EDCP: Effective Decomposable Closest Pair Algorithm for Distributed Databases

Ahmed M.Khedr

***Abstract*—Most of the existing computing algorithms are designed for environment in which all the related data are stored at a single site, horizontally distributed or particular case of vertically distributed where different nodes contain different attributes of a common set of entities. Moreover, existing algorithms that work for general vertically distributed databases have exponential messages and elapsed time, resulting in heavy network traffic. To minimize the network traffic, it is desirable to maximize the amount of local computations on each of the participating nodes. In this work, a new perspective on these two competing demands is presented to achieve a scalable and secure solution on the most general vertical and horizontal distribution of databases. The proposed algorithm efficiently compute inter-tuple distance based clustering for data in the federated databases. We show this by finding the closest pair. The computation is executed by reciprocity involving only high level of summaries, i.e., the actual tuple can never be detected by the intruder, providing high level of privacy and security of databases. The simulation results show that the proposed algorithm efficiently find exact solution with less total message exchanges and elapsed time than existing algorithm.**

***Index Terms*—Closest Pair; Decomposable Algorithm; Directed Acyclic Graph; Horizontally and Vertically Partitioned; Privacy preserving.**

I. Introduction and Motivation

New collaborations of geographically distributed data from different fields are indispensable for many types of cooperative computations with their local databases. Some of the data were created as distributed databases where aggregation was in the interest of cooperation, but most data were created to work as self-supporting databases. For some specific computation a number of these self-supporting databases may have to cooperate, and thus the problem of designing and maintaining an efficient, secure cooperation technique among the distributed databases is an essential task. Also, it is required to maintain the privacy of the distributed data through the new technique, which returns the exact result as that of running traditional algorithm by moving all the local databases to a single site.

In a typical setting a number of databases may decide to collaborate to collectively perform some global computation. Each database then contributes its information, in the form of some partial results derived from its data, while minimizing the amount of information to be exchanged, maximizing the privacy protection of its data, and requiring the least amount of coordination to control the flow of the global computation. In this work, we propose a new mechanism and algorithm for closest pair determination with such collaborative communication and computation formulations for global computations. The primary issues that arise in developing such a formulation and have been addressed by us, include the following points:

1) Any pair of databases may share some arbitrary set of attributes. A composition function that accounts for this arbitrary nature of overlap must be defined for the participating set of databases.
2) A communication pathway connecting all the participating nodes must be established for local results to travel, and get aggregated as they travel up to a single accumulating node.
3) The formulation must minimize the risk, possibly eliminating it completely, of a data tuple from one database being revealed to other databases.

A. Integration of Distributed Data

Types of Data Federated: The participating databases may form an implicit global database that is horizontally partitioned or vertically partitioned with arbitrary overlap among their attribute sets. Some recent research has focused on horizontally partitioned datasets and on vertical partitions with very restrictive sharing of attributes among the individual databases. Our formulations and algorithm developed and presented in this paper are designed for arbitrary overlap among the participating databases.

Communication: In any instance of the global computation we assume that one of the participating nodes is the one that needs the result of the global computation, and we mark it as the Learning node. This is in contrast to the assumption in [1] where all the collaborating nodes must know the final result of the global computation. A simple communication model is one in which the Learning node directly communicates with each of the other databases [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12]. A more efficient communication may be a tree-like structure with the Learning node at the root. Messages flow up and down this tree and information is synthesized or inherited as it moves up or down the tree structure. This structure better preserves the locality of information and hence enhances the data privacy. Also, it shares the computational load of performing the aggregation among various nodes. Construction of this communication tree is driven by the pattern of attribute sharing among the databases.

Security and Privacy: In terms of a database, security is relative to the knowledge of complete tuples, sensitive subset of attributes and their corresponding tuples, corruption of the data, and capture of the data by malicious agents. Ignoring the communication aspect and considering just the computation of the joint function or relation introduces the privacy issues. In [13] privacy means that the communicating parties do not disclose their actual data during or after the communication. Only the final results of computations are

Manuscript received September 14, 2019; revised February 15, 2020.
Professor Ahmed M.Khedr is with Department of Computer Science, University of Sharjah, UAE, Email: akhedr@sharjah.ac.ae.

allowed to be disclosed. Security on the other hand relates to the interception and disclosure of the communication messages sent between the parties. Both privacy and security have associated levels of granularity. At one level, these deal with the entire database or an entire message that would be at risk. The protection of these usually involves encryption, or some form of perturbation of the data. At the lowest level of the security and privacy are the computations themselves. Their inputs and outputs are protected by some use of secure multi-party computation. Both of these leaves a middle level which is referred to in this research as the information level, e.g., statistical summary level. This level addresses the concern of the loss of utility in the information due to distortion, and the high computation cost of the secure multi-party computation. This level has to address the inferencing problem, but it has available to it many interesting properties that can be exploited to aid scalability, protecting the computation and the messages exchanged.

Distributed Computation with Privacy and Security: Computation over a secure channel highlights the privacy issues while those over non secure channels are concerned with both privacy and security. Federated databases allow great flexibility in data storage, fault tolerance, and data availability. This added flexibility introduces significant challenges when computation needs to aggregate a subset of the distributed databases to reason on. The health-care industry collects medical and insurance records of millions of patients. Naturally, people are wary of their personal and private information getting in the wrong hands. This, however, does not call for abolishing the practice of data collection. While, consumers may indeed be worried about their private data, they also enjoy the services made available to them because of this. So, instead, a balance must be struck between data collection, processing and storage and security. The exponential blow up in data size associated with explicit rendering of the aggregated databases renders most centralized algorithms to be unfeasible. With data privacy in mind, this paper presents the closest pair problem modified as a decomposable algorithm for use in distributed datasets. The goal is to present an algorithm that provides better data privacy by using the Directed Acyclic Graph (DAG) data structure. The main is that local computation is done at each site and so there is no need to send the full tuple to a central site.

B. Data Structure

In a distributed database scenario, there are n databases located at n distinct sites of the network and these distributed databases jointly produce the global implicit D for computation. Any two arbitrary distributed databases may share one or more attributes. In the situation modelled here, a graph $G(V, E)$ is used to represent the pattern of attribute sharing among the databases, where ($|V| = n$) represents the number of participating sites (each site is represented by a node), and $|E|$ represents the number of edges between all participating nodes. We add an edge between two nodes when their databases are sharing at least one attribute. We assume that, for any two adjacent participating nodes the shared attribute-names between them are known. This graph structure was not efficient for the messaging requirements, and the presence of cycles can falsify calculated results. Therefore, a DAG has been used to control the movement of information among participating nodes, and avoiding cycles can improve the number messages that must be exchanged between participating nodes, therefore, a DAG has more desirable properties. Any node can be chosen as a Learner node, to manage the global computations. DAG provides parent/child relationships among the nodes. There are two types of topologies for the constructed DAG. The first topology is a directed tree rooted at the Learner, in which all nodes have a single-parent node as shown in Fig. 1. The second topology is a general DAG (or general digraph) rooted at the Learner, in which some nodes may have a multi-parent node as shown in Fig. 2. We assume that there can be arbitrary overlap in the attribute sets of any pair of databases.

In abstract modeling, every database D_i at i^{th} site is represented by a flat table that consists of a set of tuples with set of attributes X_i. For any pair of relations D_i and D_j the sets X_i and X_j may share some attributes (set S_{ij}). The implicit global data D with which the computation is to be executed is a subset of the set of generated tuples by a join of the participating D_1, D_2, $\ldots D_n$ and D cannot be explicitly generated at any site because D_i's cannot be moved in their entirety to other sites. The tuples of D, therefore, must stay only implicitly specified to an agent. This disability of explicitly creating the tuples of D is the major challenge (generalized decomposition of global algorithms) in handling many problems which will be discussed in this paper.

For any parent and child nodes (databases D_i and D_j, $i \neq j$) they share some attributes, and we define the set of all those attributes that are shared between the parent and its children as S_{ij}.

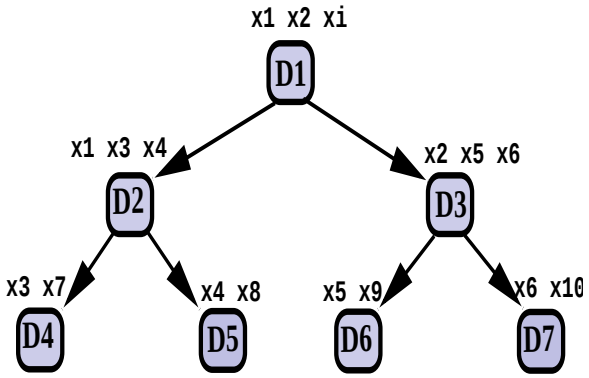


Fig. 1. Single-Parent DAG

C. Cost Models for Algorithmic Complexity

Traditionally, the algorithm complexity is evaluated in terms of memory and CPU time, however, this cost model is well-suited for computations on a single computer and the closely-coupled processors. When a number of loosely networked nodes are involved in a cooperative computation, the communication cost becomes the overwhelmingly dominant component of the total cost. According to our experience in designing and analysis of network decomposable algorithms, every algorithm step must have a number of exchanged messages for computing the various quantitative values. In this work and previous works [3], [4], [5], [6], [8], we use

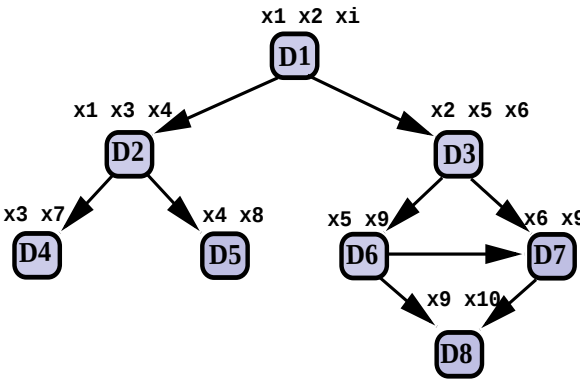


Fig. 2. Multi-Parents DAG

complexity models that involves the number of exchanged messages and reflecting the efficiency of decomposition carried out by the network algorithm. The following two Models are used to analyze our algorithms.

- **Cost Model 1: Exchanging One Summary Per Message (Un-optimized)** only one local computation request is exchanged per message at a time. The messages are exchanged in Bottom-Up or a Top-Down manner, that is, the request message is sent from upper level node(s) to the next down level nodes and the summary information is sent from down level nodes to upper level node(s) and so on until reaching the root node.
- **Cost Model 2: Exchanging all summaries per message (optimized)** all local requested computations which correspond to all tuples of shared will be sent to the children in one exchanged message.

In this work, in the two cost models above, we represent the complexity of the proposed algorithm as expressions in terms of number of messages that need to be exchanged.

The remaining of this work is organized as follows: In section II, we discuss the related work. The description of the proposed mechanism for handling the proposed problem with example scenario is given in section III. In section IV, the complexity analysis and the privacy and security discussions of the proposed algorithm are presented. In section V, we study the properties of our algorithm via simulation. We conclude our paper in section VI.

II. Related Research

The closest-pair problem is a computational geometry problem. The popularity of this problem becomes apparent because of its productiveness as a general-purpose means of comparing data and indexing [14], [15], [16], [17]. Given a set of m points in R^d ($d \geq 1$), and a metric to measure the distance between points. The problem is to find a pair of points whose whose distance is minimal. Many sequential, parallel, distributed, approximate, and random algorithms were proposed for finding the closest pair such as works in [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], however they assumed that the m points belong to one site, public data, or special case of distributed data.

In [22], the authors proposed algorithm to find the nearest neighbor of point q, in a set of n points in R^d, whose Euclidean distance to q is minimum. They pre-processed the set s, which helps to efficiently answer the queries. In [23], the authors proved that a set of n points in R^d can be preprocessed in O(d*n log (n)) time and O(dn) space. In [30], given a set S of n points S with metric space X, the authors proposed a technique to preprocess S which helps to efficiently find the closest point in S to a query point. In [31], the authors proposed a new technique to find the nearest neighbor, where the search returns the correct nearest neighbor with a given probability assuming that the queries are drawn from some known distribution using overlapped split tree, principal component analysis and support vector machines are used to analyze the structure of the data and training points in order to better adapt the tree structure to the data sets. All above proposed algorithms are working with data at one site. The proposed algorithm is designed for distributed databases.

A few works had been proposed in the area of distributed databases over the past few years. Most of works in this area addresses the horizontally partitioned data or very special case of vertically partitioned data where they assume that m tuples are partitioned between two sites such that each site contains a key to identify some individual, and a set of attributes that does not overlap with the attribute-set at any other site.

The task of computing from vertically distributed data while maintaining data privacy has been looked at from the view of exchanging high level summaries between the data sites [2], [3], [4], [5], [6], [8]. This is common on application areas with approximation result is not good enough, where database is vertically distributed, the databases are extremely large and arbitrary number of attribute overlap can be occurred. This is the most general view of the need for an implicit join for vertically distributed data. Also, the interconnections of the databases can have complex communication pathways which makes the global function evaluation an intimidating job. This view is the motivation for our formulation presented in the next sections. To the best of our knowledge there is no much work that explores the situations of unconstrained overlap in the database schema with the computations being performed in the implicit join of the databases.

The Algorithms that conduct computation on distributed databases can be divided into two main groups: firstly those that operate on a horizontally partitioned database, such as [32], and secondly those that operate on a vertically partitioned database such as [2], [3], [4], [5], [6], [8]. Vertical partitioned databases unlike horizontal partitioned databases have a range of interpretations, [33] for example assumes the availability of a global index that relates all the rows from each of the distributed databases. The join in this case is the union of all the databases. The effect of the union is that row i in the global joined database is row i in each of the local databases. While there are scenarios such as large business enterprises that vertically partition their large databases in this manner to increase data availability, this is not the case in environments such as databases across possibly competing entities like, universities, or businesses.

In [34] each database site is heterogeneous but locally can construct reduced dimensional data to be sent to a central site to formulate a global model of clustering properties of the data. From this centralized but significantly reduced data, Principal Components Analysis (PCA) can be done

with quantifiable bounds in error. The aggregation of the data at a site is on key attributes sent with the models. This approach computes an estimate using distorted data. It also creates a centralized, however smaller the sized data. Our approach does not make any such explicit rendering on which computation has to be performd. Furthermore, the amount of information sent across the network for their method is still much higher than what is desirable. [13] is a more general approach and computes decsion trees on exact data with the restriction of a single shared attribute between the attribute sets of the databases. [2], [3], [4], [5], [6], [8] removed the single attribute overlap assumption, i.e., arbitrary overlap but restricted the communications between the nodes to one central node communicating directly with all the other nodes. In this paper, we maintain the arbitrary overlap between attribute sets and expand the communications between the nodes. Our research takes a practical and most general approach. We assume the overlap in attribute sets can be on one or several attributes and that each of the databases can evolve independently of each other over time.

The idea is to divide the work by giving more computations to the local sites and then having them sending back summarized results rather than transmitting their data to the coordinator site such as the work in [35], [36], [8]. In [36], nearest neighbor pair was determined by transmitting more than 25% of the data sites to the coordinator. Moreover, the authors used the approximation in joining of the values if these values are not exactly matching. In [8], the authors proposed two algorithms to determine the nearest point from a given one, and the closest pair without any movement of data sites and without approximation in joining. However, the time and messages are exponential.

In the proposed work, using Directed Acyclic Graph (DAG), we propose a new framework that includes a general model and a decomposable version of closest pair algorithm (EDCP) for input stored in geographically distributed databases in most general case. The goal of our algorithm is to minimize the cost of communication among the database nodes by gathering statistical summaries at each distributed database and then passing messages describing those summaries between the participating sites which preserves the privacy of the data. The proposed technique is applicable for the horizontal and vertical distributed databases need to cooperate for finding the closest pair of points.

We believe that this is the second work that can determine the closest pair across vertically distributed data, where in the first work [8], sites exchange appropriate high level data summaries. The difference between our proposed work here and the algorithm in [8] is the communication structure, where in [8] the Learning node directly communicates with each of the other databases while in the proposed algorithm, we use DAG structure which is better to preserve the locality of information and hence enhances the data privacy, reduces the communication among the participating nodes, reduces the number exchanged messages and increases the security level where the communication will be executed only between the parent and its children. Our proposed algorithm only transmits minimal data between the parents and their children.

III. Effective Decomposable Closest Pair Algorithm (EDCP)

We consider a network with n sites and the i^{th} site contains the local database D_i. We assume that there can be arbitrary overlap in the attribute sets of any pair of databases. We use a graph $G(V, E)$ to depict the pattern of attribute sharing among the databases. It has n nodes, each representing a participating site. Here, we present an algorithm for finding the closest pair in a set of data points in d-dimensional space which are stored in geographically distributed databases $D_1, D_2, \ldots, D_n$. The minimal sufficient statistics for finding the closest pair of points are the distances between each pair of points. The algorithm shows that this can be calculated even when the data is vertically distributed among set of nodes. First we describe how to compute the distance between two points in distributed database (Decomposition of distance procedure) then we describe the details of EDCP algorithm. The EDCP algorithm includes four phases: Organization, Creating shared relation, Local computation and Global computation. In Organization phase, the DAG will be created from the n participating nodes and an edge will be added between two participating sites D_i and D_j, if their databases share at least one attribute as in [4]. It is assumed that the attribute-names present in all the databases are common knowledge and thus known to each participating node. We assume that the DAG data structure will be constructed based on the shared attributes among data sites. In Creating shared relation phase, every parent generates the relation shared using shared attributes with its children. In Local computation phase, at every parent for every shared tuple in shared relation, compute the distance between every pair of tuples corresponding to that shared tuple. In Global computation phase, the results of local computation are aggregated at the Learner.

A. Decomposition of Distance Procedure

We use Euclidean distance for generating a general form for computing distance between any pair of tuples $p1$ and $p2$ from implicit global database. The Euclidean distance between the two tuples $p1$ and $p2$ in local database is given by:

$$Dist(p1, p2) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}, \tag{1}$$

where $p1 = (x_1, x_2, \ldots, x_d)$, and $p2 = (y_1, y_2, \ldots, y_d)$. Eq. 1 can be rewritten as:

$$Dist(p1, p2) = Dist_{D_l}(p1_i, p2_i) + \sum_{D_t, t \neq l} Dist(p1_j, p2_j), \tag{2}$$

Here, D_l is the database that resides at *Learner* node, and i refers to the shared attributes between the *Learner* and its children, while $D_t(t = 1, \ldots, n)$ are all participating databases except D_l, and j refers to the unshared attributes at each local database in D_t. Finally, we can rewrite Eq. 2 as:

$$Dist(p1, p2) = \sqrt{\sum_{i}(x_i - y_i)^2 + \sum_{D_t, t \neq l} \sum_{unshared} (x_j - y_j)^2} \tag{3}$$

In Eq. 3, the first summation represents the computed distance at the *Learner*. The second summation represents the sum of distances between the unshared attributes of local sites which will be aggregated starting from leaves to level 2.

B. Organization Phase

In this phase, the main steps of DAG construction procedure to organize the data sources D_is in a DAG structure is presented. The pattern of overlapping between the participating nodes is structured by a graph $G(V, E)$. This general graph structure is not efficient in the messaging requirements and the presence of cycles can falsify the calculated results. The DAG has more desirable properties. The graph $G(V_G, E_G)$, is passed to the algorithm along with the Learner node. The output is the DAG $G'(V_{G'}, E_{G'})$ where $|V_{G'}| = |V_G|$ and $|E_{G'}| = |E_G|$. The participating nodes D_i's are the nodes V_G of the graph $G(V_G, E_G)$. We add an edge between two nodes D_i and D_j when their databases share one or more attributes. The main steps for constructing DAG are as follows:

1) Input: Learner D_l and graph $G(V_G, E_G)$.
2) Output: $G'(V_{G'}, E_{G'})$ DAG rooted at D_l.
3) Initialize $V_{G'} = D_l$,
4) Select a node $D_i \in V_G$ such that D_i has shared attribute(s) with D_l.
5) Connect D_i to G' by adding an edge between D_l and D_i.
6) Remove D_i from G, and add it to G'
7) While (V_G is not empty)
   a) Select a $D_j \in V_G$ such that D_j has shared attribute(s) with one of the leaves in G'
   b) Connect D_j to G' by adding an edge between D_j and the node that has shared attribute with it.
   c) Remove D_j from G, and add it to G'.
8) End procedure

C. Creating Shared Relation Phase

The following steps are executed at every participating node to create the shared relation between the parent and its children:

1) At every participating node D_i (from $level_{log(n)}$ to $level_{(1)}$)
   a) Send all different values of shared attributes to its parent.
2) At every non-leaf participating node do
   a) From local database, select all tuples that meet the received values from the children.
   b) Using the shared attributes and values, the parent creates the preShared relation as the cross product of the different values of the shared attributes between parent and its children.
   c) Each parent generates the relation Shared from preShared by deleting any tuple from preShared that does not meet at least one tuple at its localdata or at children data.

D. Local Computations (Bottom-Up strategy) Phase

Every participating node (from $level_{log(n)}$ to $level_1$ will execute the following steps:

1) Each parent, send the created shared relation to its children.
2) Starting from leaves to up, select all tuples that belong to a shared tuple or any combination of received shared tuples from parent and for each selected tuple v create an ordered list that consists of:
   a) the shared value of the combination between current node and its parent,
   b) the second shared value of the combination between current node and its parent,
   c) The distance d_{uv}= $\sum_{unshared}(x'_i - y'_i)^2$, where x'_i is the value of unshared attribute in v, and y'_i is the corresponding value of the unshared attribute in the second point u.
3) Return to parent only the ordered lists with minimum distance value.
4) In case of multi-parent, send all selected ordered lists to one parent and send ordered lists with zero distances to the other parents.

E. Global Computations Phase

This phase will be executed at Learner to aggregate all local distances and obtain the global minimum distance.

1) A Distance table stays at the Learner. It has three attributes:
   a) Learner-Distance to store the distance between all Learner database attributes,
   b) Children-Distance to store the distance between Unshared attributes for all children nodes,
   c) Total-Distance to store the sum and square root of Learner and children nodes distances.
2) From the received ordered lists, select all tuples that belong to any combination of the shared values.
3) For every combination of selected tuple pairs u and v:
   a) Compute the distance d_{uv} as in local computation phase then store it in the Learner-Distance column.
   b) From the received lists, store the corresponding distance of the shared values in Children-Distance column.
   c) Compute the sum and square root of the two columns and store the value in the Total-Distance column.
4) Select the minimum Total Distance from the table. This represents the distance between the closest tuples. We can find the two closest points by querying all children (top-down) for the values in the corresponding tuple pair and then join them.
5) End Algorithm

F. Example Scenario

In this section, we show a simple execution example of the proposed technique on vertically distributed data, where we have three local sites, each one has a simple data, see Table I. The 4-dimensional space global data $\mathcal{D}$ can be

TABLE I
EXPLICIT COMPONENT DATABASES AT LOCAL SITES

Site1	
x_1	x_2
4	4
5	2
2	3
5	7

Site2	
x_2	x_3
3	7
4	6
2	5
3	1

Site3	
x_1	x_4
4	5
4	3
5	6
2	1

TABLE II
THE SELECTED TUPLES AT D_1 ACCORDING TO THE RECEIVED ORDERED LISTS FROM D_2 AND D_3

x_1	x_2
4	4
5	2
2	3
5	7

defined implicitly in these tree data sites. The main goal of this example is to find the closest pair in $\mathcal{D}$.

1) **Organization Phase:** According to our organization phase the corresponding DAG is shown in Fig. 3

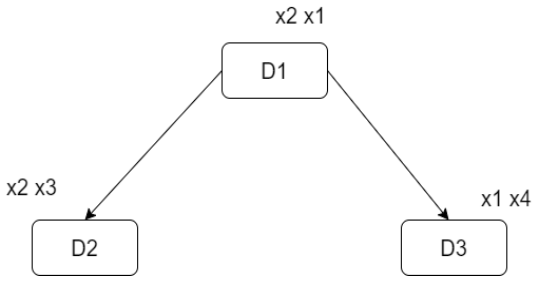


Fig. 3. The Generated DAG for the above Databases

2) **Creating shared relation phase:** Selecting the tuples of the corresponding shared values: At D_1, the values of shared attribute $\{x_2\}$ with D_2 are { 2, 3, 4} which will be sent to D_2. Similarly, D_1 and D_3 share x_1 with shared values { 2, 4, 5} and will be sent to D_3.
Local Computations:
   - At D_2, the ordered lists that will be sent to D_1 will be ⟨4, 4, 4⟩, ⟨4, 5,1⟩, ⟨4, 2, 16⟩, ⟨4, 5, 9⟩, ⟨4, 2, 4⟩, ⟨5, 2, 25⟩
   - At D_3, the ordered lists that will be sent to D_1 will be ⟨3, 3, 36⟩, ⟨3, 4, 1⟩, ⟨3, 2,4⟩, ⟨4, 2, 1⟩, ⟨4, 3, 25⟩, ⟨2, 3, 16⟩.

Global Computations:

- At *Learner* node D_1
   1) From the received ordered lists from D_2, the shared values of $\{x_2\}$ are { 2, 4, 5}.
   2) From the received ordered lists from D_3, the shared values of $\{x_1\}$ are {2, 3, 4}.
   3) From the local database D_1, we select all tuples that belong to any combination of shared values of $\{x_1, x_2\}$. The selected tuples will be as in Table III
   4) for each selected tuple, we compute the distances between all combinations as follows: ⟨4, 5, 5⟩, ⟨4, 2, 5⟩, ⟨4, 5, 10⟩, ⟨5, 5, 25⟩, ⟨2, 5, 25⟩.
   5) From the received ordered lists from D_2 and D_3 and the above computed distances, *Distance* table will be filled as in Table III.

TABLE III
DISTANCE TABLE FOR THE RECEIVED ORDERED LISTS FROM D_2 AND D_3 AND THE DISTANCES COMPUTED FROM D_1

$Dist(D_1)$	$Dist(D_2)$	$Dist(D_3)$	T_{Dist}
0	4	0	2.0
5	1	1	2.65
5	16	25	6.78
5	16	1	4.69
5	4	1	3.16
5	4	25	5.83
10	25	4	6.24
10	25	16	7.14

From the Table, first row has the minimum value and so by back tracking the corresponding closest pair will be $< 4, 4, 6, 5 >$ and $< 4, 4, 6, 3 >$.

IV. COMPLEXITY AND PRIVACY PRESERVING ANALYSIS

A. Complexity Analysis

Here, we compute the number of exchanged messages between the sites to run the proposed Effective Decomposable Closest Pair algorithm (EDCP) using DAG and stationary agents with vertically distributed data. Assume n is the number of nodes in G, and l is the average number of shared tuples at each node. We have 2 scenarios to analyze the total complexity as follows:

1) Exchange one summary per message (un-optimized): Each parent node send one message for each shared tuple and for each combination to its children to find the unshared distance, each child node will send shared tuple values and the corresponding unshared distance to its parent node, so (l+ $\binom{l}{2}$ *$log(n)$) messages will be exchanged between the children and parents during the local computation stage. Also, we need $2 * log(n)$ exchanged messages to create the shared at every parent, therefore, the total of exchanged messages by the algorithm will be:

$$Total\ Exchange\ Messages\ = log(n)*(2+l(l+1)/2) \quad (4)$$

2) Exchange all summaries in one message (optimized): Each parent node will send all values of shared attribute(s) to its children, so $log(n)$ messages will be exchanged between parents and children for computing the global unshared distance. Also, we need $log(n)$ to create the shared at every parent, therefore, the total of exchanged messages by the algorithm will be:

$$Total\ Exchange\ Messages\ = 2 * log(n) \quad (5)$$

B. Privacy Preserving and Security Analysis

In this paper, we have demonstrated that EDCP will output the same results for distributed databases as the traditional algorithm without moving and join the databases to a central node. The proposed algorithm has the following properties:

1) Instead of moving data tuples from nodes to a central node, local sites perform computation on their own data. This minimizes security risks, and increases data

privacy. It further spreads the computation load across many nodes instead of relying on one central node.
2) Only the minimum unshared distances and the shared values associated with them are transmitted.
3) The proposed algorithm exchanges only high level of information summaries between the sites and does not release any data tuple out of a database for transmission over the network.
4) The communication between nodes restricted to what is presented on the parent and the child databases.
5) The shared relation will be generated at every parent, the parent requests the shared attributes from its children, children will send a hash digest for each attribute. The proposed algorithm utilizes secure hash function to avoid sending the shared attribute names. Also, the children transmit the hash of the vector of distinct values of the shared attributes. Utilizing secret key in the hash makes it impossible for the anonymous listener to reveal the actual transmitted values.
6) Participating nodes transmit distances between the tuples that contain the received unshared actual values, rather they multiply the distance by a secret constant.

Thus in summary, anonymous listener cannot figure out the name and the values of the attributes, the distinct values of the unshared attributes or even number of tuples in any participating node.

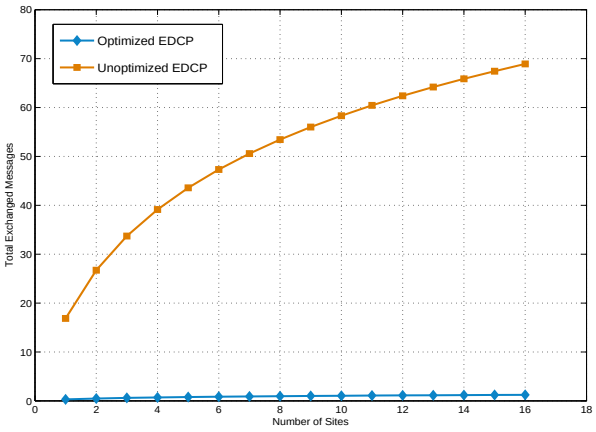
V. Simulation Results

The algorithm is simulated using Java Aglet. The tests were performed to find out the effect of various parameters on the final result. The main parameters are the following:

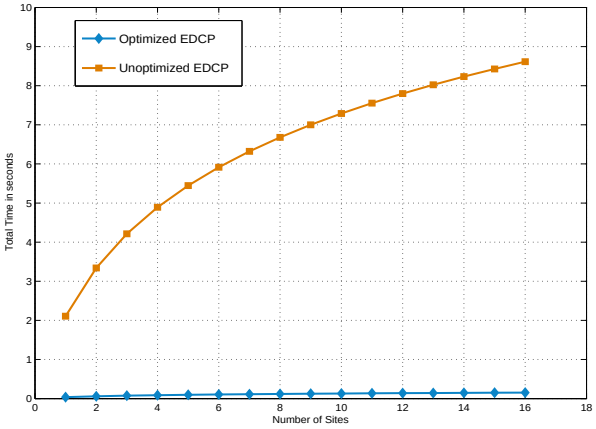
- the number of tuples per database,
- the number of sites, and
- the average number of shared tuples between the local databases.

Through the simulation, we prove that the EDCP can be efficiently executed in distributed databases environment without moving and joining the databases at central site. The tests have been executed on a network of workstations connected by a LAN and tested on the main parameters mentioned above.

1) **Number of Local Sites:** The first test was designed to show the effect of number of local sites on the number of messages exchanged and the elapsed time. We run the program by varying the number of local sites to 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 14. Fig. 4 shows how the effect of increasing the number of local sites on the exchanged messages and elapsed time in EDCP in both un-optimized and optimized scenarios. It is clear from the figure that the number of exchanged messages and elapsed time to execute EDCP increases as the number of local sites increases because as the number of local sites increases, the exchanged messages to deal with the databases at these sites increase and so the elapsed time for these messages increases.
2) **Average Number of Shared Tuples:** In the second test, we show the effect of average number of shared tuples as a main parameter on the exchanged messages and elapsed time. We vary the average number of shared tuples between local databases as 5, 10, 20, and 25. Fig. 5 shows the execution results of EDCP the exchanged messages and the elapsed time by changing the average number of shared tuples in case of un-optimized and optimized. It is clear that the total messages exchanged and the total elapsed time to run EDCP increase as the number of shared values increases.
3) **Total Number of Tuples in the Implicit Database:** In the last test, we show the effect of number of tuples in the global implicit database on the total exchanged messages and elapsed time and we compare our results with DCP algorithm [8] where the authors solve the same problem using different structure. The number of tuples in the whole database varies between 500 and 6000. Fig. 6 shows the total exchanged messages and the elapsed time for EDCP and DCP algorithms. It is clear that the proposed algorithm has less exchanged messages and elapsed time than DCP algorithm in the cases un-optimized and optimized. In Fig. 6, we can note that in case of un-optimized, the elapsed time and the exchanged messages of DCP are exponentially as the size of the database increases, and this not the case of EDCP because in DAG structure the communication

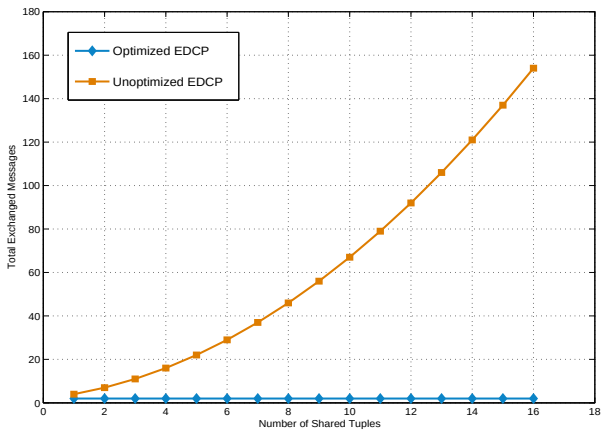


(a)

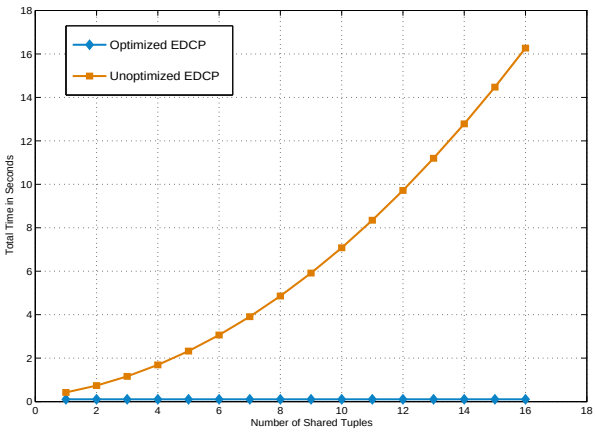


(b)

Fig. 4. Exchanged messages and Elapsed time of EDCP on vertically distributed databases with varying the number of local sites.

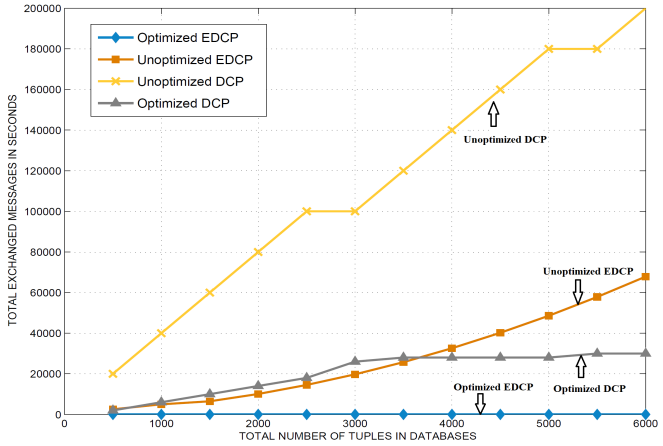


(a)

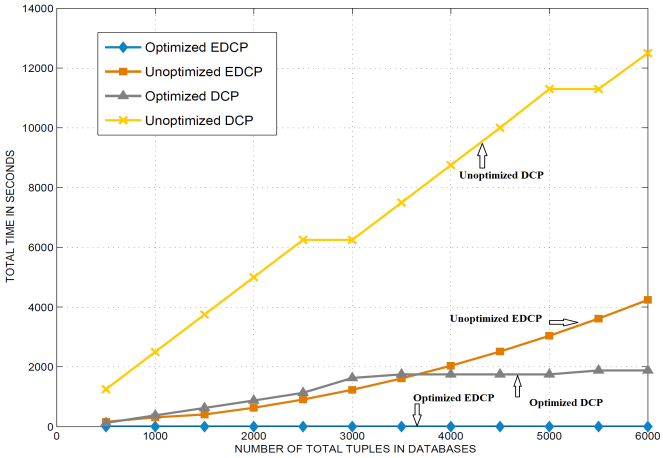


(b)

Fig. 5. Exchanged messages and Elapsed time to run EDCP on vertically distributed databases with varying the number of shared tuples.



(a)



(b)

Fig. 6. Exchanged messages and elapsed time of EDCP and DCP on vertically distributed databases with varying the number of tuples in implicit databases.

will only be between the parent and its children and this will reduce the number of messages and so the elapsed time.

VI. Conclusion

In this paper, we have presented a new efficient decomposable algorithm to find the closest pair between several points in a vertically distributed database. The algorithm uses a DAG structure to deal with data in a decomposable fashion. The algorithm achieves accurate results to those that would be obtained by moving and joining all local data to one single node, and then running the classical closest pair finding algorithm. Also, this algorithm preserves the privacy and the security of the local databases by moving high level of summaries between the parent and its children. The proposed algorithm efficiently performs global computations across geographically distributed databases without transferring any data tuples across the network. The simulation results show significant reduction in amount of disclosed information, communications, and computational costs comparing with existing algorithm.

References

[1] Z. Yang and W. Rebecca, "Privacy-preserving computation of Bayesian networks on vertically partitioned data," IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 9, pp. 1253-1264, 2006.

[2] A. M. Khedr, "Decomposable Algorithm for computing k-Nearest Neighbors across Partitioned Data," International Journal of Parallel, Emergent and Distributed Systems, vol. 31, no. 4, pp. Pages 334-353, 2016.

[3] A. M. Khedr and Raj Bhatnagar, "New Algorithm for Clustering Distributed Data using k-means," Computing and Informatics, vol. 33, pp. 1001-1022, 2014.

[4] A. M. Khedr, "Decomposable Naive Bayes Classifier for Partitioned Data," Computing and Informatics, vol. 31, pp. 1511-1531, 2012.

[5] A. M. Khedr and R. Mahmoud, "Agents for Integrating Distributed Data for Function Computations," Computing and Informatics, vol. 31, pp. 1101-1125, 2012.

[6] A. M. Khedr, "Nearest Neighbor Clustering over Partitioned Data," Computing and Informatics, vol. 30, pp. 1001-1026, 2011.

[7] A. M. Khedr and R. Bhatnagar, "A Decomposable Algorithm for Minimum Spanning Tree," Distributed Computing-Lecture Notes in Computer Science Springer-Verlag Hmidmlferg, Vol. 29, 181, pp. 33-44, 2004.

[8] A. M. Khedr and Ahmed Salim, "Decomposable Algorithms for Finding the Nearest Pair," J. Parallel Distrib. Comput., vol. 68, pp. 902-912, 2008.

[9] A. M. Khedr and Raj Bhatnagar, "Agents for Integrating Distributed Data for Complex Computations," Computing and Informatics Journal, Vol. 26, No. 2, pp. 149-170, 2007.

[10] A. M. Khedr, Ibrahim Attia, "Classification of Vertically Distributed Data using Directed Acyclic Graph," ICGST Conference on Com-

puter Science and Engineering, Istanbul, Turkey, pp. 207-215, 19-21 December 2011.

[11] A. M. Khedr and Ibrahim Atia, "Decomposable Naive Bayes Classifier for Vertically Distributed Data using Directed Acyclic Graph," ISBN: 978-0-9891305-4-7, 2014 SDIWC TAEECE2014, pp. 38-49, 2014.

[12] A. M. Khedr, "Learning k-Classifier from Distributed Databases," Computing and Informatics Journal, Vol. 27, pp. 355-376, 2008.

[13] J. Vaidya and C. Chris, "Privacy-preserving decision trees over vertically partitioned data," ACM Transactions on Knowledge Discovery from Data (TKDD) TKDD Homepage archive vol. 2 no. 3, pp.1-27, 2008.

[14] D. Baptista, J. C. Ferreira, R. Pereira, and M. Baptista, "Structured and Unstructured Data Integration with Electronic Medical Records", Proceedings of the World Congress on Engineering 2019, pp. 105-110.

[15] A. Mateen, Q. Zhu, S. Afsar, and J. Maqsood, "An Improved Technique for Data Retrieval in Distributed Systems," Proceedings of The International MultiConference of Engineers and Computer Scientists 2019, pp. 188-195.

[16] N. Ahmed, "Non-Linear Skeletal Fusion with Multiple Kinects for Unified Skeletal Animation Reconstruction," IAENG International Journal of Computer Science, vol. 45, no.1, pp63-73, 2018

[17] N. Ahmed, and M. Lataifeh, "A Comparative Analysis of Time Coherent 3D Animation Reconstruction Methods from RGB-D Video Data," IAENG International Journal of Computer Science, vol. 45, no.4, pp592-600, 2018

[18] JC. Pereira and L. FG, "An optimized divide-and-conquer algorithm for the closest-pair problem in the planar case," Journal of Computer Science And Technology vol. 27, no. 4, pp. 891-896, 2012.

[19] F. Angiulli, C. Pizzuti, "An approximate algorithm for top-k closest pairs join query in large high dimensional data," Data & Knowledge Engineering 53, pp. 263-281, 2005.

[20] Y. Park, J.K. Min, and K. Shim, "Parallel computation of skyline and reverse skyline queries using MapReduce," PVLDB, vol.6, no. 14, pp. 2002-2013, 2013.

[21] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov, "Scalable Distributed Algorithm for Approximate Nearest Neighbor Search Problem in High Dimensional General Metric Spaces," Proceedings of the 5_{th} international conference on Similarity Search and Applications, 2012.

[22] F. Garcıa-Garcıa, A. Corral1, L. Iribarne, M. Vassilakopoulos, and Y. Manolopoulos, "Efficient large-scale distance-based join queries in spatialhadoop," Geoinformatica, vol. 22, pp. 171-209, DOI 10.1007/s10707-017-0309-y, 2018.

[23] Y-R Wang, S-J Horng, and C-H Wu, "Efficient Algorithms for the All Nearest Neighbor and Closest Pair Problems on the Linear Array with a Reconfigurable Pipelined Bus System," IEEE Transaction on Parallel and Distributed Systems, vol. 16, no. 3, pp. 193-206, 2005.

[24] A. Abboud, A. Rubinstein, and R. Williams, "Distributed PCP Theorems for Hardness of Approximation in P," 58th Annual IEEE Symposium on Foundations of Computer Science, DOI 10.1109/FOCS.2017.12, pp. 25-36 , 2017

[25] A. Backurs and P. Indyk, "Edit Distance Cannot Be Computed in Strongly Subquadratic Time (unless SETH is false)," in Proc. of the 47th Annual ACM SIGACT Symposium on Theory of Computing (STOC), pp. 51–58, 2015.

[26] J. Gao, R. Impagliazzo, A. Kolokolova, and R. R. Williams, "Completeness for first-order properties on sparse structures with algorithmic applications," in Proc. of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 2162-2181, 2017.

[27] S. Arya, D.M. Mount, Approximate nearest neighbor queries in fixed dimensions, in: Proc. 4th ACM-SIAM Sympos. Discrete Algorithms, 1993, pp. 271–280.

[28] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, A. Wu, An optimal algorithm for approximate nearest neighbor searching, in: Proc. 5th ACMSIAM Sympos. Discrete Algorithms, 1994, pp. 573-582.

[29] J.M. Kleinberg, Two algorithms for nearest-neighbor search in high dimension, in: Proc. 29th Annu. ACM Sympos. Theory Comput., 1997, pp. 599–608.

[30] P. Indyk, R. Motwani, Approximate nearest neighbors: Towards removing the curse of dimensionality, in: Proc. Annu. ACM Sympos. Theory Comput., 1998, pp. 604–613.

[31] S. Maneewongvatana, D.M. Mount, An empirical study of a new approach to nearest neighbor searching, ALENEX01.

[32] R. Canetti, I. Yuval, K. Ravi, R. Michael, R. Ronitt, and N. W. Rebecca, "Selective private function evaluation with applications to private statistics," In PODC '01: Proceedings of the twentieth annual ACM symposium on Principles of distributed computing, pp. 293-304, New York, NY, USA, ACM Press, 2001.

[33] Z. Yang and W. Rebecca, "Privacy-preserving computation of Bayesian networks on vertically partitioned data," IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 9, pp. 1253-1264, 2006.

[34] H. Kargupta, H. Weiyun, S. Krishnamoorthy, and J. Erik, "Distributed clustering using collective principal component analysis," Knowledge and Information Systems Journal, vol. 3, no. 4, pp. 422-448, November 2000.

[35] A. M. Khedr, Learning k-classifier from distributed data, Comput. Informatics J. 27 (3) (2008) (in press).

[36] S. Kumar, Nearest neighbor search in distributed database, Masters Thesis, University of Cincinnati, Cincinnati, OH, 2001.