Computing and Informatics, Vol. 33, 2014, 943–964

NEW ALGORITHM FOR CLUSTERING DISTRIBUTED DATA USING K-MEANS

Ahmed M. Khedr

Computer Science Department, Faculty of Sciences
University of Sharjah, Sharjah, UAE
&
Mathematics Department, Faculty of Science
Zagazig University, Zagazig, Egypt
e-mail: akhedr@sharjah.ac.ae, amkhedr@zu.edu.eg

Raj K. Bhatnagar

School of Electronic and Computing Systems
University of Cincinnati, Cincinnati, 45221, USA
e-mail: bhatnark@ucmail.uc.edu

Abstract. The internet era and high speed networks have ushered in the capabilities to have ready access to large amounts of geographically distributed data. Individuals, businesses, and governments recognize the value of this available resource to those who can transform the data into information. These databases, though valuable as individual entities, become significantly more valuable when they function as parts of a federated database and their data can be aggregated for collective mining or computations. This requires new algorithms to shift their focus from working with single databases to efficiently working with federated databases. In this paper, we propose a new decomposable version of the popular k-means clustering algorithm that works in this desired manner with a set of networked databases. We show that it is possible to perform global computation in a reasonably secure manner for either horizontally or vertically distributed databases. The computation is completed by only exchanging a few local summaries among the databases. An empirical and analytical validation of our results is also presented.

Keywords: Data privacy, decomposable algorithm, k-means clustering, vertically and horizontally distributed data

Mathematics Subject Classification 2010: 68U99

1 INTRODUCTION

Our society is generating an ever increasing volume of data that resides in multiple databases connected by communication networks. These include data from scientific, engineering, medical, business, finance, and social domains, representing every aspect of our society. Some of these were designed and implemented as distributed databases where aggregation was in the interest of a single enterprise, but a majority of these were designed to work as independent databases. For some specific computation a number of these independent databases may want to cooperate, and thus the problem of developing efficient collaboration mechanisms among a federation of databases becomes significant. Proliferation of independent databases and the need for aggregating information from multiple such databases will present an increasing number of situations where this problem needs to be addressed. Security and privacy concerns for the data in individual databases must be addressed when they participate in collaborative computations.

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 paper, we present such cooperative communication-plus-computation formulations of algorithms for global computations for clustering k-means algorithm.

Clustering algorithms have been studied extensively in the context of single processor or parallel processor systems [1, 2, 3, 4] and there has been relatively recent work on clustering horizontally and vertically distributed databases while preserving data security and privacy during the exchange of information among the nodes [5, 7, 6].

The parallel clustering algorithms are tailored for situations in which:

1. all data resides in main memory or distributed shared memory of a set of closely connected processors;
2. a large number of closely connected processors are needed at a single site that can access the shared memory to achieve the reasonable performance; and
3. Inter-processor communication is extremely fast and involves reading data in the main or shared memory.

Our algorithm is tailored for very different situations in which we do not have closely connected processors. There are multiple processors but they are independent and reside at geographically distant sites and communication among them may be many orders of magnitude slower than the rates of inter-processor communication in a set of closely coupled processors. This is the case in many real-life scenarios. Therefore, our formalism implements a methodology wherein each site works as

much as possible with its own local data and then communicates with others at the level of local results of some partial computations.

The work in [5] discussed a privacy-preserving k-means algorithm for horizontally distributed databases that exists in two sites. In [6], the authors presented a method for k-means clustering when different sites contain different attributes for a common set of entities. Security requirements to be met in this algorithm demand that a site can know only its own attributes – value pairs when they become parts of the global cluster centers. In [7], the authors presented an algorithm for clustering high dimensional heterogeneous data using a distributed principal component analysis (PCA) technique. In their approach partial results (principle eigenvectors) are computed at each site and transmitted to a central site along with a number of data tuples corresponding to each eigenvector. The error between the actual global result and the computed result decreases as the number of tuples transmitted from each site to the central site increases. In [8], the authors discussed a method of k-means for distributed databases where a small number of passes need to be made on the entire database. This may not be feasible in all cases. Our proposed algorithm is designed for vertically and horizontally distributed datasets, does not put any limitation on the number of participating sites, and is applicable for the most general situations in which existing distributed databases want to cooperate for k-means clustering by properly accounting for any number of shared attributes. It computes global cluster centers at any of the participating sites with significantly small communication cost without transferring any tuple among the database sites which preserves local data privacy. Also, the proposed algorithm does not require even a single pass on the whole dataset, thus making it more useful in real life scenarios

Distributed knowledge discovery work such as [9, 10, 11, 12, 13, 14, 15] merges the computation with communication but either at the raw data level or at the local final result level. The former is highly insecure, and the latter adds high level of noise in the global results. Our past work [18, 16, 19, 20, 21, 22, 17] and the algorithm presented here work by exchanging summaries at intermediate level so as to preserve the data privacy and also reduce the amount of error.

The rest of the paper is organized as follows: In the following section, we briefly describe integration of distributed data. In Section 3, we give a step-by-step outline of our proposed algorithm. In Section 4, the complexity computing and the privacy and security discussion are given. The simulation results of our algorithm are given in Section 5. We conclude our work in Section 6.

2 INTEGRATION OF DISTRIBUTED DATA

In a distributed setting, a dataset $\mathcal{D}$ is implicitly defined in n explicit databases D_i located at n different sites. We model databases D_i at the i^{th} site, by a *relation* containing a number of tuples. Each D_i contains a set of attributes. Since an arbitrary

number of independent, already existing databases may be consulted for a computation, we cannot assume any data normalization to have been performed for their schemas. The implicit data set $\mathcal{D}$ with which the computation is to be performed is a subset of the set of tuples generated by a *Join* operation performed on all D_i's. However, the tuples of $\mathcal{D}$ cannot be made explicit at any network site because entire databases, D_i's, cannot be moved to other sites. The tuples of $\mathcal{D}$, therefore, must remain implicitly specified. This inability to make explicit the tuples of $\mathcal{D}$ is the main problem addressed in the generalized decomposition of global algorithms.

2.1 Nature of Data Distribution

Horizontally Distributed Datasets: The global database $\mathcal{D}$ exists as a set of components $D_1, D_2, \ldots, D_n$ such that each D_i contains tuples consisting of an identical set of attributes A; but a distinct set of data tuples resides at each site. Each D_i resides on a different site of the network and the tuples contained in all the D_i's, taken together, constitute the complete dataset $\mathcal{D}$.

Vertically Distributed Datasets: In this case, each component D_i consists of tuples formed with a different set of attributes but each D_i may share some attributes with those of some other databases, D_j, $j \neq i$. Each D_i may also contain some attributes that are unique to the local site and are not shared with a database at any other site. In effect, each D_i is a projection of the implicit global $\mathcal{D}$.

Vertically distributed datasets require computations to be performed in the implicit *Join*, $\mathcal{D}$, of all the D_is, but without ever making explicit the tuples of $\mathcal{D}$. The decomposed algorithm must appropriately account for all the shared attributes that would have played a role in enumerating the tuples of the *Join*-ed $\mathcal{D}$, if it were to be made explicit. This formulation models more general circumstances than the case of a single key and non-overlapping attribute sets for single records distributed at various sites [6]. Our target is to enable those databases for participation that were designed independently and may have arbitrary overlap of attribute sets with the other databases they have to collaborate with. The database for which the k-means clustering is performed is the implicit cross product of the relations stored at the distributed sites.

2.2 Distributed Data Sources

A number of vertically or horizontally geographically distributed databases together form an implicitly *Joined* global dataset that contains all the data relevant for mining or other computational tasks. It would be desirable to have algorithms that let the individual databases reside at their own sites and work with an imagined implicit *join* of the databases. Let us say D_1, D_2, ..., D_n are n local databases and $\mathcal{D}$ is the implicit global database formed by Merging (in case of horizontal distribution) or Joining (in case of vertical distribution) all the participating local D_i's. Let us

say a result R is obtained by applying a function (or running an algorithm) F on $\mathcal{D}$, that is:

$$R = F(\mathcal{D}). \tag{1}$$

In our case, $\mathcal{D}$, the global database, cannot be made explicit and is known only implicitly in terms of the explicit components D_1, D_2, ..., D_n. The implementation of F can now be redesigned by a functionally equivalent formulation:

$$R = G(g_1(D_1), g_2(D_2), \ldots, g_n(D_n)). \tag{2}$$

That is, a local computation $g_i(D_i)$ is performed at *Site* i using the database D_i. The results of these local computations are aggregated using the operation G. Our notion of data privacy requires that when the g_i's are exchanged over the network, even if they are captured by someone, they should not enable reconstruction of any single tuple residing in any of the participating databases. It is facilitated, partly, by the absence of knowledge of G by the network intruder; and with this constraint, any of the participating sites should be able to determine the k cluster centers for the collective data.

2.3 Addressed Problem

In this paper, we present a decomposable version of k-means algorithm, where the function F to be computed in Equation (1) is the same set of k clusters that would be produced by the k-means algorithm if the data at the networked databases were to be collected at one site. The spirit here is similar for the k-means algorithm but the final composition operator creates a close approximation of the cluster centers and not the exact ones. However, the same level of privacy protection for data is maintained.

A coordinator site that seeks to compute the global results of the algorithm first determines all the databases and sites that should be involved in a clustering task and then communicates to them requests for results of some computations performed locally at each site. Only the results of these local computations are transmitted to the coordinator site, followed by new requests for results of more local computations, until the global computation is completed and the results are obtained at the coordinator site.

At first glance, this might appear simple – every site runs the k-means algorithm on its own data. This would preserve complete privacy. Figure 1 shows why this will not work. Assume we want to perform 2-means clustering on the data in the figure. From y's point of view, it appears that there are two clusters centered at about 2 and 5.5. However, in two dimensions it is clear that the difference in the horizontal axis dominates. The clusters are actually "left" and "right", with both having a mean in the y dimension of about 3. The problem is exacerbated by higher dimensionality.

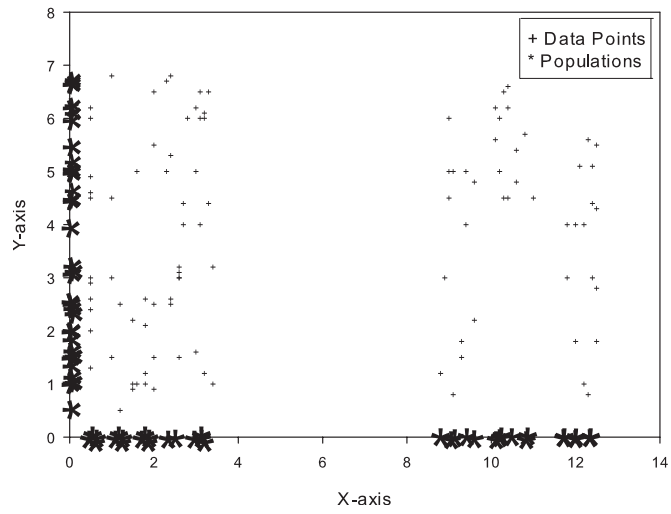


Figure 1. Two-dimensional problem that cannot be decomposed into two one-dimensional problems

3 DECOMPOSABLE K-MEANS ALGORITHM

In this section, we discuss how the algorithm performs clustering in geographically distributed databases. The algorithm includes six phases: managing the implicit join, clustering at local sites, globalizing local cluster centers, populations around globalized cluster centers, clustering globalizing data points, and recomputing cluster centers. Managing the implicit join phase shows how we deal with the implicit database. In clustering at local sites phase, we find the clustering data information at each site. In globalizing local cluster centers phase, we show how to find global cluster centers from local cluster centers. In populations around globalized cluster centers phase, we find the population of each candidate global cluster center. In clustering globalizing data points phase, using the set of globalized cluster centers, Q, available at the coordinating site, we determine the cluster centers for all the data points in the implicit $\mathcal{D}$. This is to be done in such a way that the error for k-means clustering of $\mathcal{D}$ is minimized. Finally, recomputing cluster centers phase recomputes the global cluster centers according to the computed total error.

The *total clustering error* for clustering of data points is defined as follows:

$$TotalError = \sum_{i,j} modulus(distance(p_{ij}, c_j)), \tag{3}$$

where p_{ij} is the i^{th} data point in the j^{th} cluster and c_j is the center of the j^{th} cluster. It has been shown in [24] that the clusters and cluster centers computed

by k-means algorithm are located in such a way that they minimize the magnitude of the square of *TotalError* quantity as defined above, even though it may be a local minimum they settle down in. Our decomposable version of k-means is guided by the goal to minimize a close approximation of this same error for the networked databases, and thus mimic the behavior of the k-means algorithm run on the implicit $\mathcal{D}$. An outline of our k-means algorithm for the vertically distributed databases is as follows:

1. Each site computes *local cluster centers* for its d_i-dimensional data space and sends them to the coordinating site. This is elaborated in Section 3.2 below.
2. The central site performs a cross-product of the various local clusters in smaller dimensional spaces received from local sites to generate larger dimensional (full dimensions/attributes of $\mathcal{D}$) *globalized cluster center* candidates. This is elaborated in Section 3.3 below.
3. The central site then runs a k-means algorithm on these *globalized cluster centers* candidates into desired number of k final clusters and also tries to minimize an estimate of the quantity *TotalError* mentioned in Equation (3) above. The main algorithm is explained in Section 3.5. This algorithm needs populations of data points in $\mathcal{D}$ around the potential cluster centers and the way to compute these is elaborated in Section 3.4 below.

3.1 Managing the Implicit Join

If an explicit $\mathcal{D}$ were to be generated from the D_i's, the process would have been mediated by the attributes shared among the D_i's. Let us say the set of attributes contained in relation D_i is represented as A_i. For a pair of databases D_i and D_j the corresponding sets of attributes A_i and A_j may have a set of shared attributes given by S_{ij} such that

$$S_{ij} = A_i \cap A_j.$$

For vertically distributed $\mathcal{D}$ a subset of A_i and A_j will be obtained as S_{ij} and for horizontally distributed $\mathcal{D}$ we will have $S_{ij} = A_i = A_j$. To facilitate clustering in the implicitly described $\mathcal{D}$, we define a set $\mathcal{S}$ that is the union of all the intersection sets defined above. That is,

$$\mathcal{S} = \cup_{i,j,\, i \neq j} S_{ij}.$$

The set $\mathcal{S}$, in effect, contains all those attributes that occur in more than one D_i.

We define a relation called *PreShared* on the attributes in the set $\mathcal{S}$. This relation, *PreShared*, contains tuples corresponding to all possible combinations of values for the attributes in $\mathcal{S}$. The relation *PreShared* would have mediated the creation of the explicit $\mathcal{D}$, if it was attempted, and is used by us in a very similar role. Then we generate the relation *Shared* by removing from *PreShared* all tuples that have zero count at any participating site.

3.2 Clustering at Local Sites

The following steps are executed at each site on its local database.

- The i^{th} site determines the number of local clusters, k_i, that it should generate with the local D_i. In the final composition step G, the local cluster centers will be used as representative of the data at the local sites.
- Using k-means clustering algorithm, locally cluster data in the database D_i into k_i clusters.
- Send the following information to the central coordinating site:
  - The set of local cluster centers $\{C_{ij}$: the center of the j^{th} cluster at the i^{th} site$\}$.
  - r_{ij} is the radius of the local cluster centered at C_{ij} and is the maximum of the distances between the cluster center and the points in the cluster.

3.3 Globalizing Local Cluster Centers

A cross product of the cluster centers from the local sites is computed at the coordinating site to get the *globalized cluster centers*. The cross product of local d_i-dimensional points from each local site will result in the global d-dimensional vector.

After the local clustering phase each Site i returns cluster centers that lie in one plane. Candidates for the global cluster centers, Q_i's can be generated by performing a cross-product of the cluster center points from the distinct planes. One problem in determining the Q_i's relates to determining when two values of a shared attribute B may be considered identical for the purpose of performing a cross product of the two sets of local cluster centers. That is, should a value of u for B from Site i be considered the same as a value of $u \pm \varepsilon$ for B from Site j as part of a cluster center coordinate? Cluster centers in a k-means algorithm adjust with every iteration and may be somewhat different for different sets of tuples representing the same underlying process. Therefore, we need to match the values of shared attributes only approximately for determining the globalized cluster centers. Once the globalized cluster centers are processed and adjusted in future iterations the impact of approximation would be automatically eliminated here. So, for each value x of a shared attribute, we create a $x \pm \varepsilon$ window around it and whenever two windows overlap we consider the values to be identical for the purpose of cross product.

If we replace values of B throughout with windows $x \pm \varepsilon$ then we can generate a global cluster center by choosing any of the two values as the candidate value for B. These points are only candidates for the global cluster centers and will be automatically adjusted for better accuracy in later iterations. So, they do not have to be computed exactly at this stage.

3.4 Populations around Globalized Cluster Centers

The clustering algorithm presented in later subsections of this paper requires that we compute the number of data points in the implicit data space $\mathcal{D}$ that are within some fixed distance from a global cluster center q_i when the q_i is known only at the coordinating site. We have presented a decomposable algorithm for counting tuples that match some conditions in an implicit $\mathcal{D}$. This algorithm uses some counts taken from explicit local D_i's that constitute the $\mathcal{D}$ and sends them to the coordinating site.

For each globalized cluster center q_i we find population N_q of data points around it in $\mathcal{D}$. For an implicitly stated set of tuples the counting process proceeds in such a way that each decomposed part can be sent as a request to an explicit database site and the responses composed to reconstruct the counts. The decomposition for obtaining the count N_q for each globalized cluster center candidate $q \in Q$ is as follows:

$$N_q = \sum_{tup\_in\_shrd} \left(\prod_{t=1}^{n} (N(D_t)_{tup\_in\_shrd}) \right), \tag{4}$$

where n is the number of participating database sites (D_i's), *tup_in_shrd* is a tuple selected by the coordinating site from the relation *Shareds*, as defined in Section 3.1 above, and sent as part of request to each local site; and $(N(D_t)_{tup\_in\_shrd})$ is the count of those tuples in D_t that meet the following conditions:

1. the values of shared attributes of D_t in the local tuple are the same as in the tuple *tup_in_shrd* selected from the relation *Shareds*; and
2. the value of each attribute in the counted tuple at the local site has a distance from the value of the same attribute in q_i that is less than some threshold radius value r. In effect, we are counting the number of data points in $\mathcal{D}$ that are within the hypersphere of radius r centered at the point q_i. A hypersphere of radius r in a d-dimensional space when projected to an $(d-1)$-dimensional space retains the same radius, as a sphere of radius r in 3-dimensions when projected in 2-dimensions becomes a circle of the same radius r. Therefore, it is justified to use the same value of r at each local database.

For each *sum_of_products* term in the above expression the coordinating site sends a message to the corresponding local site to obtain the count of tuples satisfying the conditions of *tup_in_shrd*. A number of such count requests may be combined in a single message to reduce the number of messages exchanged.

The product, in the above expression, is performed for the counts $N(D_t)$'s obtained from all the n sites for each tuple from *Shareds*; and the summation is performed over the product values for all the tuples *shrd*'s in the relation *Shareds*. The coordinating site stores the relation *Shareds* and sends out messages to individual sites to obtain various $N(D_t)$'s. Here the decomposition of the global counting can be related to the discussion in Section 1 in the context of Equation (2) as follows.

The *sum_of_products* is the composing function G and the individual counts $N(D_t)$'s are the local g_i's.

The expression for N_q, therefore, simulates the effect of a *Join* operation on all the n D_i's without explicitly enumerating the tuples.

3.5 Clustering of Globalized Data Points

The next task we examine is using the set of globalized cluster centers, $\mathcal{Q}$, available at the coordinating site to determine the cluster centers for all the data points in the implicit $\mathcal{D}$. This is to be done in such a way that the error for k-means clustering of $\mathcal{D}$ is minimized. To get an intuitive feel we can say that the $\mathcal{D}$ may contain tens of thousands of data points, and if each local site sends about 15 local cluster centers and there are 3 participating sites, then at the coordinating site approximately thousand globalized cluster centers are created. These global cluster centers can now be used to find the final k, say 10, cluster centers for $\mathcal{D}$.

Analytical Justification: The expression for total clustering error is stated earlier in Equation (3). By triangle law of inequality, the distance from a point p in $\mathcal{D}$ to its final cluster center c_i will be less than the sum of the distances from that point p to its globalized cluster center q and the distance from the globalized cluster center q to the final cluster center c. That is:

$$\begin{aligned} \mathit{distance}(p_{ij}, c_j) &\leq \mathit{distance}(p_{il}, q_{lj}) + \mathit{distance}(q_{lj}, c_j), \\ \mathit{TotalError} &\leq \sum_i (\mathit{distance}(p_{il}, q_{lj}) + \mathit{distance}(q_{lj}, c_j), \end{aligned} \tag{5}$$

where p_{il} is the i^{th} data point from $\mathcal{D}$ and belongs to globalized cluster center q_{lj}, q_{lj} belongs to the j^{th} final cluster and c_j is the center of the final j^{th} cluster. The total error can be minimized by individually minimizing both the quantities.

The first quantity in Equation (5) is less than $\sum_{i,j} R_i * p_{ij}$, where p_{ij} is the j^{th} point counted in the i^{th} globalized cluster center and R_i is its radius. This is the worst case scenario, when all the points are on the periphery of the hypersphere. However, minimizing this quantity requires that we use the smallest feasible value for the radius while computing the population around each q_i. At the same time we need to have a large enough radius to include all the points of $\mathcal{D}$ around the members of $\mathcal{Q}$.

The second quantity in Equation (5), GlobalizingError, is equal to $\sum_i \mathit{distance}(q_{lj}, c_j)$ where p_i is counted in globalized cluster center l and the c_j is the final cluster center to which l is assigned. Note that the error is summed over all the points in $\mathcal{D}$.

If a_l is the number of points in the hypersphere around the l^{th} globalized cluster center, then

$$\text{GlobalizingError} = \sum_l a_l * \mathit{distance}(q_{lj}, c_j).$$

If we take the distance function as Euclidean, then

$$\text{GlobalizingError} = \sum_l a_l (q_{lj} - c_j)^2.$$

Differentiating GlobalizingError with respect to a cluster center c_j, we get $\sum_l -2a_l (q_{lj} - c_j)$. For minimum error, this quantity should be equal to 0. Therefore,

$$\begin{aligned} &\Rightarrow \sum_l -2a_l(q_{lj} - c_j) = 0 \\ &\Rightarrow \sum_l a_l(q_{lj} - c_j) = 0 \\ &\Rightarrow \sum_l a_l q_{lj} - \sum_l a_l c_j = 0 \\ &\Rightarrow c_j = \frac{\sum_l a_l q_{lj}}{\sum_l a_l}. \end{aligned} \tag{6}$$

Equation (6) implies that the final cluster center should be the mean of the globalized cluster centers included in a cluster, weighted by the population of points around each globalized cluster center. This modified rule for finding cluster center is used, and iterations of regular k-means algorithm are applied to arrive at the final cluster centers. This part is performed entirely at the coordinating site and does not require any communication with the local sites.

In a distributed environment we cannot compute the clustering error for $\mathcal{D}$; therefore we define its estimate, called *Estimated Clustering Error* (ECE), as an estimate of the total distance between a data point p of $\mathcal{D}$ and its final cluster center c (to be found in this phase of the algorithm), weighted by the population associated with c. Step 2.ii of the algorithm below shows the quantitative definition of the estimated clustering error (ECE). The coordinating site now runs a modified version of the k-means algorithm on the globalized cluster centers in set $\mathcal{Q}$ as follows:

1. Randomly choose a set C of k points as the initial candidates for *final cluster centers* c_is.
2. For each data point $q_i \in \mathcal{Q}$ having a coordinate $(x_{i1}, x_{i2}, \cdots, x_{im})$ do
   (a) For each *final cluster* j having a coordinate $(c_{j1}, c_{j2}, \cdots, c_{jm})$ do
      i Compute the weighted mid-point t_j between a final cluster center c_j and a point q_i, weighted by populations around q_is, as:

$$\begin{aligned} t_j \;=\; &((a_i * x_{i1} + b_j * c_{j1})/(a_i + b_j), (a_i * x_{i2} + b_j * c_{j2})/(a_i + b_j), \ldots, \\ &(a_i * x_{im} + b_j * c_{jm})/(a_i + b_j)), \end{aligned}$$

where a_i is the population in the hypersphere around point q_i in $\mathcal{D}$, and b_j is the population of the cluster at c_j, and

$$b_j = \sum_{q \in cluster(j)} N_q. \tag{7}$$

ii Compute the estimated contribution to the total clustering error if the globalized data point q_i is added to the *final cluster* j as:

$$ECE = distance(center(j), t_j) * b_j + distance(p_i, t_j) * a_i. \tag{8}$$

iii Include globalized data point q_i in that final cluster which results in minimum value for the estimated clustering error.

The difference between the above and the traditional clustering algorithm lies primarily in the way the membership of a point in a potential cluster is decided. Here we make this decision to minimize the potential contribution a point would make to the total clustering error when examined for its inclusion in all possible candidate clusters. This decision is weighted by the population associated with each point, which is actually a cluster center from a local database. The next phase is to recompute the final cluster centers, as per the iterations of the k-means algorithm. This part is also performed entirely at the coordinating site and does not require any communication with the local sites.

3.6 Recompute Cluster Centers

For each final cluster j:

1. Find the total population of the cluster. This is the same as b_j described above.
2. Find all the points P_i that belong to the cluster j.
3. Recompute the new center of cluster j. In this phase we have a deviation from the traditional k-means algorithm. We compute the new centers such that each point is weighted by the population of data points in $\mathcal{D}$ that is associated with it using Equation (6) above.

The last two of the above steps are repeated until the decrease in the estimated cluster error remains below a threshold for a number of iterations. The population weight related adaptations in the previous two steps can be shown to result in clusters that actually minimize the estimated cluster error.

4 COMPLEXITY ANALYSIS AND PRIVACY CONSIDERATION

The cost of working with implicitly specified set of tuples can be measured in various ways. One cost model computes the number of messages that must be exchanged among various sites. Complexity for distributed query processing in databases has

been discussed in [23] and this cost model measures the total data transferred for answering a query. In our case the amount of data transferred is very little (statistical summaries) but the number of messages to be exchanged may grow rapidly with the number of iterations for the clustering algorithm. We derive below an expression for the number of messages that need to be exchanged for our clustering algorithm dealing with the implicit set of tuples. Let us say:

1. There are n relations, $D_1 \ldots D_n$, residing at n different network sites.
2. There are r tuples in relation *PreShared*.
3. There are l tuples in relation *Shared*.
4. The number of globalized cluster centers in $\mathcal{Q}$ is p.

4.1 Complexity Analysis

According to our cost models in [16, 17, 18, 19, 20, 21, 22], we count the number of local counts, N_m, that must be exchanged among all the participating sites.

Stationary Agent Case: We give below an expression for the number of messages that need to be exchanged for dealing with the implicit tuples.

Cost Model 1 (Unoptimized): In the worst case, the number of messages needed will be the sum of the number of messages required to compute the relation *Shared*, perform local clustering, and compute the population around a globalized cluster center (N_q). Thus, the total number of exchanged messages will be:

$$\text{Total Exchanged Messages} = n(2 + r + p * l), \tag{9}$$

where we have n messages to get the different values of the shared attributes from local sites, $n * r$ messages to find relation *Shared* from the *PreShared* relation, n messages to perform local clustering, and $p * n * l$ messages to compute the population around globalized cluster centers.

Cost Model 2 (Optimized): In this cost model, all tuples of *Shared* will be sent in one request and then receive the summaries in one message. This reduces the number of messages exchanged to $4n$, the same as 4 times the number of participating sites. Thus, the total number of exchanged messages will be:

$$\text{Total Exchanged Messages} = 4n. \tag{10}$$

When computing the population around a globalized cluster center, we can also drop certain variables with too many possible values from consideration. The rationale is the same as adopted in decision tree induction algorithms [21] where we do not select an attribute with very high branching factor, such as the SS#, as a node in the decision tree. Such attributes do not contribute much to the information gain and are not to be preferred for *generalization* desired in

a learning algorithm. An example of the worst type of shared variable with too many possible values is the social security number. This may still be a smaller set of data to be transferred compared to the complete database transfer and Join; but this approach is more suitable in all those domains where shared variables are not very much fragmented. There are many such situations in inventory databases and GIS data mining situations.

Mobile Agent Case: This agent has the relation *Shareds* stored in it. During a visit to a data site, it computes all local computations for that site. The local results for computing all the counts can be gathered during three visits to a site. Thus, the mobile agent can compute all requirements by visiting each site three times ($3n$ hops) and then aggregating the local results.

The above analysis of complexity shows that the number of messages that need to be exchanged among the sites is not dependent on the size of the database at each site. The communication complexity, in the case of vertically distributed data, is dependent primarily on the number and manner in which the attributes are shared among the participating sites. This is significant because it shows that as the sizes of the individual databases grow, the communication complexity of the algorithm would remain unaffected. Computational cost of local computations would grow with the database size at each individual site but our decomposable version has an advantage in this regard also over the *transport*, *join*, and then *cluster* alternative. If a k-means algorithm runs q iterations for finding k cluster centers and it has m data points then it must compute $q * k * m$ distances.

If each local database D_i has m tuples, then in the worst case the join of n local databases would produce a relation containing order of m^n tuples. There is additional cost of order of $(n * m)$ comparisons for creating the *Join*. When the k-means algorithm is run with this explicitly created $\mathcal{D}$, we would need to compute $q * k * m^n$ distances. In our decomposable version, each of the n sites would be computing only $q * k * m$ distances. Thus, there is tremendous saving in the computational cost when the decomposable version is executed instead of moving the data, creating a *Join* and then running the clustering algorithm. Also, for the communication cost, the number of partial results that need to be transmitted is far fewer that the messages that may have to be transmitted if entire databases are collected at some central site.

Another important gain of decomposable version is that it preserves the privacy of the data by not requiring any data tuples to be placed on a communication network. It also preserves the integrity of individual databases because no site needs to update or write into any of the participating databases. All the queries are strictly reading queries.

4.2 Privacy and Security Considerations

We have demonstrated above that k-means algorithm can be very closely approximated for distributed databases without having to move the databases to a cen-

tralized site. From the point of view of data security and privacy the following observations can be made:

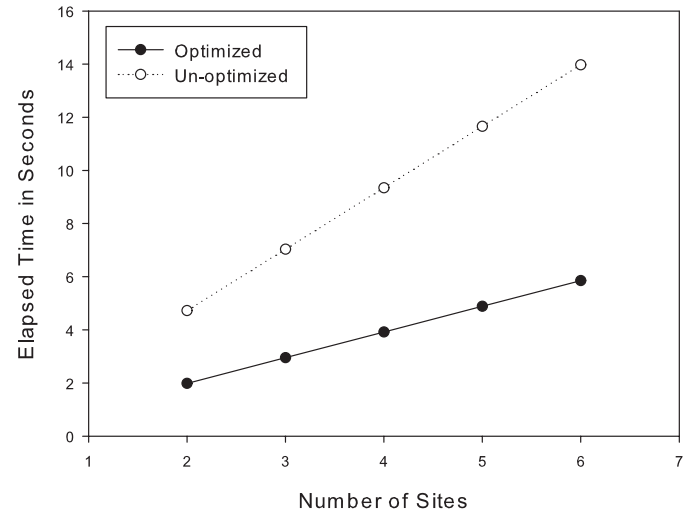
1. No data tuple is exchanged between the sites.
2. Initial cluster centers from local sites are transmitted to the central site. Global cluster centers are maintained within the coordinating site and never transmitted.
3. When computing the population around global cluster centers, the coordinating site sends only the locally relevant attributes to each site and distances from the attribute-value pairs of the local data are returned only for a subset of attributes contained in the global cluster centers.

If the information security and privacy is defined by not having to release any data tuple out of a database for transmission over the network and the reconstruction of any data tuple being impossible by the released data summaries then the above algorithm preserves the privacy of the data in each participating database. No data tuple is ever transmitted and the summaries are not sufficient to reconstruct any individual data tuple. The only loophole would be when a cluster of one data point is formed and its contents are released for communication to other sites in the form of the cluster center of this cluster. This can be easily avoided by setting a minimum threshold t (at least $t = 2$) so that any time a local site sends out a summary it must be for at least t tuples. The tradeoff would be that the algorithm would not be able to form clusters that are smaller than t in size.

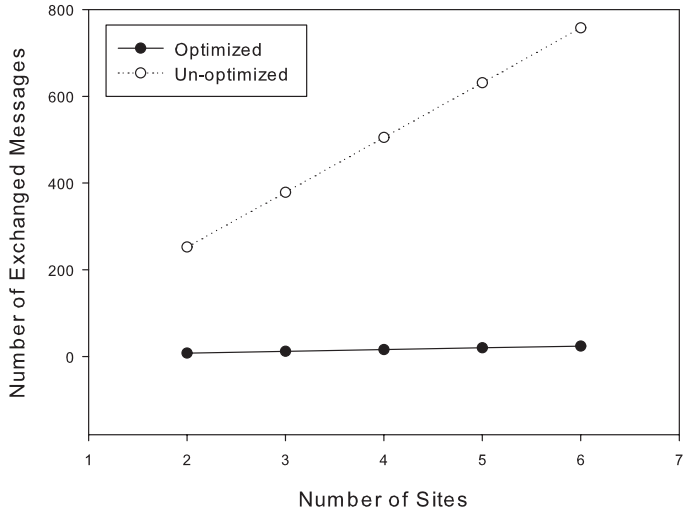
5 SIMULATION RESULTS

In order to show the advantages of our algorithm, we have performed a number of tests to demonstrate that the proposed k-means clustering algorithm can work correctly in a distributed knowledge environment without moving all the databases to a single site. The tests were performed to find out the effect of various parameters on the final result. Three very important variables that affect the result are: the number of tuples per database, the number of sites, and the average number of shared tuples between local databases. These tests have been carried out on a network of workstations connected by a LAN and tested against a number of databases of different sizes.

The first test was done to demonstrate how the elapsed time and the number of exchanged messages varies with the number of local sites. The number of local sites varies as 2, 3, 4, 5, and 6. Figure 2 a) shows how the elapsed time to compute the proposed k-means clustering algorithm in an implicit database $\mathcal{D}$ changes with the number of local sites. It can be seen easily that the elapsed time to compute the proposed k-means clustering algorithm increases as the number of local sites increases. Figure 2 b) shows how the number of exchanged messages between the local sites changes with the number of local sites. It can be seen easily that the number of exchanged messages increases as the number of local sites increases.



a)



b)

Figure 2. Results of running the proposed k-means on vertically distributed databases when the number of local sites is varied: a) elapsed time, b) the number of exchanged messages

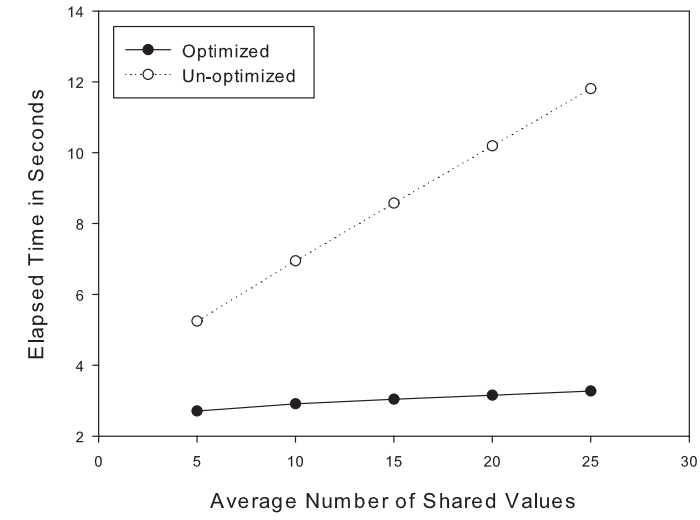
The second test was done to demonstrate how the elapsed time and the number of exchanged messages varies with the average number of shared tuples between local databases. The number of shared values varies as 5, 10, 15, 20, and 25. Figure 3 a) shows how the elapsed time to compute the proposed k-means clustering algorithm in an implicit database $\mathcal{D}$ changes with the average number of shared tuples between local databases. It can be seen easily that the elapsed time to compute the proposed k-means clustering algorithm increases as the number of shared values increases. Figure 3 b) shows how the number of exchanged messages between the local sites changes with the number of shared values. It can be seen easily that the number of exchanged messages increases as the number of shared values increases.

The third test was done to demonstrate how the elapsed time and the number of exchanged messages varies with the number of tuples in the database. Figure 4 a) shows how the elapsed time to compute the proposed k-means clustering algorithm in an implicit database $\mathcal{D}$ changes with the number of tuples in the database. Figure 4 b) shows how the number of exchanged messages between the local sites changes with the number of tuples in the database. The results show that the performance of our proposed k-means clustering algorithm continues to perform the best results.

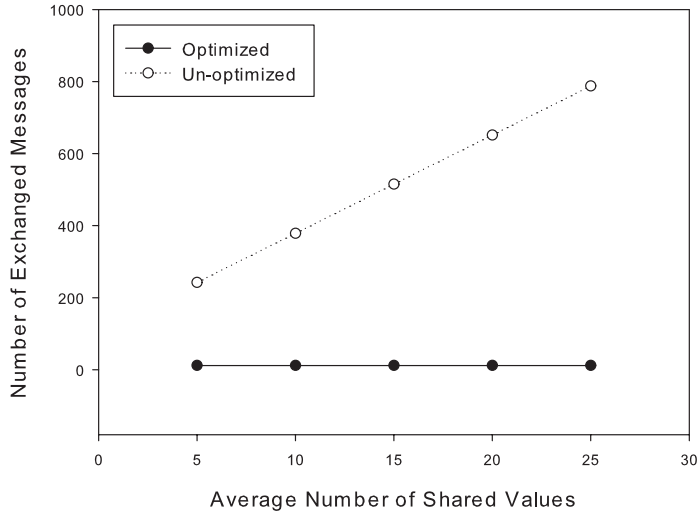
Finally, we ran the tests by varying the number of clusters for a $\mathcal{D}$ that had in excess of 500 points and were distributed similarly into two projections and measured the *clustering error* for each case. The plots in Figure 5 show the total clustering error for cluster centers determined by our decomposable algorithm and by direct application of a k-means algorithm on an explicitly created $\mathcal{D}$. It can be seen that the difference between the error quantities is very small and follows the same pattern in two tests. The plots also show the estimated clustering error that is computed by the coordinating site to guide itself towards the final cluster centers. This quantity reduces faster than the actual clustering error but follows the same trend, and thus can guide towards the minimum error cluster centers.

6 CONCLUSION

In this paper we have presented a decomposable version of k-means clustering algorithm for vertically and horizontally distributed datasets that are geographically distributed. We have also presented the analytical basis for the design of our algorithm. The algorithm succeeds in obtaining results very close to those that would be achieved by moving all the databases to one site, joining them, and then executing the *k-means* algorithm. Our distributed version of the algorithm succeeds in doing so by minimizing the total clustering error, a characteristic property of the k-means algorithm. We use the information about the clusters formed at local sites to determine the approximate locations of the possible global cluster centers. Information about the centers and an algorithm to count populations of points around cluster centers in an implicitly specified relation are used by the central coordinating site to minimize a close estimate of the total clustering error. We have demonstrated

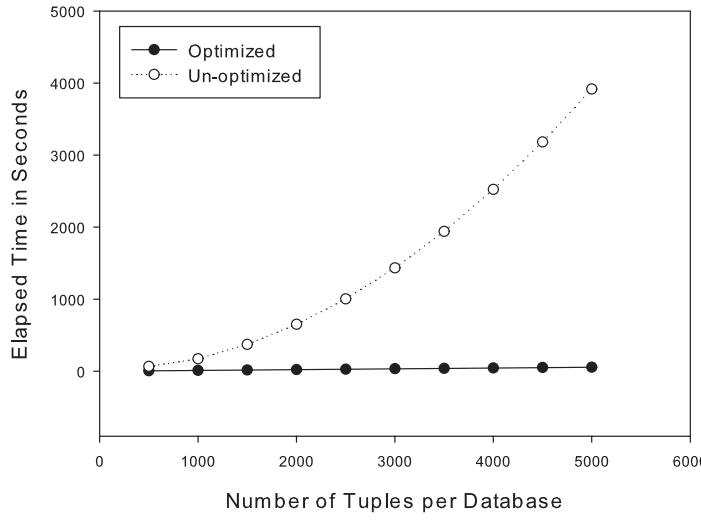


a)

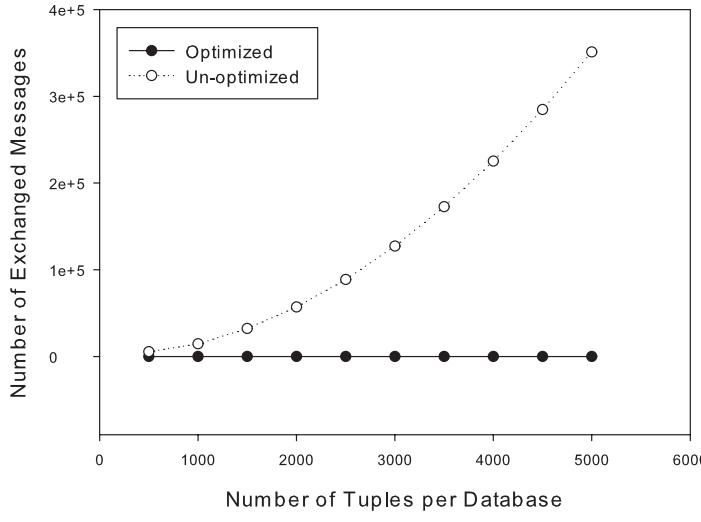


b)

Figure 3. Results of running the proposed k-means on vertically distributed databases when the average number of shared values is varied: a) elapsed time, b) the number of exchanged messages



a)



b)

Figure 4. Results of running the proposed k-means on vertically distributed databases when the number of tuples is varied: a) elapsed time, b) the number of exchanged messages

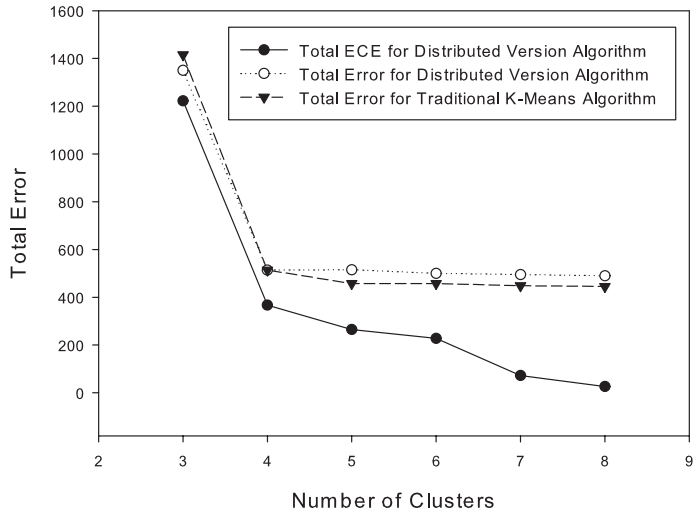


Figure 5. Total error versus the number of clusters in distributed and traditional algorithms

that the convergence of the above version and of the original k-means algorithm is to centers that are very closely placed; signified by a very small difference in the total clustering error. Our version achieves these very close results at a great saving in the total communication cost and also preserves the privacy and integrity of the individual databases.

REFERENCES

[1] Anderberg, M. R.: Cluster Analysis for Applications. Academic Press, New York, 1973.

[2] Dhillon, I. S.—Modha, D. S.: A Data Clustering Algorithm on Distributed Memory Multiprocessors. In Large-Scale Parallel Data Mining, Lecture Notes in Artificial Intelligence, Vol. 1759, Springer-Verlag 2000, pp. 245–260.

[3] Jain, A. K.—Dubes, R. C.: Algorithms for Clustering Data. Prentice-Hall International 1988.

[4] Young, T. Y.—Fu, K. S.: Handbook of Pattern Recognition and Image Processing. Academic Press 1986.

[5] Jagannathan, G.—Pillaipakkamnatt, K.—Wright, N. R.: A New Privacy-Preserving Distributed k-Means Clustering Algorithm. In Proceedings of the 2006 SIAM International Conference on Data Mining (SDM) 2006.

[6] Vaidya, J.—Clifton, C.: Privacy-Preserving k-Means Clustering over Vertically Partitioned Data. In Proceedings of KDD 2003, pp. 206–215.

[7] Kargupta, H.—Huang, W.—Sivakumar, K.—Johnson, E.: Distributed Clustering Using Collective Principal Component Analysis. Knowledge and Information System, Vol. 3, 2001, No. 4, pp. 422–448.

[8] Ruoming, J.—Anjan, G.—Agrawal, G.: Fast and Exact Out-of-Core and Distributed k-Means Clustering. Knowledge and Information Systems, Vol. 10, 2006, No. 1.

[9] Chan, P.—Stolfo, S.: On the Accuracy of Meta-Learning for Scalable Data Mining. J. Intelligent Information Systems, Vol. 8, 1997, pp. 5–28.

[10] Crestana, V.—Soparkar, N.: Mining Decentralized Data Repositories. Technical Report CSE-TR-385-399, Ann Arbor, MI, 1999.

[11] Grossman, R.—Bailey, S.—Ramu, A.—Malhi, B.—Turinski, A.: The Preliminary Design of Papyrus: A System for High Performance, Distributed Data Mining over Clusters. In Advances in Distributed and Parallel Knowledge Discovery, AAAI Press/MIT Press 2000, pp. 259–275.

[12] Lam, W.—Segre, A. M.: Distributed Data Mining for Probabilistic Knowledge. In Proceedings of the 17$^{\text{th}}$ International Conference on Distributed Computing Systems, IEEE Computer Society Press, Washington 1997, pp. 178–185.

[13] Park, H.—Ayyagari, R.—Kargupta, H.: A Fourier Analysis-Based Approach to Learn Classifiers from Distributed Heterogeneous Data. In Proceedings of the First SIAM International Conference on Data Mining 2001.

[14] Tummer, K.—Ghosh, J.: Robust Order Statistics Based Ensembles for Distributed Data Mining: In Advances in Distributed and Parallel Knowledge Discovery. MIT Press, Cambridge, MA 1999.

[15] Yamanishi, K.: Distributed Cooperative Bayesian Learning Strategies. In Proceedings of COLT '97, ACM, New York 1997, pp. 250–262.

[16] Khedr, A. M.: Decomposable Naive Bayes Classifier for Partitioned Data. Computing and Informatics, Vol. 31, 2012, No. 6+, pp. 1511–1531.

[17] Khedr, A. M.—Rania, M.: Agents for Integrating Distributed Data for Function Computations. Computing and Informatics, Vol. 31, 2012, No. 5, pp. 1101–1125.

[18] Khedr, A. M.: Nearest Neighbor Clustering over Partitioned Data. Computing and Informatics, Vol. 30, 2011, No. 5, pp. 1011–1036.

[19] Bhatnagar, R.—Young, B.: Computations in Distributed Databases. In Proceedings of the ADCOM '99 Conference, Roorkee, India 1999, pp. 32–38.

[20] Khedr, A. M.—Bhatnagar, R. K.: Agents for Integrating Distributed Data for Complex Computations. Computing and Informatics, Vol. 26, 2007, No. 2, pp. 149–170.

[21] Khedr, A. M.: Learning k-Classifier from Distributed Databases. Computing and Informatics, Vol. 27, 2008, No. 3, pp. 355–376.

[22] Khedr, A. M.—Salim, A.: Decomposable Algorithms for Finding the Nearest Pair. J. Parallel Distrib. Comput., Vol. 68, 2008, pp. 902–912.

[23] Wang, C.—Chen, M.: On the Complexity of Distributed Query Optimization. IEEE Transactions on Knowledge and Data Engineering, Vol. 8, 1996, No. 4, pp. 650–662.

[24] Theodoridis, S.—Kouteroumbas, K.: Pattern Recognition. Academic Press, New York 1999.

Ahmed M. Khedr received his B. Sc. degree in Mathematics in June 1989 and the M. Sc. degree in optimal control in July 1995, both from Zagazig University, Egypt. In July 1999 he received his M. Sc. and in March 2003 he received his Ph. D. degree, both in Computer Science and Engineering, from University of Cincinnati, Ohio, USA. From March 2003 to January 2004, he was a Research Assistant Professor at ECECS Department, University of Cincinnati, USA. From January 2004 to May 2009, he worked As Assistant Professor at Zagazig University, Egypt; from September 2009 to September 2010 he worked as Associate Professor at the Department of Computer Science, College of Computers and Information Systems, Taif University, KSA, and from September 2010 till now he is working as Associate Professor at the Department of Computer Sciences, College of Science, Sharjah University, UAE. In June 2009, he was awarded the State Prize of Distinction in Advanced Technology. In May 2013, he was awarded Sharjah Islamic Bank Prize for distinguished research. He has co-authored 46 works in journals and conferences relating to optimal control, wireless sensor networks, distributed computing, and bioinformatics.

Raj K. Bhatnagar is a Professor of computer science at University of Cincinnati. His main research interests are in data mining and pattern recognition algorithms. His research has been supported by various federal, state, and industry organizations. Over the years he has applied the results of his research to radar signatures, geographical data, and genomic data.