.

Consider an Euclidean plane; X(x_1, x_2) and Y(y_1, y_2) then the distance between X and Y or the distance between Y and X is given by:

$$dist_e(X,Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \tag{1}$$

Example: Let $X(2,3)$ and $Y(4,5)$

$$dist_e(X,Y) = \sqrt{((2-4)^2 + (3-5)^2} = 2.8284 \tag{2}$$

In general, for an *n*-dimensional space, where $X(x_1,\ldots,x_n)$ and $Y(y_1,\ldots,y_n)$ the distance is:

$$dist_e(X,Y) = \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2} \tag{3}$$

Algorithm 1. RSSMSO Algorithm

Build Phase
Input: Set of original objects
Output: Distance for each object in the data set with respect to anchor object
Function: Build *(P, CK)*
1) Choose an object randomly from the data set *P* as an anchor object;
2) For each object $p_i$uptop_n where $p_i \in P$ and p_i*!= Anchor* object A_o do;
3) Compute the $dist_e(A_o, p_i)$;
Compute the $dist_o$ /*distance between anchor object and object*/
4) Compute *ECR*(p_i, *CK*); /*encrypt each object with secret key*/
5) Send the *tuple<p.id, ECR(p, CK), dist>* to the server;
Query Phase
Input: Client Sends *q*, A_o and θ to the Server
Output: Decrypted Result Object
Function: Client Request (*q*, A_o, θ)
1) Compute the $dist_e$ (A_o,*q*);
Compute the $dist_o$ ($dist_e$);/* qdist*/
2) Send query to the server
qdist ≤ distance object tree = dist1; with respect to θ
/*qdist is the query object distance with respect to the anchor object*/
qdist ≥distance object tree = dist2; with respect to θ
/*any one of the distance among dist1 and dist2 are selected, hence θ =1*/
3) Request θ *tuples* < θ=1, $dist_o$> ;
/*tuple contains the θ value and the distance value */
4) An encrypted result object is received by the client;
5) *DCR*(p_i', *CK*);/* Decrypt the object to get the final search result*/

DT2: This stage is required to give a sufficient amount of obfuscation about the values indexed in the server.

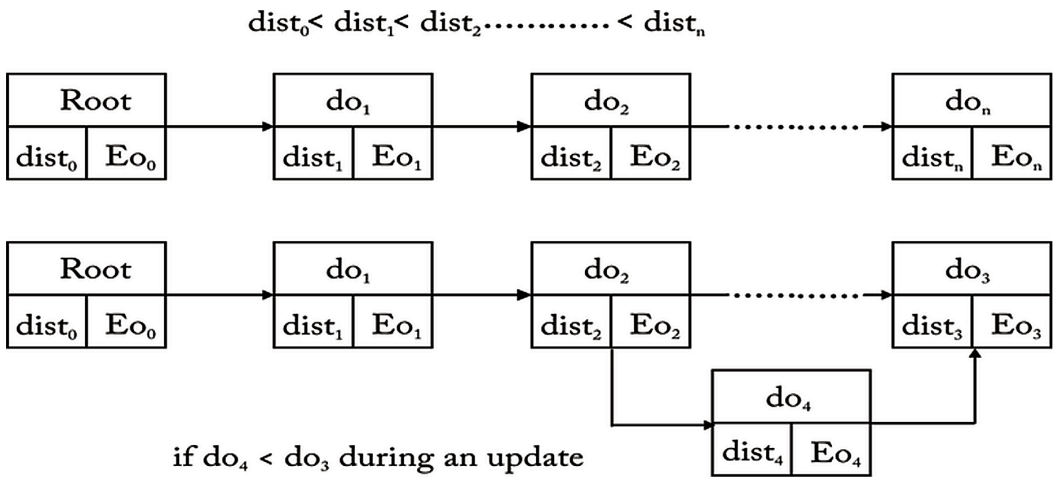
$$dist_o = \sqrt{dist_e / S} \tag{4}$$

substituting (2) in (4)

$$dist_o = \sqrt{(2.8284/2)} = 1.189 \tag{5}$$

Assume S to be the size of the data set(in terms of number of dimensions).
As a result, the original objects $P(p_1, p_2, \ldots, p_n)$ have been transformed to $P'(p_1', p_2', \ldots, p_n')$

Figure 2. Distance Object Tree



5.2. Build Phase

Build phase takes data set *P* as the input and the encryption key CK. The data set *P* can contain *n* number of objects, which can be represented as $P(p_1, p_2, \ldots, p_n)$; these objects may also be called as original objects. A symmetric key algorithm is used for encryption. The choice of our algorithm is AES, just because of its sheer advantages. The key length is around 128 bits or could have a bigger key length according to the confidentiality and the size of the data set. AES is easy to implement, more secure and less prone to attacks.

The build phase mainly occurs at the data owner side. Any random object $p_i \in P$ is selected to be an anchor object. The similarity or dissimilarity of the query object is found with respect to the other objects in the data set with reference to the anchor object. Each object is given an *id* denoted as *p.id*. Distance computation (as per Section 5.1) of all the objects in the data set is computed with respect to the anchor object. These transformed distances are indexed at the server side. In the tree structure as shown in Fig 2, every object is linked except the anchor object. The root node is having the least DT2 value (that is the object which is most nearest to the anchor object), the second one having the second least DT2 value with respect to the anchor object and so on. The data set gets sorted after it gets uploaded to the server. The tree structure might change dynamically during an update because an anchor object is randomly chosen every time when the data set is uploaded. The objects are encrypted and stored at the server side in the file system or a database. In a tuple data owner sends the *id*, encrypted object and the distance.

5.3. Query Phase

Query phase occurs at the client side. The query is submitted to the server in order to know if there exist data which is required by the client. In the previous FDH algorithm the client had to choose various θ values in order to get approximation of the result. In the proposed RSSMSO algorithm, the value of θ is 1, we get the most apt and exact result from the server side. The result received from the server is in encrypted format, the client uses its key to decrypt the object.

5.4. Search Phase

The search phase takes place on the server. The main concern is to search for most similar object with respect to the query object and at the same time maintaining the privacy of the data stored at the server.

The server finds the nearest object with respect to the queried object at the Server side when the query is received from the client:

qdist $\leq$ distance object tree = *dist1* with respect to θ // qdist is the query object distance with respect to the anchor object.

qdist $\geq$ distance object tree = *dist2* with respect to θ

if θ=*1* ; we consider a value which is greater (*qdist* $\leq$ distance object tree = *dist1*) and a value which is lesser (*qdist* $\geq$ distance object tree = *dist2*) than the query distance value. These values are fetched from the distance object tree. The value nearest to the *qdist* is chosen as the resultant object.

if θ= *2;* we consider two values which is greater (*qdist* $\leq$ distance object tree = *dist1*) and two value which is lesser (*qdist* $\geq$ distance object tree = *dist2*) than the query distance value. These values are fetched from the distance object tree. Among these four values, one value which is closest to the *qdist* is chosen as the resultant object.

Note: our aim is to always have θ=*1,* hence we consider any θ value greater than 1 as a ruled out option.

1. if *dist1* and *dist2* are equal distance from the anchor object then, *dist1* from distance object tree is sent to the client.
2. if *dist1* is more closer to the query object than *dist2,* then *dist1* is sent from distance object tree to the client.
3. if *dist2* is more closer to the query object than *dist1* then, *dist2* is sent from distance object tree to the client.

The following example shows how the algorithm works for various theta values:
Example:

When $\theta = 1$

- if the query object distance with respect to the anchor is 7 and the following are distances in the distance object tree; 3,4,5,8,10
- 7 is compared with values stored in the tree
- *qdist* $\leq$ distance tree = *dist1*

$7 \leq 8$, hence the dist1=8

- *qdist* $\geq$ distance tree = *dist2*

$7 \geq 5$, hence the *dist2* = 5

- *dist1* is chosen to be sent to the client as it is only one unit far from the query, where as *dist2* is 2 units away from the query
- More closer the distance from the query, more accurate is the result.

When $\theta > 1$; assuming θ =2, two value with respect to *dist1* and *dist2* are chosen from the tree, the nearest among the 4 values is chosen to return to the client. The nearest would be one among the four values .This is a ruled out option since exact result is got when θ =1, θ >1 would increase the overhead.

When $\theta < 1$; it is not possible since the result has to be returned to the client.

6. PERFORMANCE

We evaluate the performance of the RSSMSO based on the real world data set gene expression data matrix. The experiment involves a server and a client. The implementation is done using Java on windows platform using *Intell core i3* CPU 3217 U, with a processor speed of 1.80 GHz. The distance tree is hosted on Open shift cloud; Redhat. Gene expression data matrix gained from a Microarray experiment on YEAST. Each entry signifies the expression level of a specific gene at a specific

Table 2. Comparison of result measure

FDH (Yiu et al., 2012)		RSSMSO	
θ	Result measure	θ	Result measure
10	163.521	1	0
50	150.087	50	0
200	121.07	200	0
300	113.688	300	0
450	108.056	450	0
600	108.056	600	0
900	108.056	900	0
1500	0	1500	0

condition. The data set used is highly enriched for genes of similar function. YEAST expression matrix datasets is taken from (Cheng and Church, 2000). The matrix consists of 8224 rows and 17 columns, each element occupies 4 bytes of data, where -1 in the matrix indicates a missing value.

In FDH algorithm the value of θ should be varied in order to get a result nearer to the query object. Table 2 shows the result measure of FDH in comparison with the RSSMSO algorithm. At FDH, lesser the θ value, greater the result measure and a very high θ value is needed to make the result measure 0. The RSSMSO algorithm returns the exact result at one communication round without varying the θ value. The result measure remains 0 for any value of θ. The time complexity applicable for the result measure with respect to FDH (Yiu et al., 2012) is O(n) where θ value has to be varied n times to get result measure as zero, in comparision to RSSMSO the complexity is O(1) because result measure is zero at $\theta = 1$

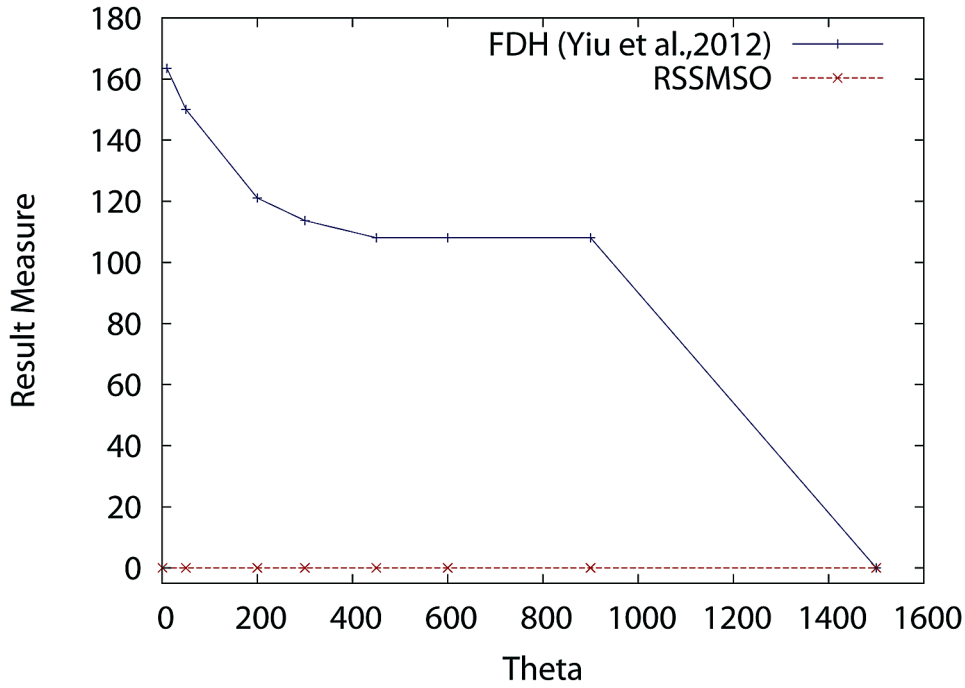
In Figure 3, FDH algorithm for $\theta = 10$, the graph starts from higher value; the θ value increases then the result measure reduces slowly. For $\theta = 600$ and $\theta = 900$ there is constant line RM=108 and hence we increase the θ to a higher value; for example, if $\theta = 1500$ then RM=0, hence there is a steep reduction in the graph. The RSSMSO algorithm shows no variations and hence the graph remains constant at zero, because the result measure is zero starting from the first attempt.

Table 3 shows reading based on communication cost. Communication cost is the amount of data that is transferred between the server and the client while varying the value of θ. The communication cost increase in FDH as the θ value increases, the proposed RSSMSO algorithm gets the exact result at θ=1. In FDH, $\theta = 10$ gives communication cost of 12.302 Kb. As the value of θ is varied to get the exact result which occurs at $\theta = 1500$ the communication cost of 1829.526 Kb is very high in comparision to RSSMSO where the communication cost is 1.342 Kb with $\theta = 1$. The complexity of communication cost between client and the server in FDH is *O(n)*, because the value of θ is varied n times until exact results is found. In comparison to FDH, RSSMSO has complexity *O(1)*, since the result is obtained at the first attempt; the communication cost will be low between the client and the server.

Table 4 shows the comparison between existing method FDH (Yiu et al., 2012) and the proposed method RSSMSO with respect to various parameters like θ, Communication cost (in terms of Kbytes and in millisecond) and result measure.Where, θ is an integer value required to get the accurate result.

Communication Cost: Measure of cost during communication between server and the client. This can be measured in 2 terms:

Figure 3. Distance between the Query Object and Result Object in Comparison with FDH and RSSMSO



A. Kilo bytes(Kb): amount of data communicated from server to client after the query has been submitted to the client.
B. Milli seconds (ms): time required to send the result after every query.

Result Measure (RM): Distance between the query object and the result object.

The first row in the table 4 shows the experiment done on a data set of having 6273 objects. FDH requires various values from (1, 10 and 20) to get the accurate result. RSSMSO on the other hand needs θ=1 to get the accurate results. The cost in terms of bytes is more 1.353+12.501+24.853=38.707 Kb) compared to (1.341 Kb) in RSSMO. The cost in milli seconds (143+116+118=877 ms) compared to 165 ms to get the accurate result. RM=0 represents that there is zero distance between the query object and the result object. Experiments have been carried out on various sizes of objects to prove the efficiency of the proposed system.

Table 3. *Communication cost with respect to θ value*

FDH (Yiu et al., 2012)		RSSMSO	
	Communication cost in Kb	**θ**	**Communication cost in Kb**
10 50 300 600 1500	12.302 61.054 366.014 731.790 1829.526	1	1.342

Table 4. *Comparison between FDH and RSSMSO with respect to θ, Communcation Cost and Result Measure*

	FDH (Yiu et al., 2012)				RSSMSO			
Size	**θ**	**Cost**	**ms**	**RM**	**θ**	**Cost**	**ms**	**RM**
369×17 = 6273	1	1.353	143	215.242	1	1.341	165	0
	10	12.501	116	63.984				
	20	24.853	118	0				
700×17 = 11900	1	1.353	420	310.942	1	1.341	260	0
	150	185.813	110	52.431				
	200	247.729	461	0				
1500×17 = 25500	1	1.358	202	160.885	1	1.342	189	0
	150	185.990	110	42.556				
	400	491.566	732	0				
2885×17 = 49045	1	1.354	252	327.434	1	1.342	189	0
	300	371.938	495	42.556				
	1500	1856.558	15556	0				

7. DISCUSSION

Our aim is to reduce the communication round to as low as 1. We consider communication round as number of times the server and the client need to communicate to get to the result object.

Distance function is closely related to similarity function. A similarity function is defined over pairs of points which measures the similarity of the two points. Similarity function is inversely realted to distance function. If a pair of points are very similar to one another, the distance between them is small. We make use of a distance function (Euclidean distance) . DT1(Data Transformation1) and DT2(Data Transformation2) are combinedly used in both build phase and quey phase. We also consider them as steps for data transformation.

- DT1 is used for finding the similarity distance between the objects and also we consider it as the first step to data transformation. The similarity value gained from DT1 is used in the second step of transformation.
- If we store the DT1 values on the server there are possibilities that an intruder would observe the values and the privacy of the data owner would be invaded. Hence we introduce a second step "DT2" to further transform the DT1 values. Data owner computes DT1 and DT2 on the metric space objects (original objects). To add security, only DT2 values are uploaded to the server by the data owner. The distance object tree stores the DT2 values on the server. Similarly, the client computes the DT1 and DT2 values, but sends only the DT2 values to match for a similar object on the server during the query phase.

Theta (θ) is a measure used to increase the accuracy of the results (result object) fetched from the server and reduce the communication cost.

- Our aim is to have $\theta=1$, hence we consider any θ value greater than 1 as a ruled out option.
- θ value is mentioned by the client when it communicates to the server in order to have better accuracy. Client considers starting from $\theta=1$ and it does not exceed 1.

When θ=1

- Only one tuple (result object) which is exactly similar to the query object is fetched from the server in a single communication round, hence low communication cost and less time consuming.
- RM=0 the object which is exactly similar to the query object is fetched from the server.

8. CONCLUSION

In this paper, we propose a similarity search technique which works on cloud computing environment without letting the server invade the privacy of the data stored in it. Existing solutions offer tradeoffs between privacy, query accuracy and communication cost. We introduce a concept which outsources the search service to the server. The proposed RSSMSO algorithm performs data transformation on the original objects before storing it on the server which ensures privacy. It also retrieves accurate result at a θ value as low as 1; hence reduces communication cost between server and the client. We express its efficiency by experimenting on real data set. We have compared with FDH (Yiu et al., 2012) our proposed method outperform FDH in terms of communication cost and accuracy. It can be applied to any application where the similarity search hypothesis can be modelled using metric spaces.

REFERENCES

Aggarwal, G., Feder, T., Kenthapadi, K., Khuller, S. R., Panigrahy, R., Thomas, D., & Zhu, A. (2006). Achieving Anonymity via Clustering.*Proceedings of the Twenty-Fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (pp. 153-162). doi:10.1145/1142351.1142374

Agrawal, R., Kiernan, J., Srikant, R., & Xu, Y. (2004). Order Preserving Encryption for Numeric Data. *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data* (pp. 563-574). doi:10.1145/1007568.1007632

Amato, G., & Savino, P. (2008). Approximate Similarity Search in Metric Spaces using Inverted Files. *Proceedings of the 3rd international conference on Scalable information systems* (pp. 1-10). doi:10.4108/ICST.INFOSCALE2008.3486

Bozkaya, T., & Ozsoyoglu, M. (1999). Indexing large metric spaces for similarity search queries. *ACM Transactions on Database Systems*, *24*(3), 361–404. doi:10.1145/328939.328959

Chang, V., Kuo, Y., & Ramachandran, M. (2016). Cloud computing adoption framework: A security framework for business clouds. *Future Generation Computer Systems*, *57*, 24–41. doi:10.1016/j.future.2015.09.031

Chang, V., & Ramachandran, M. (2015). *Towards achieving Data Security with the Cloud Computing Adoption Framework* (pp. 1–1). IEEE Transactions on Services Computing.

Cheng, Y., & Church, G. (2000). *Biclustering of Expression Data*. Retrieved from http://arep.med.harvard.edu/biclustering

Ciaccia, P., Patella, M., & Zezula, P. (1997). DEIS-CSITE-CNR. *Proceedings of the International Conference on Very Large Data Bases* (pp. 426-435).

Connor, M., & Kumar, P. (2010). Fast construction of k-nearest neighbor graphs for point clouds. *IEEE Transactions on Visualization and Computer Graphics*, *16*(4), 599–608. doi:10.1109/TVCG.2010.9 PMID:20467058

Cui, B., Coi, B. C., & Su, J. (2005). Indexing high-dimensional data for efficient in-memory similarity search. *IEEE Transactions on Knowledge and Data Engineering*, *17*(3), 339–353. doi:10.1109/TKDE.2005.46

Gil-Costa, V., & Marin, M. (2012). Load Balancing Query Processing in Metric-Space Similarity Search. *Proceedings of the12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)* (pp. 368-375). doi:10.1109/CCGrid.2012.30

Hajebi, K., Abbasi-Yadkori, Y., Shahbazi, H., & Zhang, H. (2011). Fast Approximate Nearest-Neighbor Search with k -Nearest Neighbor Graph.*Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 1312-1317).

Hjaltason, G., & Samet, H. (2003). Index-driven similarity search in metric spaces. *ACM Transactions on Database Systems*, *28*(4), 517–580. doi:10.1145/958942.958948

Hubble, J., Demeter, J., Jin, H., Mao, M., Nitzberg, M., Reddy, T., Wymore, F., Zachariah, Z., Sherlock, G. and Ball, C. (2009). Implementation of GenePattern within the Stanford Microarray Database. *Nucleic Acids Research*, 37(Database), pp. D898-D901.

Jang, M., Yoon, M., & Chang, J. (2013). A k-Nearest Neighbor Search Algorithm for Privacy Preservation in Outsourced Spatial Databases. International Journal of Smart Home, 21, 239-247.

Khoshgozaran, K., & Shahabi, S. (2007). Blind Evaluation of Nearest Neighbor Queries using Space Transformation to Preserve Location Privacy. In Advances in Spatial and Temporal Databases (pp. 239-257). doi:10.1007/978-3-540-73540-3_14

Kozak, S., Novak, D., & Zezula, P. (2012). Secure Metric-Based Index for Similarity Cloud. In Secure Data Management (pp. 130-147). doi:10.1007/978-3-642-32873-2_9

Kozak, S., & Zezula, P. (2013). Efficiency and Security in Similarity Cloud Services.*Proceedings of the Very Large Data Base Endowment* (pp. 1450-1455). doi:10.14778/2536274.2536334

Kuzu, M., Islam, M., & Kantarcioglu, M. (2012). Efficient Similarity Search over Encrypted Data. *Proceedings of the IEEE 28th International Conference on Data Engineering (ICDE)* (pp. 1156-1167). doi:10.1109/ICDE.2012.23

Lu, H., Ooi, B. C., Shen, H. T., & Xue, X. (2006). Hierarchical Indexing Structure for Efficient Similarity Search in Video Retrieval. *IEEE Transactions on Knowledge and Data Engineering*, *18*(11), 1544–1559. doi:10.1109/TKDE.2006.174

Pepsi, M., & Mala, K. (2013). Similarity Search on Metric Data of Outsourced Lung Images. *Proceedings of theIEEE International Conference on Green High Performance Computing (ICGHPC)* (pp. 1-6). doi:10.1109/ICGHPC.2013.6533912

Popivanov, I., & Miller, R. (2002). Similarity Search over Time-Series Data using Wavelets.*Proceedings of the 18th International Conference on Data Engineering* (pp. 212-221). doi:10.1109/ICDE.2002.994711

Raghavendra, S., Geeta, C., Shaila, K., Buyya, R., Venugopal, K., & Patnaik, L. (2015). MSSS: Most Significant Single-keyword Search over Encrypted Cloud Data.*Proceedings of the 6th Annual International Conference on ICT: Big Data, Cloud and Security* (pp. 43-48).

Sun, X., Zhu, Y., Xia, Z., Chen, L., Li, T., & Zhang, D. (2014). Enabling Similarity Search over Encrypted Images in Cloud. *Information Technology J.*, *13*(5), 824–831. doi:10.3923/itj.2014.824.831

Tsymbal, A., Meissner, E., Kelm, M., & Kramer, M. (2014). Towards Cloud-Based Image-Integrated Similarity Search in Big Data. *Proceedings of theIEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)* (pp. 593-596). doi:10.1109/BHI.2014.6864434

Wong, W., Cheung, D., Kao, B., & Mamoulis, N. (2009). Secure k N N Computation on Encrypted Databases. *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data* (pp. 139-152).

Xia, Z., Zhu, Y., Sun, X., & Chen, L. (2014). Secure semantic expansion based search over encrypted cloud data supporting similarity ranking. *Journal of Cloud Computing*, *3*(1).

Xia, Z., Zhu, Y., Sun, X., & Wang, J. (2013). A Similarity Search Scheme over Encrypted Cloud Images based on Secure Transformation. *International Journal of Future Generation Communication and Networking*, *6*(6), 71–80. doi:10.14257/ijfgcn.2013.6.6.08

Yiu, M., Assent, I., Jensen, C., & Kalnis, P. (2012). Outsourced Similarity Search on Metric Data Assets. *IEEE Transactions on Knowledge and Data Engineering*, *24*(2), 338–352. doi:10.1109/TKDE.2010.222

Zezula, A. G., Dohnal, V. and Batko, M. (2006). The Metric Space Approach. *Proceedings of the Advances in Database Systems* (Vol. 32).