Distributed Data Mining and Agents

Josenildo C. da Silva[2], Chris Giannella[1], Ruchita Bhargava[3], Hillol Kargupta[1,4], and Matthias Klusch[2]

[1] Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County,
Baltimore, MD 21250 USA
{cgiannel,hillol}@cs.umbc.edu
[2] German Research Center for Artificial Intelligence
Stuhlsatzenweghaus 3, 66121 Saarbruecken, Germany
{jcsilva,klusch}@dfki.de
[3] Microsoft Corporation
One Microsoft Way
Redmond, WA 98052 USA
[4] AGNIK LLC
8840 Stanford Blvd. Suite 1300
Columbia, Maryland 21045 USA

Abstract. Multi-Agent Systems (MAS) offer an architecture for distributed problem solving. Distributed Data Mining (DDM) algorithms focus on one class of such distributed problem solving tasks—analysis and modeling of distributed data. This paper offers a perspective on DDM algorithms in the context of multi-agents systems. It discusses broadly the connection between DDM and MAS. It provides a high-level survey of DDM, then focuses on distributed clustering algorithms and some potential applications in multi-agent-based problem solving scenarios. It reviews algorithms for distributed clustering, including privacy-preserving ones. It describes challenges for clustering in sensor-network environments, potential shortcomings of the current algorithms, and future work accordingly. It also discusses confidentiality (privacy preservation) and presents a new algorithm for privacy-preserving density-based clustering.

Keywords: multi-agent systems, distributed data mining, clustering, privacy, sensor networks

1 Introduction

Multi-agent systems (MAS) often deal with complex applications that require distributed problem solving. In many applications the individual and collective behavior of the agents depend on the observed data from distributed sources. In a typical distributed environment analyzing distributed data is a non-trivial problem because of many constraints such as limited bandwidth (*e.g.* wireless networks), privacy-sensitive data, distributed compute nodes, only to mention a few. The field of Distributed Data Mining (DDM) deals with these challenges in analyzing distributed data and offers many algorithmic solutions to perform different data analysis and mining operations in a fundamentally distributed manner that pays careful attention to the resource constraints.

Since MAS are also distributed systems, combining DDM with MAS for data intensive applications is appealing.

This paper underscores the possible synergy between MAS and DDM technology. It particularly focuses on distributed clustering, a problem finding increasing number of applications in sensor networks, distributed information retrieval, and many other domains. The paper provides a detailed literature review of existing clustering algorithms in DDM (including privacy-preserving ones). Then, it discusses one application domain, sensor networks, underscoring some challenges, potential shortcomings of the current algorithms, and future work accordingly. Finally, this paper discusses privacy (confidentiality) and presents a new algorithm for privacy-preserving clustering.

The paper is organized as follows. Section 2 provides the motivation behind the material presented in this paper. Section 3 introduces DDM and presents an overview of the field. Section 4 focuses on a particular portion of the DDM literature and takes an in-depth look at the distributed clustering literature. Section 5 considers distributed clustering algorithms in the context of sensor networks which are drawing an increasing amount of interest from the multi-agent systems community. Sections 6 and 7 discuss the issue of privacy (confidentiality) and present a new algorithm for privacy-preserving density-based clustering. Finally, Section 9 concludes the paper.

2 Motivation

Agents in MAS need to be pro-active and autonomous. Agents perceive their environment, dynamically reason out actions based on conditions, and interact with each other. In some applications the knowledge of the agents that guide reasoning and action depend on the existing domain theory. However, in many complex domains this knowledge is a result of the outcome of empirical data analysis in addition to pre-existing domain knowledge. Scalable analysis of data may require advanced data mining for detecting hidden patterns, constructing predictive models, and identifying outliers, among others. In a multi-agent system this knowledge is usually collective. This collective "intelligence" of a multi-agent system must be developed by distributed domain knowledge and analysis of distributed data observed by different agents. Such distributed data analysis may be a non-trivial problem when the underlying task is not completely decomposable and computing resources are constrained by several factors such as limited power supply, poor bandwidth connection, and privacy sensitive multi-party data, among others.

For example, consider a defense related application of monitoring a terrain using a sensor network that has many tiny mote-type [28] sensors for measuring vibration, reflectance, temperature, and audio signals. Let us say the objective is to identify and track a certain type of vehicle (*e.g.* pick-up trucks). The sensors are battery-powered. Therefore, in the normal mode they are designed not be very active. However, as soon as someone detects a possible change in scenario, the sensors must wake up, observe, reason, and collaborate with each other in order to track and identify the object of interest. The observations are usually time-series data sampled at a device specific rate. Therefore, collaboration with other sensor-nodes would require comparing data observed at different nodes. This usually requires sending a window of observations from one node to another node. This distributed problem solving environment appears to fit very well with the multi-agent framework since the solution requires semi-autonomous behavior, collaboration and reasoning among other things. However, regardless of how sophisticated the agents are, from the domain knowledge and reasoning perspective, they must perform the underlying data analysis tasks very efficiently in a distributed manner. The traditional framework of centralized data analysis and mining algorithms does not really scale very well in such distributed applications. For example, if we want to compare the data vectors observed at different sensor nodes the centralized approach will be to send the data vectors to the base station (usually connected through a wireless network) and then compare the vectors using whatever metric is appropriate for the domain. This does not scale up in large sensor networks since data transmission consumes a lot of battery power and heavy data transmission over limited bandwidth channel may produce poor response time. Distributed data mining technology offers more efficient solutions in such applications. The following discussion illustrates the power of DDM algorithms using a simple randomized technique for addressing this sensor network-related problem.

Given vectors $\boldsymbol{a} = (a_1, \ldots, a_m)^T$ and $\boldsymbol{b} = (b_1, \ldots, b_m)^T$ at two distributed sites A and B, respectively, we want to approximate the Euclidean distance between them using a small number (compared to m) of messages between A and B. Note that the problem of computing the Euclidean distance between a pair of data tuples $\boldsymbol{a}$ and $\boldsymbol{b}$ can be represented as the problem of computing the inner products between them as follows:

$$d_e^2(\boldsymbol{a}, \boldsymbol{b}) = <\boldsymbol{a}, \boldsymbol{a}> + <\boldsymbol{b}, \boldsymbol{b}> - 2 <\boldsymbol{a}, \boldsymbol{b}>$$

where $d_e^2(\boldsymbol{a}, \boldsymbol{b})$ denotes the Euclidean distance between $\boldsymbol{a}$ and $\boldsymbol{b}$; $<\boldsymbol{a}, \boldsymbol{b}>$ represents the inner product between $\boldsymbol{a}$ and $\boldsymbol{b}$, defined as $\sum_{i=1}^{m} a_i b_i$. Therefore, the core challenge is to develop an algorithm for distributed inner product computation. One can approach this problem in several ways. Algorithm 2.0.1 is a simple, communication-efficient randomized technique for computing the inner product between two vectors observed at two different sites.

Algorithm 2.0.1 Distributed Dot Product Algorithm($\boldsymbol{a}, \boldsymbol{b}$)

```
1: A sends B a random number generator seed. [1 message]
2: A and B cooperatively generate $k \times m$ random matrix $R$ where $k \ll m$. Each entry is
       generated independently and identically from some fixed distribution with mean zero
       and finite variance. A and B compute $\hat{a} = R\boldsymbol{a}$, $\hat{b} = R\boldsymbol{b}$, respectively.
3: A sends $\hat{a}$ to B. [k messages]
4: B computes $D = \frac{\hat{a}^T \hat{b}}{k}$.
```

So instead of sending a m-dimensional vector to the other site, we only need to send a k-dimensional vector where $k \ll m$ (a user-defined parameter) and the dot product can still be estimated accurately. Indeed, it can be shown that the expected value of D is $<\boldsymbol{a}, \boldsymbol{b}>$ and Figure 1 shows some experimental results concerning accuracy.

This algorithm illustrates a simple communication-efficient way to compare a pair of data vectors observed at two different nodes. It potentially offers a building block to support the collaborative object identification and tracking problem in sensor networks where the centralized solution does not work because of limited bandwidth and power supply for the sensor nodes.

k	Mean	Var	Min	Max
100(1%)	0.1483	0.0098	0.0042	0.3837
500(5%)	0.0795	0.0035	0.0067	0.2686
1000(10%)	0.0430	0.0008	0.0033	0.1357
2000(20%)	0.0299	0.0007	0.0012	0.0902
3000(30%)	0.0262	0.0005	0.0002	0.0732

Fig. 1. Relative errors in computing the dot product between two synthetic binary vectors each with 10000 elements. k is the number of randomized iterations. k is also represented as the percentage of the size of the original vectors. Each entry of the random matrix is chosen independently from U(1,-1).

Privacy of the data can be another reason for adopting the DDM technology. In many applications, particularly in security-related applications, data is privacy-sensitive (confidential). As such, centralizing the distributed data sets is not acceptable. Therefore, data mining applications in such domains must analyze data in a distributed fashion without having to first down-load everything to a single site. Furthermore, these applications must pay careful attention to the amount and type of information revealed to each site about the other sites' data. In some cases, strict privacy requirements must be met, namely, no information regarding other sites' data can be obtained beyond that of the analysis output *e.g.* the decision tree learned. *Privacy preserving data mining* has emerged in the last five years to address these needs.

3 Distributed Data Mining: A Brief Overview

Data mining [20], [21], [22],and [61] deals with the problem of analyzing data in scalable manner. DDM is a branch of the field of data mining that offers a framework to mine distributed data paying careful attention to the distributed data and computing resources.

In the DDM literature, one of two assumptions is commonly adopted as to how data is distributed across sites: homogeneously (horizontally partitioned) and heterogeneously (vertically partitioned). Both viewpoints adopt the conceptual viewpoint that the data tables at each site are partitions of a single global table. In the homogeneous case, the global table is horizontally partitioned. The tables at each site are subsets of the global table; they have exactly the same attributes. In the heterogeneous case the table is vertically partitioned, each site contains a collection of columns (sites do not have the same attributes). However, each tuple at each site is assumed to contain a unique identifier to facilitate matching. It is important to stress that the global table viewpoint is strictly conceptual. It is not necessarily assumed that such a table was physically realized and partitioned to form the tables at each site. Figures 2 and 3 illustrate the homogeneously distributed case using an example from weather data. Both tables use the same set of attributes. On the other hand, Figures 4 and 5 illustrate the heterogeneously distributed case. The tables have different attributes and tuples are linked through a unique identifier, *Timestamp*.

The development of data mining algorithms that work well under the constraints imposed by distributed datasets has received significant attention from the data mining community in recent years. The field of DDM has emerged as an active area of study. The bulk of DDM methods in the literature operate over an abstract architecture which includes multiple sites having independent computing power and storage capability. Local computation is done on each of the sites and either a central site communicates with each distributed site to compute the global models or a peer-to-peer architecture is used. In the latter case, individual nodes might communicate with a resource rich centralized node, but they perform most of the tasks by communicating with neighboring nodes by message passing over an asynchronous network. For example, the sites may represent independent sensor nodes which connect to each other in an ad-hoc fashion.

Some features of a distributed scenario where DDM is applicable are as follows.

1. The system consist of multiple independent sites of data and computation which communicate only through message passing.
2. Communication between the sites is expensive.
3. Sites have resource constraints *e.g.* battery power.
4. Sites have privacy concerns.

Typically communication is a bottleneck. Since communication is assumed to be carried out exclusively by message passing, a primary goal of many DDM methods in the literature is to minimize the number of messages sent. Some methods also attempt to load-balance across sites to prevent performance from being dominated by the time and space usage of any individual site. As pointed out in [47], "Building a monolithic database, in order to perform non-distributed data mining, may be infeasible or simply impossible" in many applications. The cost of transferring large blocks of data may be prohibitive and result in very inefficient implementations.

Surveys [31] and [45] provide a broad, up-to-date overview of DDM touching on issues such as: clustering, association rule mining, basic statistics computation, Bayesian network learning, classification, the historical roots of DDM. The collection [30] describes a variety of DDM algorithms (association rule mining, clustering, classification, preprocessing, *etc.*), systems issues in DDM (security, architecture, *etc.*), and some topics in parallel data mining. Survey [63] discusses parallel and distributed association rule mining in DDM. Survey [64] discusses a broad spectrum of issues in DDM and parallel data mining and provides a survey of distributed and parallel association rule mining and clustering. Many of the DDM applications [52, 32] deal with continuous data streams. Therefore, developing DDM algorithms that can handle such stream scenarios is becoming increasingly important. An overview of the data stream mining literature can be found elsewhere [4].

City	Humidity	Temperature	Rainfall
Baltimore	10%	$23°$ F	0 in.
Annapolis	13%	$43°$ F	0.2 in.
Bethesda	56%	$67°$ F	1 in.
Glen Burnie	88%	$88°$ F	1.2 in.

Fig. 2. Homogeneously distributed weather data at site 1

City	Humidity	Temperature	Rainfall
San Jose	12%	$69°$ F	0.3 in.
Sacramento	18%	$53°$ F	0.5 in.
Los Angeles	86%	$72°$ F	1.2 in.
San Diego	8%	$58°$ F	0 in.

Fig. 3. Homogeneously distributed weather data at site 2

Timestamp	Humidity	Temperature	Rainfall
t_0	10%	$23°$ F	0 in.
t_1	13%	$43°$ F	0.2 in.
t_2	56%	$67°$ F	1 in.
t_3	88%	$88°$ F	1.2 in.

Fig. 4. Heterogeneously distributed weather data

Timestamp	Body Temp.	Heart Rate
t_0	$98.5°$ F	60 bpm
t_1	$98.8°$ F	70bpm
t_2	$99°$ F	75 bpm
t_3	$99.3°$ F	80 bpm

Fig. 5. Heterogeneously distributed health data

3.1 Privacy Preserving Data Mining

Privacy plays an important role in DDM as some participants may wish to not share their data, but still participate in DDM. Next we briefly review PPDM providing a high level overview of the field. For detailed literature survey of PPDM, we refer the reader to [60]. Moreover, in Section 4, we provide a detailed survey of privacy-preserving distributed clustering.

One of the earliest discussions about privacy in the context of data mining can be found in [7]. Most recent efforts addressing the privacy issue in data mining include the *sanitation*, *data distortion* approaches and *cryptographic methods*.

Sanitation: These approaches aim to modify the dataset so that sensitive patterns cannot be mined. They were developed primarily to handle privacy in association rule mining. Techniques were developed by Atallah *et al.* [2] and Dasseni *et al.* [9]. Their basic idea is to remove or modify items in a database to reduce or increase the support of some frequent itemsets. By doing so, the data owner expects to hide some sensitive frequent itemsets with as little as possible impact on other non-sensitive frequent itemsets. Further developments of this technique can be found in [50], [51], [43], [26], and [6].

Data distortion: These approaches (also called data perturbation or data randomization) provides privacy for individual data records through modification of the original data. These techniques aim to design distortion methods after which the true value of any individual record is difficult to ascertain, but "global" properties of the data remain largely unchanged. For example, the use of random noise on market basket data is studied in [48], [14], and [15]. The authors argue that individual transactions are difficult to recover but frequent itemsets remain largely unchanged. Data distortion has also been applied to decision tree based classification [1], among other places. However, one weakness of using data distortion for preserving data privacy is that under certain conditions randomization does not prevent an attacker from reconstructing original data with reasonably high probability [29, 33].

Cryptographic Methods: These apply techniques from cryptography to carry out a data mining task and provably not reveal any information regarding the original data except that which can be inferred from the task output. *Secure Multi-Party Computation (SMC)* offers an assortment of basic tools for allowing multiple parties to jointly compute a function on their inputs while learning nothing except the result of the function. A survey of such tools can be found in [46, 5]. Application of SMC can be found, for example, in [29], [58], [57](association rule mining) and [39] and [11] (decision-tree based data classification). However, the primary weakness of SMC based methods is their high communication and computational costs – a problem made worse by the typically large data sets often encountered in data mining.

3.2 Our Focus: Distributed Clustering

Instead of looking at the broad spectrum of DDM algorithms, we restrict ourselves to distributed clustering methods and their applicability in MAS. In the next section we provide a detailed survey of distributed clustering algorithms that have appeared in the data mining literature. The survey is organized into two broad categories: *efficiency focused* and *privacy focused*. Efficiency focused algorithms strive to increase communication and/or computational efficiency. While in some cases they can offer nice privacy preservation, this is not their primary goal. Privacy focused algorithms hold privacy maintenance as their primary goal. While they also try to maximize communication and computational efficiency, they first preserve privacy.

Section 5 describes an application domain for efficiency focused algorithms, sensor networks with a peer-to-peer communication architecture. It identifies some of the constraints in clustering data in such environments, offers a perspective of the existing distributed clustering algorithms in the context of this particular application, and points out areas that require further research.

Sections 6 and 7 address privacy issues. Unlike the previous section, however, challenges from an application domain are not described. Instead, a new privacy preserving algorithm for density based clustering is presented.

4 Survey of Distributed Clustering Algorithms

We first describe efficiency focused algorithms, then privacy focused. Efficiency focused algorithms are further classified into two sub-categories. The first consists of methods requiring multiple rounds of message passing. These methods require a significant amount synchronization. The second sub-category consists of methods that build local clustering models and transmit them to a central site (asynchronously). The central site forms a combined global model. These methods require only a single round of message passing, hence, modest synchronization requirements.

4.1 Efficiency Focused: Multiple Communication Round

Dhillon and Modha [10] develop a parallel implementation of the K-means clustering algorithm on distributed memory multiprocessors (homogeneously distributed data). The algorithm makes use of the inherent data parallelism in the K-means algorithm. Given a dataset of size n, they divide it into P blocks, (each of size roughly n/P). During each iteration of K-means, each site computes an update of the current K centroids based on its own data. The sites broadcast their centroids. Once a site has received all the centroids from other sites it can form the global centroids by averaging.

Forman and Zhang [19] take an approach similar to the one presented in [10], but extend it to K-harmonic means. Note that the methods of [10] and [19] both start by partitioning and then distributing a centralized data set over many sites. This is different than the setting we consider: the data is never centralized – it is inherently distributed. However, their ideas are useful for designing algorithms to cluster homogeneously distributed data.

Kargupta *et al.* [34] develop a collective principle components analysis (PCA)-based clustering technique for heterogeneously distributed data. Each local site performs PCA, projects the local data along the principle components, and applies a known clustering algorithm. Having obtained these local clusters, each site sends a small set of representative data points to a central site. This site carries out PCA on this collected data (computes global principal components). The global principle components are sent back to the local sites. Each site projects its data along the global principle components and applies its clustering algorithm. A description of locally constructed clusters is sent to the central site which combines the cluster descriptions using different techniques including but not limited to nearest neighbor methods.

Klusch *et al.* [36] consider kernel-density based clustering over homogeneously distributed data. They adopt the definition of a density based cluster from [23] data points which can be connected by an uphill path to a local maxima, with respect to the kernel density function over the whole dataset, are deemed to be in the same cluster. Their algorithm does not find a clustering of the entire dataset. Instead each local site finds a clustering of its *local* data based on the kernel density function computed over *all* the data. In principle, their approach could be extended to produce a global clustering by transmitting the local clusterings to a central site and then combining them. However, carrying out this extension in a communication efficient manner is non-trivial task and is not discussed by Klusch *et al.*

An approximation to the global, kernel density function is computed at each site using sampling theory from signal processing. The sites must first agree upon a cube and a grid (of the cube). Each corner point can be thought of as a sample from the space (not the data set). Then each site computes the value of its local density function at each corner of the grid and transmits the corner points along with their local density values to a central site. The central site computes the sum of all samples at each grid point and transmits the combined sample grid back to each site. The local sites can now independently estimate the global density function over *all* points in the cube (not just the corner points) using techniques from sampling theory in signal processing. The local sites independently apply a gradient-ascent based density clustering algorithm to arrive at a clustering of their local data.

Eisenhardt *et al.* [13] develop a distributed method for document clustering (hence operates on homogeneously distributed data). They extend K-means with a "probe and echo" mechanism for updating cluster centroids. Each synchronization round corresponds to a K-means iteration. Each site carries out the following algorithm at each iteration. One site initiates the process by marking itself as engaged and sending a probe message to all its neighbors. The message also contains the cluster centroids currently maintained at the initiator site. The first time a node receives a probe (from a neighbor site p with centroids C_p), it marks itself as engaged, sends a probe message (along with C_p) to all its neighbors (except the origin of the probe), and updates the centroids in C_p using its local data as well as computing a weight for each centroid based on the number of data points associated with each. If a site receives an echo from a neighbor p (with centroids C_p and weights W_p), it merges C_p and W_p with its current centroids and weights. Once a site has received either a probe or echo from all neighbors, it sends an echo along with its local centroids and weights to the neighbor from which it received its *first* probe. When the initiator has received echos from all its neighbors, it has the centroids and weights which take into account all datasets at all sites. The iteration terminates.

While all algorithms in this section require multiple rounds of message passing, [34] and [36] require only two rounds. The others require as many rounds as the algorithm iterates (potentially many more than two).

4.2 Efficiency Focused: Centralized Ensemble-Based

Many of the distributed clustering algorithms work in an asynchronous manner by first generating the local clusters and then combining those at the central site. These approaches potentially offer two nice properties in addition to lower synchronization requirements. If the local models are much smaller than the local data, their transmission will result is excellent message load requirements. Moreover, sharing only the local models may be a reasonable solution to privacy constraints in some situations; indeed, a trade-off between privacy and communication cost is for one particular algorithm is discussed in Section 4.3.

We present the literature in chronological order. Some of the methods were not explicitly developed for distributed clustering, rather for combining clusterings in a centralized setting to produce a better overall clustering. In these cases we discuss how well they seem to be adaptable to a distributed setting.

Johnson and Kargupta [12] develop a distributed hierarchical clustering algorithm on heterogeneously distributed data. It first generates local cluster models and then combines these into a global model. At each local site, the chosen hierarchical clustering algorithm is applied to generate local dendograms which are then transmitted to a central site. Using statistical bounds, a global dendogram is generated.

Lazarevic *et al.* [38] consider the problem of combining spatial clusterings to produce a global regression-based classifier. They assume homogeneously distributed data and that the clustering produced at each site has the same number of clusters. Each local site computes the convex hull of each cluster and transmits the hulls to a central site along with regression model for each cluster. The central site averages the regression models in overlapping regions of the hulls.

Samatova *et al.* [49] develop a method for merging hierarchical clusterings from homogeneously distributed, real-valued data. Each site produces a dendogram based on local data, then transmits it to a central site. To reduce communication costs,they do not send a complete description of each cluster in a dendogram. Instead an approximation of each cluster is sent consisting of various descriptive statistics *e.g.* number of points in the cluster, average square Euclidean distance from each point in the cluster to the centroid. The central site combines the dendogram descriptions into a global dendogram description.

Strehl and Ghosh [54] develop methods for combining cluster ensembles in a centralized setting. They argue that the best overall clustering maximizes the average normalized mutual information over all clusters in the ensemble. However, they report that finding a good approximation directly is very time-consuming. Instead they develop three more efficient algorithms which are not theoretically shown to maximize mutual information, but are empirically shown to do a decent job. Given n data points and N clusterings (clustering i has k_i clusters), consider an $n \times (\sum_{i=1}^{N} k_i)$ matrix H constructed by concatenating the collection of $n \times k_i$ matrices H_i for each clustering. The (ℓ, j) entry of H_i is one if data point ℓ appears in cluster j in clustering i, otherwise zero. One algorithm simply applies any standard similarity based clustering over the following similarity matrix $\frac{HH^T}{N}$. The (p, q) entry is the fraction of clusterings in which data point p and q appear in the same cluster. The other two algorithms apply hyper-graph based techniques where each column of H is regarded as a hyperedge.

In principle, Strehl and Ghosh's ideas can be readily adapted to heterogeneously distributed data (they did not explicitly address this issue). Each site builds a local clustering, then a centralized representation of the H matrix is constructed. To compute H directly, each site sends H_i to a central site. This, however, likely will involve too much communication on datasets with large numbers of tuples (n) because H_i is $n \times k_i$. For Strehl and Ghosh's ideas to be adapted to a distributed setting, the problem of constructing an accurate centralized representation of H using few messages need be addressed.

Fred and Jain [18] report a method for combining clusterings in a centralized setting. Given N clusterings of n data points, their method first constructs an $n \times n$, *co-association matrix* (the same as $\frac{HH^T}{N}$ as described in [54]). Next a merge algorithm is applied to the matrix using a single link, threshold, hierarchical clustering technique. For each pair (i, j) whose co-association entry is greater than a predefined threshold, merge the clusters containing these points.

In principal Fred and Jain's approach can be adapted to heterogeneously distributed data (they did not address the issue). Each site builds a local clustering, then a centralized co-association matrix is built from all clusterings Like Strehl and Ghosh's ideas; in order for Fred and Jain's approach to be adapted to a distributed setting, the problem of building an accurate co-association matrix in a message efficient manner must be addressed.

Jouve and Nicoloyannis [27] also develop a technique for combining clusterings. They use a related but different approach than those described earlier. They reduce the problem of combining clusterings to that of clustering a centralized categorical data matrix built from the clusterings and apply a categorical clustering algorithm (KEROUAC) of their own. The categorical data matrix has dimensions $n \times N$ and is defined as follows. Assume clustering $1 \leq i \leq N$ has clusters labeled $1, 2, \ldots, k_i$. The (j, i) entry is the label of the cluster (in the i^{th} clustering) containing data point j. The KEROUAC algorithm does not require the user to specify the number of clusters desired in the final clustering. Hence, Jouve and Nicoloyannis' method does not require the desired number of clusters in the combined clustering to be specified.

Like the approaches in [54] and [18], Jouve and Nicoloyannis' technique can be readily adapted to heterogeneously distributed data. A centralized categorical data matrix is built from the local clusterings, then the central site applies KEROUAC (or any other categorical data clustering algorithm). However, the problem of building an accurate matrix in a message efficient manner must be addressed (despite the fact that their title contains "Applications for Distributed Clustering", they did not address the issue).

Topchy *et al.* [56] develop an intriguing approach based on combining many weak clusterings in a centralized setting. One of the weak clusterings used projects the data onto a random, low-dimensional space (1-dimensional in their experiments) and performs K-means on the projected data. Then, several methods for combining clusterings are used based on finding a new clustering with minimum sum "difference" between each of the weak clusterings (including methods from [54]). His idea does not seem directly applicable to a distributed setting where reducing message communication is the central goal. Hence, the work saved at each site by producing a weak clustering is not of much importance. However, he discusses several new ideas for combining clusterings which are of independent interest. For example, he shows that when using generalized mutual information, maximizing the average normalized mutual information consensus measure of Strehl and Ghosh is equivalent to minimizing a square-error criterion.

Merugu and Ghosh [40] develop a method for combining generative models produced from homogeneously distributed data (a generative model is a weighted sum of multi-dimensional probability density functions *i.e.* components). Each site produces a generative model from its own local data. Their goal is for a central site to find a global model from a pre-defined family (*e.g.* multivariate, 10 component Gaussian mixtures).

which minimizes the average Kullback-Leibler distance over all local models. They prove this to be equivalent to finding a model from the family which minimizes the KL distance from the mean model over all local models (point-wise average of all local models).

They assume that this mean model is computed at some central site. Finally the central site computes an approximation to the optimal model using an EM-style algorithm along with Markov-chain Monte-carlo sampling. They did not discuss how the centralized mean model was computed. But, since the local models are likely to be considerably smaller than the actual data, transmitting the models to a central site seems to be a reasonable approach. They also discuss the privacy implications of this algorithm. We summarize their discussion in Section 4.3.

Januzaj *et al.* [25] extend a density-based centralized clustering algorithm, DBSCAN, by one of the authors to a homogeneously distributed setting. Each site carries out the DBSCAN algorithm, a compact representation of each local clustering is transmitted to a central site, a global clustering representation is produced from local representations, and finally this global representation is sent back to each site. A clustering is represented by first choosing a sample of data points from each cluster. The points are chosen such that: (i) each point has enough neighbors in its neighborhood (determined by fixed thresholds) and (ii) no two points lie in the same neighborhood. Then K-means clustering is applied to all points in the cluster, using each of the sample points as an initial centroid. The final centroids along with the distance to the furthest point in their K-means cluster form the representation (a collection point, radius pairs). The DBSCAN algorithm is applied at the central site on the union of the local representative points to form the global clustering. This algorithm requires an ϵ parameter defining a neighborhood. The authors set this parameter to the maximum of all the representation radii.

Methods [25], [40], and [49] are representatives of the centralized ensemble-based methods. These algorithms focus on transmitting compact representations of a local clustering to a central site which combines to form a global clustering representation. The key to this class of methods is in the local model (clustering) representation. A good one faithfully captures the local clusterings, requires few messages to transmit, and is easy to combine.

Both the ensemble approach and the multiple communication round-based clustering algorithms usually work a lot better than their centralized counterparts in a distributed environment. This is well documented in the literature. While, the DDM technology requires further advancement for dealing with peer-to-peer style and heterogeneous data, the current collection of algorithms offer a decent set of choices. The following section organizes the distributed clustering algorithms based on the data distribution (homogeneous vs. heterogeneous) they can handle.

4.3 Privacy Focused

Klusch *et al.* [35], [36] develop the KDEC Scheme for kernel-based distributed clustering. Each site transmits the local density estimate to a helper site, which builds a global density estimate and sends it back to the peers. Using the global density estimate the sites can execute locally a density-based clustering algorithm. Due to fact that only sample of the local densities are shared, a degree of privacy is maintained. This issue is further discussed in [8].

The paper by Merugu and Ghosh [40] described in Section 4.2 also discussed privacy. Recall, their algorithm outputs a global model F from a predefined fixed family of models *e.g.* multivariate, 10 component Gaussian mixtures. The global model approximates the underlying probability model that generated the global dataset Z. Assuming elements of Z are drawn independently, the average log-likelihood of Z given F is $AL(Z|F) = \frac{\sum_{z \in Z} log_2(Pr(z|F)}{|Z|}$. They define privacy as $P(Z, F) = 2^{-AL(Z|F)}$. Intuitively, the larger the likelihood that the data was generated by the global model, the less privacy is retained. If the predefined family has a very large number of mixture components then the privacy is likely to be low.

Vaidya and Clifton [59] develop a privacy-preserving K-means algorithm on heterogeneously distributed data using cryptographic techniques. They offer a proof that each site does not learn anything beyond its part of each cluster centroid and the cluster assignment of all points at each iteration. The key problem faced at each iteration is securely assigning each point to its nearest cluster. Since each site owns a part of each tuple (which must remain private), this problem is non-trivial. It is solved by applying the following algorithm for each point x (assuming $r \geq 3$ sites).

Let x_j and μ_j^i be the portions of x and the i^{th} centroid at the j^{th} site, respectively. Let $\overrightarrow{y}_j$ be the length K vector where y_j^i is the distance between x_j and μ_j^i. The problem boils down to securely computing $argmin_{i=1}^{K}\{\sum_{j=1}^{r} y_j^i\}$. Site 1 computes random vectors (length K) $\overrightarrow{v}_1, \ldots, \overrightarrow{v}_r$ whose sum is zero and, π, a random permutation of $\{1, \ldots, K\}$. For each $2 \leq j \leq r$, site 1 then engages in a secure algorithm allowing site j to compute $\pi(\overrightarrow{v}_j + \overrightarrow{y}_j)$. At the end of this algorithm site 1 does not know anything new and site 2 does not know π or $\overrightarrow{v}_j$. This algorithm uses homomorphic encryption to achieve security. Next, sites $1, 3, \ldots, r-1$ send $\pi(\overrightarrow{v}_j + \overrightarrow{y}_j)$ to site r. Site r sums these vectors with its own (note site r does not know the vector at site 2). Now site r and site 2 uses SMC to securely determine the index ℓ of the minimum entry of vector $\sum_{j=1}^{r} \pi(\overrightarrow{v}_j + \overrightarrow{y}_j)$. Now site 2 knows the minimum distance but not to which centroid it corresponds (due to the permutation known only to site 1). Site 2 sends ℓ to site 1, which then broadcasts $\pi^{-1}(\ell)$ to all sites *i.e.* the closest centroid.

Note that above we limited our discussion to privacy-preserving algorithms for which multiple sites compute a clustering in a distributed manner. We did not include data transformation based approaches where a data owner transforms a dataset and allows it to be download by others who then perform clustering in a centralized manner. The reader is referred to [42], [44] for two example of this approach.

4.4 Homogeneous vs. Heterogeneous Clustering Literature

A common classification of DDM algorithms in the literature is: those which apply to homogeneously distributed (horizontally partitioned) or heterogeneously distributed (vertically partitioned) data. To help the reader sort out the clustering methods we have described, we present the six-way classification seen in Figure 6.

	Homogeneous	Heterogeneous
Centralized Ensemble	[25], [38], [40], [49]	[12], [18], [27], [54]
Multiple Rounds of Communication	[10], [13], [19], [36]	[34]
Privacy Preserving	[35], [40]	[59]

Fig. 6. Six-way clustering algorithms classification

5 Sensor Networks, Distributed Clustering, and Multi-Agent Systems

Sensor networks are finding increasing number of applications in many domains, including battle fields, smart buildings, and even the human body. Most sensor networks consist of a collection of light-weight (possibly mobile) sensors connected via wireless links to each other or to a more powerful gateway node that is in turn connected with an external network through either wired or wireless connections. Sensor nodes usually communicate in a peer-to-peer architecture over an asynchronous network. In many applications, sensors are deployed in hostile and difficult to access locations with constraints on weight, power supply, and cost. Moreover, sensors must process a continuous (possibly fast) stream of data. The resource-constrained distributed environments of the sensor networks and the need for collaborative approach to solve many of the problems in this domain make multi-agent systems-architecture an ideal candidate for application development. For example, a multi-agent sensor-network application utilizing learning algorithms is reported in [52]. This work reports development of embedded sensors agents used to create an integrated and semi-autonomous building control system. Agents embedded on sensors such as temperature and light-level detectors, movement or occupancy sensors are used in conjunction with learning techniques to offer smart building functionalities. The peer-to-peer communication-based problem solving capabilities are important for sensor networks and there exists a number of multi-agent system-based different applications that explored these issues. Such systems include: an agent based referral system for peer-to-peer(P2P) file sharing networks [62], and an agent based auction system over a P2P network [41]. A framework for developing agent based P2P systems is described in [3]. Additional work in this area can be found elsewhere [52, 53, 16]. The power of multi-agent-systems can be further enhanced by integrating efficient data mining capabilities and DDM algorithms may offer a better choice for multi-agent systems since they are designed to deal with distributed systems.

Clustering algorithms may play an important role in many sensor-network-based applications. Segmentation of data observed by the sensor nodes for situation awareness, detection of outliers for event detection are only a few examples that may require clustering algorithms. The distributed and resource-constrained nature of the sensor-networks demands a fundamentally distributed algorithmic solution to the clustering problem. Therefore, distributed clustering algorithms may come in handy [32] when it comes to analyzing sensor network data or data streams.

Clustering in sensor networks offers many challenges, including:

1. limited communication bandwidth,
2. constraints on computing resources,
3. limited power supply,
4. need for fault-tolerance, and
5. asynchronous nature of the network.

Distributed clustering algorithms for this domain must address these challenges. The algorithms discussed in the previous section addresses some of the issues listed above. For example, most of these distributed clustering algorithms are lot more communication efficient compared to their centralized counterparts. There exists several exact distributed clustering algorithms, particularly for homogeneous data. In other words, the outcome of the distributed clustering algorithms are provably same as that of the corresponding centralized algorithms. For heterogeneous data, the number of choices for distributed clustering algorithms is relatively limited. However, there do exist several techniques for this latter scenario. Most of the distributed clustering algorithms are still in the domain of academic research with a few exceptions. Therefore, the scalability properties of these algorithms are mostly studied for moderately large number of nodes. Although the communication-efficient aspects of these distributed clustering algorithms help addressing the concerns regarding restricted bandwidth and power supply, the need for fault-tolerance and P2P communication-based algorithmic approach are yet to be adequately addressed in the literature.

The multiple communication round-based clustering algorithms described in Section 4 involve several rounds of message passing between nodes. Each round can be thought of as a node synchronization point (multiple sensor synchronizations are required). This may not go very well in a sensor network-style environment.

Centralized ensemble-based algorithms provide us with another option. They do not require global synchronization nor message passing between nodes. Instead, all nodes communicate a model to a central node(which combines the models). In absence of a central controlling site one may treat a *peer* as a central combiner and then apply the algorithms. We can envision a scenario in which an agent at a sensor node initiates the clustering process and as it is the requesting node, it performs the process of combining the local cluster models received from the other agents. However, most of the centralized ensemble-based method algorithms are not specifically designed to deal with stream data. This is a good direction for future research. Algorithms such as [25], [40], [49] deal with the limited communication issue by transmitting compact, lossy models (rather than complete specifications of the clusterings), which may be necessary for a sensor-network-based application.

6 Confidentiality Issues in Distributed Data Clustering

As discussed in Section 3.1, privacy issues can play an important role in DDM. In an agent-based scenario, we emphasize the privacy threat of data inference carried out by potentially colluding agents.

The inference problem involves one, possible several adversaries which try to reconstruct hidden data by drawing inferences based on known information. The problem is difficult because inference channels are not easy to detect and currently has no known general solution. The problem was first studied in statistical databases and secure multi-level databases [17]. Many approaches for the inference control were developed including based on security constraints, conceptual structures, and logic [24, 55, 17].

In a open agent system peers may collude. In general collusion is difficult to detect and is a powerful form of attack. Therefore, we also consider collusion in our discussion to follow

Definition 1 (Malicious Peers and Collusion). *Denote $\mathcal{L} = \{L^j \mid 1 \leq j \leq p\}$ a group of p peers. A peer L^k is said to be malicious if it tries to learn information about the data sets D^j, $k \neq j$. A collusion group $\mathcal{C} \subset \mathcal{L}$ is a group of malicious peers trying to learn about D^j from $L^j \in \mathcal{L} \setminus \mathcal{C}$.*

In the following we assume that the above mentioned issues are potential threats in distributed data clustering (DDC) system. In general, distributed data mining algorithms require that the sites share local information, such as data summaries or even raw data, to build up a global data mining result. Since we are assuming a peer-to-peer model, the problem becomes: how to perform DDC without compromising confidentiality of local data in each peer, even with collusion of part of the group?

6.1 Confidentiality Measure

Our starting point is the definition of a confidentiality measure, which could be used in our underling DDM model. Intuitively a measure of confidentiality in a distributed data mining context has to quantify the difficulty for one mining peer to disclose the confidential information owned by other peers. In general, the critical point in a distributed data mining algorithm, with respect to confidentiality, is the data required to be exchanged among the peers.

One way to measure how much confidentiality some algorithm preserves, is to ask how close one attacker can get from the original data objects. In the following we define the notion of confidentiality of data with respect to reconstruction. Considering multidimensional data objects, we consider each dimension individually.

Definition 2 (Reconstruction precision). *Let $\mathcal{L}$ be a group of peers, each of them with a local set of data objects $D^j = \{\mathbf{x}_i \mid i = 1, \ldots, N\} \subset \mathbb{R}^n$, with $\mathbf{x}_i^{(d)}$ denoting the d-th component of $\mathbf{x}_i$. Let $\mathcal{A}$ be some DDC algorithm executed by the members of $\mathcal{L}$. Denote by $R = \{\mathbf{r}_i \mid i = 1, \ldots, N\} \subset \mathbb{R}^n$ a set of reconstructed data objects owned by some malicious peer after the computation of the data mining algorithm, such that $\mathbf{r}_i \in R$ is the reconstructed version of $\mathbf{x}_i \in D^j$,for all i. We define the reconstruction precision of $\mathcal{A}$ as:*

$$Rec_{\mathcal{A}}(D^j, R) = \min\{|\mathbf{x}_i^{(d)} - \mathbf{r}_i^{(d)}| : \mathbf{x}_i \in D^j,\ \mathbf{r}_i \in R,\ 1 \leq i \leq N,\ 1 \leq d \leq n\}$$

Definition 3 (Confidentiality in presence of collusion). *Let $\mathcal{A}$ be a distributed data mining algorithm.Denote $D^j = \{\mathbf{x}_i \mid i = 1, \ldots, N\} \subset \mathbb{R}^n$, a set of data objects owned by peer j. Denote by $R_c \subset \mathbb{R}^n$ a set of data objects reconstructed through collusion of c peers. We define the function $Conf_{\mathcal{A}} : \mathbb{N} \rightarrow \mathbb{R}_+ \cup \{0\}$, which represents $Rec_{\mathcal{A}}$ when c peers collude, as follows:*

$$Conf_{\mathcal{A}}(c) = Rec_{\mathcal{A}}(D^j, R_c)$$

Definition 4 (Inference Risk Level in presence of collusion). *Let $\mathcal{A}$ be a DDC algorithm being executed by a group $\mathcal{L}$ with p peers, where c peers in $\mathcal{L}$ forms a collusion group. Then we define:*

$$\mathrm{IRL}_{\mathcal{A}}(c) = 2^{(-Conf_{\mathcal{A}}(c))}$$

One can easily verify that $\mathrm{IRL}_{\mathcal{A}}(c) \rightarrow 0$ when $Conf_{\mathcal{A}}(c) \rightarrow \infty$ and $\mathrm{IRL}_{\mathcal{A}}(c) \rightarrow 1$ when $Conf_{\mathcal{A}}(c) \rightarrow 0$. In other words, the better the reconstruction the higher the risk. Therefore, we can capture the informal concepts of *insecure* algorithm ($\mathrm{IRL}_{\mathcal{A}} = 1$) and *secure* ($\mathrm{IRL}_{\mathcal{A}} = 0$) as well.

In the following, we propose a classification with respect to our confidentiality level measure.

6.2 Distributed Data Mining Scenarios with Malicious Peers

Here we will define possible scenarios, with respect to the number of malicious peers, which may occur in an distributed data mining group.

Scenario 1: Individual malicious. In this scenario we consider that each peer may act maliciously, *i.e.* each peer is a potential attacker. The peers may be active attackers or just curious (semi-honest behavior). This scenario will be denoted by $c = 1$.

Scenario 2: Collusion. In this scenario the malicious peers try to form a group (or groups) of attackers, exchanging among them all necessary information to get the victim's data set reconstructed. This scenario will be represented by $c \geq 2$.

Since we cannot assure that a system will operate in a specific scenario, we have to analyze our algorithm in all possible scenarios, which will imply different $\mathrm{IRL}_{\mathcal{A}}$.

6.3 Building More Confidential Densities

In density-based DDC each peer contributes to the mining task with a local density estimate of the local data set and not with data (neither original nor randomized). Thus, we need a measure to indicate how much confidentiality a density estimate can provide.

As shown in [8], in some cases, knowing the inverse of the kernel function implies reconstruction of original (confidential) data. Therefore, we look for a more confidential way to build the density estimate, *i.e.* one which doesn't allow reconstruction of data.

Definition 5. *Let $f : \mathbb{R}_+ \cup \{0\} \to \mathbb{R}_+$ be a decreasing function and $\tau \in \mathbb{R}_+$ be a sampling rate. Let $\mathbf{v} \in \mathbb{R}^n$ be a vector of iso-levels*[5] *of f, whose component $v^{(i)}$, $1 \leq i \leq n$, is defined as follows: $v^{(n)} = f(0)$ and $\forall 1 \leq j \leq n, v^{(n-j)} = f(min\{z \in \mathbb{Z}_+ | f([z-1]\tau) = v^{(n-j+1)}$ and $f([z-1]\tau) > f(z\tau)\}\tau)$. If f is strictly decreasing, then $v^{(i)} = f([n-i]\tau)$.*

Definition 6. *Let $f : \mathbb{R}_+ \cup \{0\} \to \mathbb{R}$ be a decreasing function. Let $\mathbf{v}$ be a vector of iso-levels of f. Then we define the function $\psi_{f,\mathbf{v}}$ as:*

$$\psi_{f,\mathbf{v}}(x) = \begin{cases} 0, \textit{if } f(x) < v^{(0)} \\ v^{(i)}, \textit{if } v^{(i)} \leq f(x) < v^{(i+1)} \\ v^{(n)}, \textit{if } v^{(n)} \leq f(x) \end{cases} \tag{1}$$

Definitions 5 and 6 together define a step function based on the shape of some given function f. Figure 7 shows an example of $\psi_{f,\mathbf{v}}$ applied to a Gaussian[6] function with $\mu = 0$ and $\sigma = 2$, using four iso-levels and sampling rate τ equal to one.

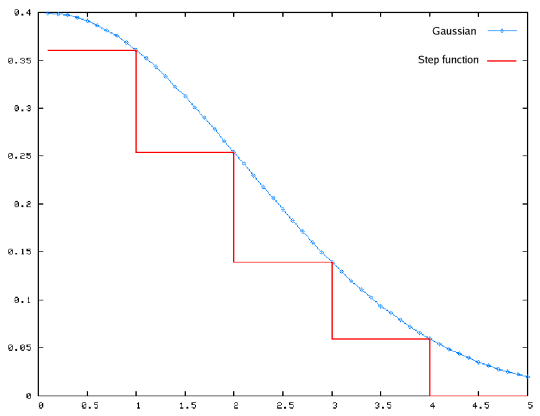

Fig. 7. $\psi_{f,\mathbf{v}}$, where f is a Gaussian.

Lemma 1. *Let $\tau \in \mathbb{R}_+$ be a sampling interval and f_1, f_2 be decreasing functions $\mathbb{R}_+ \cup \{0\} \to \mathbb{R}_+$ with the same iso-level vector v. Then, it follows that $\psi_{f_1,v} = \psi_{f_2,v}$.*

Proof. For $k = 0$ we get $f_2(x) = f_1(x - 0)$ and its is trivial to see that the assertion holds. For $0 < k < \tau$ we have $f_2 = f_1(x - k)$. Without loss of generality, let $z > 0$ be some integer. So, $f_2(z \cdot \tau) = f_1(z \cdot \tau - k) = f_1([z - k/\tau] \cdot \tau)$. If $f_1([z-1] \cdot \tau) = a > f_1(z \cdot \tau) = b$ then we have $\psi_{f_1,\mathbf{v}}(z \cdot \tau) = a$. Since $z - 1 < z - k/\tau < z$, and since f_1 is decreasing, $f_1([z-1] \cdot \tau) = a > f_1([z - k/\tau] \cdot \tau) > b = f_1(z \cdot \tau)$. By the definition 6 we can write $\psi_{f_1,\mathbf{v}}([z - k/\tau] \cdot \tau) = b = \psi_{f_1,\mathbf{v}}(z \cdot \tau)$

[5] iso-lines used in contour plots

[6] Gaussian function is defined by $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$

This lemma means that we have some ambiguity associated with the function $\psi_{f,\mathbf{v}}$, given some τ and $\mathbf{v}$, since two functions will issue the same values iso-levels around the points close together.

With this definition we return to our problem of uncertainty of local density. Since density estimation are additive, we can define the global density estimate, given a decreasing kernel function K and bandwidth $h > 0$ as:

$$\varphi[D](x) = \sum_{j=1}^{p} \varphi[D^j](x) \tag{2}$$

where $\varphi[D^j](x) = \sum_{x_i \in D^j} K(\frac{d(x,x_i)}{h})$.

We will replace kernel K with $\psi_{K,\mathbf{v}}$, given a sample rate τ. According to Lemma 1, we should expect to localize the points in a interval not smaller than $|(0,\tau)|$, *i.e.* the confidentiality will be $Conf_{\mathcal{A}} \geq \tau$. So, we will compute a rough approximation of the local density estimate using:

$$\sum_{x_i \in D^j} \psi_{K,\mathbf{v}}(\frac{d(x,x_i)}{h}) \tag{3}$$

Since $\psi_{K,\mathbf{v}}$ is a non-increasing function, we can use it as a kernel function. For input values close to 0 the function $\psi_{K,\mathbf{v}}$ will issue the max value and for larger values $\psi_{K,\mathbf{v}}$ will be zero. Therefore, for points where there is a higher concentration of points, a dense region, bigger values of $\tilde{\varphi}[D^j](x)$ will be computed.

The global approximation will be computed as follow:

$$\tilde{\varphi}[D](x) = \sum_{j=1}^{p} \tilde{\varphi}[D^j](x) \tag{4}$$

7 A Secure Density-Based Distributed Clustering Algorithm

The basic idea of KDEC-S is the observation that the clustering algorithm doesn't need the exact density estimate function but an essential approximation. The "essential approximation" in this case is a sampling of points which is as coarse as possible to preserve data confidentiality while maintaining information to guide the clustering process.

7.1 Basic definitions

Definition 7. *Given two vectors $\mathbf{z}_{low}, \mathbf{z}_{high} \in \mathbb{Z}^n$, which differ in all coordinates (called the sampling corners), we define a grid G as the filled-in cube in $\mathbb{Z}^n$ defined by $\mathbf{z}_{low}, \mathbf{z}_{high}$. Moreover for all $z \in G$, define $n_z \in \mathbb{N}$ as a unique index for z (the index code of z). Assume that z_{low} has index code zero.*

Definition 8. *Let G be a grid, and $\tau \in \mathbb{R}^n$ be a sampling rate. We define a sampling $\mathcal{S}^j$ of $\tilde{\varphi}[D^j]$, as:*

$$\mathcal{S}^j = \left\{(\tilde{\varphi}[D^j](z\tau), z) \mid z \in G,\ \tilde{\varphi}[D^j](z\tau) > 0\right\}$$

where $z\tau$ denote coordinate-wise multiplication. Similarly, the global sampling set will be defined as:

$$\mathcal{S} = \{(\tilde{\varphi}[D](z\tau), z) \mid z \in G,\ \tilde{\varphi}[D](z\tau) > 0\}$$

Definition 9 (Cluster-guide). *A cluster guide $CG_{i,\theta}$ is a set of index codes representing the grid points forming a region with density above some threshold θ:*

$$CG_{i,\theta} = \{n_z \,|\tilde{\varphi}[D](z\tau) \geq \theta\}$$

such that

$$\forall n_{z_1}, n_{z_2} \in CG_{i,\theta} : z_1 \textit{ and } z_2 \textit{ are grid neighbors}$$

Observe that any two cluster guides are either equal or disjoint *i.e.* there are no partially overlapping guides. Let CG_θ denote the collection of all cluster guides. CG_θ is called a complete cluster guide.

$$CG_\theta = \{CG_{i,\theta} |\, i = 1, \ldots, C\}$$

where C is the number of clusters found using a given θ.

A cluster-guide $CG_{i,\theta}$ can be viewed as a contour defining the cluster shape at level θ (an iso-line), but in fact it shows only the internal grid points and not the true border of the cluster, which should be determined using the local data set.

7.2 Detailed description

Our algorithm has two parts: Local Peer and Helper. The local peer part of our distributed algorithm is density-based, since this was shown to be a more general approach to clustering [23].

Initialization. The first step is the function negotiate(), which succeeds only if an agreement on the parameters is reached.

Sampling set. Using Definition 8, each local peer builds its local sampling set and sends it to the helper site, $\mathcal{H}$.

Clustering. The clustering step (line 5 in Algorithm 7.2.1) is performed as a lookup in the cluster-guide CG_θ. The function cluster() shows the details of the clustering step. The data object $\mathbf{x} \in D^j$ will be assigned to the cluster i, the cluster label of the nearest grid point z, if $n_z \in CG_{i,\theta}$.

Algorithm 7.2.1 Local Peer

```
Require:
    a local data set $D^j$
    a list of peers $\mathcal{L}$ and a special helper peer $\mathcal{H}$;
Ensure: $clusterMap$;

 1: negotiate($\mathcal{L}, K, h, G, \tau, \theta$);
 2: $S^j \leftarrow$ buildSamplingSet($K, h, D^j, G, \theta, v, \tau$);
 3: send($\mathcal{H}, S^j$);
 4: $CG_\theta \leftarrow$ request($\mathcal{H}, \theta$);
 5: $clusterMap \leftarrow$ cluster($CG_\theta$, $D^j$, $G$);
 6: return $clusterMap$

 7: function CLUSTER($CG_\theta, D^j, G$)
 8:     for each $\mathbf{x} \in D^j$ do
 9:         $z \leftarrow$ nearestGridPoint($\mathbf{x}$, $G$);
10:         if $n_z \in CG_{i,\theta}$ then
11:             $clusterMap(\mathbf{x}) \leftarrow i$;
12:         end if
13:     end for
14:     return $clusterMap$;
15: end function
```

Algorithm 7.2.2 Helper

```
 1: $\{S^j\} \leftarrow$receive($\mathcal{L}$);
 2: $S \leftarrow$reconstructGlobalSampling($\{S^j\}$);
 3: $CG_\theta \leftarrow$ buildClusterGuides($S, \theta$);
 4: send($\mathcal{L}$, $CG_\theta$);

 5: function BUILDCLUSTERGUIDES($S, \theta$)
 6:     $cg \leftarrow \{n_z | (\tilde{\varphi}[D](z\tau), z) \in S, \tilde{\varphi}[D](z\tau) \geq \theta\}$;
 7:     $i \leftarrow 0$;
 8:     Let $n$ be any element of $cg$;
 9:     $CG_{i,\theta} \leftarrow \{n\}$;
10:     while $cg$ is not empty do
            $CG_{i,\theta} \leftarrow \{n\} \cup CG_{i,\theta}$;
            $cg \leftarrow cg \setminus \{n\}$;
11:         if $\exists a \in cg$ such that $a \in neighbors(n)$ then
12:             $n \leftarrow a$;
13:         else
14:             $i$++;
15:         end if
16:     end while
17:     return $CG_\theta$, the collection of all $CG_{i,\theta}$ ;
18: end function
```

Building Cluster-Guides. The helper first reconstructs the global sampling set from all the local sampling sets. Note, only the grid points which appear in some local sampling set need be considered. Next, given θ, the helper uses Definition 9 to construct the cluster guides CG_θ. Function buildClusterGuides() in Algorithm 7.2.2 shows the details of this step.

7.3 Performance Analysis

Local Site. The computation performed by local site j has worst case time $O([M^j + log(|G|)]N)$ where M^j is the number of grid points whose $\tilde{\varphi}[D^j]$ density is non-zero. Note, since $\tilde{\varphi}[D^j]$ has bounded support, then grid points sufficiently far away from any point in D^j will have zero density. For datasets D^j occupying only a small portion of the grid space, M^j will be much smaller than $|G|$.

To build sample set S^j, the site only must examine those grid points z for which there is a data point in D^j whose distance from $z\tau$ is within the support range of $\tilde{\varphi}[D^j]$. This set of grid points (size M^j)can be computed in time $O(M^jN)$. Then, for each of these grid points, a linear scan of D^j will yield their density. Hence S^j can be computed in time $O(M^jN)$. To assign each data point x in D^j to a cluster based on the cluster guides received from the helper (function cluster), the site must determine which cluster contains x. Assuming a constant number of clusters and $log(|G|)$ look-up time for any given cluster, the complexity of this step is $O(log(|G|)N)$.

Helper. The computation carried out by the helper has worst case time complexity $O(pM)$ where $M = \sum_{j=1}^{p} M^j$. Reconstructing the global sample points requires, for each grid point occurring in some local sample, summing over all local samples. Building the cluster guide from the global sampling can be done in time $O(M)$ provided that the time to find all neighbors of a given point in the grid is constant.

Communication The total number of messages sent is at most $O(Mp)$. Each site will have at most M sampling points to send to the helper site. A total of Mp messages. The helper site will need to send the cluster guide back to each site, at most Mp messages.

7.4 Security Analysis

We will use our scenarios ($c = 1$, and $c \geq 2$) to analyze the inference risk level of KDEC-S (denoted $\mathrm{IRL}_{\text{KDEC-S}}$).

Lemma 2. *Let $\mathcal{L}$ be a mining group formed by $p > 2$ peers, one of them being the helper, and $c < p$ malicious peers form a collusion group in $\mathcal{L}$. Let $\tau \in \mathbb{R}$ be a sampling rate. If $c \geq 1$ then $\mathrm{IRL}_{\text{KDEC-S}}(c) \leq 2^{-\tau}$.*

Proof. Assume that $c = 1$, and that each peer has only its local data set and the cluster-guides he gets from the helper. The cluster-guides, which are produced by the helper, contains only code-index representing grid points where the threshold θ is reached. This is not enough to reconstruct the original global estimation. The Helper has all sampling points from all peers, but it has no information on the kernel nor on sampling parameters. Hence, the attackers can not use the inverse of Kernel function to reconstruct the data. The best precision of reconstruction have to be based on the cluster guides. So, one attacker may use the width of the clusters in each dimension as the best reconstruction precision. This lead to $Conf_{\text{KDEC-S}}(1) = a\tau$, with $a \in \mathbb{N}$, since each cluster will have at least a points spaced by τ in each dimension. Hence, if $c = 1$ then $\text{IRL}_{\text{KDEC-S}}(c) = 2^{-a\tau} \leq 2^{-\tau}$.

Assume $c \geq 2$. Clearly, any collusion group with at least two peers, including the helper, will produce better results than one collusion which doesn't include the helper, since the helper can send to the colluders the original sampling sets from each peer. However, each sampling set $\mathcal{S}^j$ was formed based on the $\tilde{\varphi}[D^j]$ (cf. eq. (3)). Using lemma 1 we expect to have $Conf_{\text{KDEC-S}}(c) = \tau$. With more colluders, say $c = 3$, one of them being the helper, there are no new information which could improve the reconstruction. Hence, $\forall c \in \mathbb{N}(c > 1 \rightarrow Conf_{\text{KDEC-S}}(c) = \tau)$. Therefore, if $c \geq 2$ then $\text{IRL}_{\text{KDEC-S}}(c) = 2^{-\tau}$.

Comment: In the Statistics literature, many techniques have been developed for selecting the bandwidth parameter, h, *e.g.* Silverman's rule-of-thumb, direct plug-in, smoothed cross-validation. In our distributed setting, all of these will reveal some information regarding sites' local data. Currently, we do not consider these in our privacy analysis and assume that h is chosen objectively by the sites in such a way to not reveal extra information. For example, h could be set to some fraction of a public bound on the data range and readjusted given the algorithm results. As future work we are investigating the incorporation of bandwidth selection techniques from the Statistics literature in our privacy framework.

8 Experimental Evaluation

We conducted some experiments to evaluate how the increasing privacy (τ) affects clustering results.[7] We used two synthetic, two-dimensional data sets. The first consists of 500 points, generated from a mixture model of four Gaussian distributions, each with $\sigma^2 = 1$ in all dimensions. The second consists of 400 points. First, 200 points were generated from a Gaussian with $\mu = 0$ and $\sigma^2 = 5$. Next, 200 points were generated around the center each with radius $R \sim N(20, 1)$ and angle $\sim U(0, 2\pi)$.

We applied our algorithm to both data sets with the following parameters: bandwidth $h = 1$,neighborhood radius fixed in 4, reference tau set to $\tau_{ref} = h/2$, and value of τ going from 0.5 to 3.0 with step 0.1. For the Gaussian data set we used grid corners ((-15,-15), (15, 15)) and threshold $\theta = 1.0$. For the polar data set we used grid corners ((-30, -30),(30,30)) and threshold $\theta = 0.1$.

We counted the mislabeling error, considering as correct the clustering obtained with τ_{ref} = h/2. We follow [37] and compute the clustering error as follows:

$$E = \frac{2}{|D|(|D| - 1)} \sum_{x_i, x_j \in D, i<j} e_{ij}$$

[7] Since the goal was to measure clustering results and not communication or computational complexity, a distributed algorithm was not necessary. All experiments involve a centralized data set and algorithm.

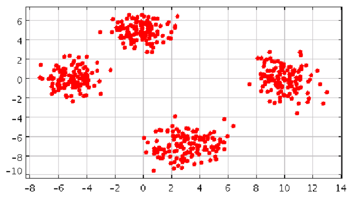

Fig. 8. Gaussian Data: four clusters generated from a Gaussian mixture.

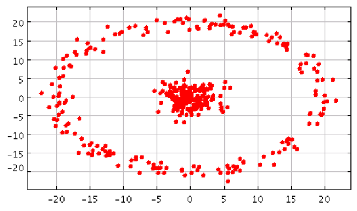

Fig. 9. Polar Data: two clusters with arbitrary shape.

where $|D|$ is the size of the data set and e_{ij} is defined as:

$$e_{ij} = \begin{cases} 0 & \text{if } (c(x_i) = c(x_j) \wedge c'(x_i) = c'(x_j)) \quad \vee \\ & (c(x_i) \neq c(x_j) \wedge c'(x_i) \neq c'(x_j)) \\ 1 & \text{otherwise} \end{cases}$$

with $c(x)$ denoting the reference cluster label assigned to object x, *i.e.* the label found using τ_{ref}, and $c'(o)$ denoting the new label found with $\tau > \tau_{ref}$.

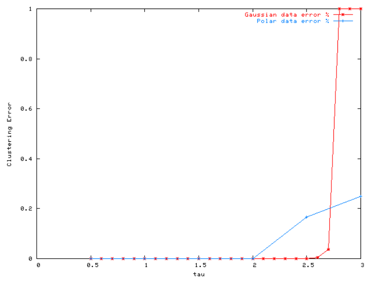

Fig. 10. Clustering error bandwidth $h = 1$ and τ ranging from 0.5 to 3.

From the tests results we see that τ can be set as large as $2h$ with no change in clustering results. For the Gaussian data, with $h = 1$, error appears just after $\tau = 2.5$ and in the polar data set, just after $\tau = 2$. The increase in error is due to the fact that with larger τ, more grid points are considered as outliers *i.e.* their density does not exceed the threshold θ. Since the Gaussian kernel goes to zero exponentially fast (is nearly zero around $3h$), the error ought to become large since the iso levels summation does not reach the given threshold. Consequently the correspondent grid point is left out from the cluster guides. A possible solution is to use adaptive thresholds or even adaptive iso-lines. We are working on this issue.

Comment: We have left the choice of θ to be made objectively by the sites so as to not reveal any extra information. For example, θ could be set very small and increased based on the output of the algorithm. As future work we are investigating data dependent methods for helping users tune θ. For example, if K is assumed to satisfy the conditions of a probability distribution and $\tilde{\varphi}[D]$ is divided by $|D|h^n c(K, \mathbf{v})$ where $c(\mathbf{v}) = \int_{\mathbb{R}^n} \psi_{K,\mathbf{v}}(z)dz$, the resulting function also satisfies the conditions of a probability distribution. Hence the density at a grid point can be naturally interpreted as a probability, thus, helping the user choose θ. However, securely normalizing $\tilde{\varphi}[D]$ in this way is non-trivial since $|D|$ cannot be learned by any peer. We are investigating secure multi-party techniques for addressing this issue.

9 Conclusions

Multi-agent systems are fundamentally designed for collaborative problem solving in distributed environments. Many of these application environments deal with empirical analysis and mining of data. This paper suggests that traditional centralized data mining techniques may not work well in many distributed environments where data centralization may be difficult because of limited bandwidth, privacy issues and/or the demand on response time.

This paper pointed out that distributed data mining algorithms may offer a better solution since they are designed to work in a distributed environment by paying careful attention to the computing and communication resources. The paper focused on distributed clustering algorithms. It surveyed the data mining literature on distributed and privacy-preserving clustering algorithms. It discussed sensor networks with peer-to-peer architectures as an interesting application domain and illustrated some of the existing challenges and weaknesses of the DDM algorithms. It noted that while these algorithms usually perform better than their centralized counter-parts on grounds of communication efficiency and power consumption, there exist several open issues. Developing peer-to-peer versions of these algorithms for asynchronous networks and paying attention to fault-tolerance are some examples. Also, this paper presents a new privacy-preserving density based clustering algorithm (not for sensor networks).

In closing, existing pleasures of distributed clustering algorithms do provide a reasonable class of interesting choices for the next generation of multi-agent systems that may require analysis of distributed data.

Acknowledgments

C. Giannella, R. Bhargava, and H. Kargupta thank the U.S. National Science Foundation for support through grants IIS-0329143 and IIS-0093353. They also thank Haimonti Dutta for many useful discussions. J. Silva thanks the CAPES (Coord. de Aperfeicoamento do Pessoal de Nivel Superior) of Ministry for Education of Brazil for support through grant No. 0791/024. This article is part of the special issue of selected best papers of the 9th international workshop on cooperative information agents (CIA 2004) organised by Matthias Klusch, Rainer Unland, and Sascha Ossowski.

References

1. Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000.
2. M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure limitation of sensitive rules. In *Proceedings of 1999 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'99)*, pages 45–52, Chicago,IL, November 1999.
3. Babaoglu O., Meling H., and Montresor A. Anthill: a Framework for the Development of Agent-Based Peer-to-Peer Systems. Technical Report 9, Department of Computer Science, University of Bologna, November 2001.
4. Babcock B., Babu S., Datar M., Motwani R., and Widom J. Models and Issues in Data Stream Systems. In *Proceedings of the 21th ACM SIGMOD-SIGACT-SIGART Symposium on Principals of Database Systems (PODS)*, pages 1–16, 2002.
5. C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M.Y. Zhu. Tools for privacy preserving data mining. *ACM SIGKDD Explorations Newsletter key*, 4(2):28–34, 2002.
6. Chris Clifton. Using sample size to limit exposure to data mining. *Journal of Computer Security*, 8(4):281–307, 2000.
7. Chris Clifton and Don Marks. Security and privacy implications of data mining. In *Workshop on Data Mining and Knowledge Discovery*, pages 15–19, Montreal, Canada, 1996. University of British Columbia Department of Computer Science.
8. Josenildo C. da Silva, Matthias Klusch, Stefano Lodi, and Gianluca Moro. Inference attacks in peer-to-peer homogeneous distributed data mining. In *16th European Conference on Artificial Intelligence (ECAI 04)*, Valencia, Spain, August 2004.
9. Elena Dasseni, Vassilios S. Verykios, Ahmed K. Elmagarmid, and Elisa Bertino. Hiding association rules by using confidence and support. *Lecture Notes in Computer Science*, 2137:369–??, 2001.
10. Dhillon I. and Modha D. A Data-clustering Algorithm on Distributed Memory Multiprocessors. In *Proceedings of the KDD'99 Workshop on High Performance Knowledge Discovery*, pages 245–260, 1999.
11. Wenliang Du and Zhijun Zhan. Building decision tree classifier on private data. In Chris Clifton and Vladimir Estivill-Castro, editors, *IEEE ICDM Workshop on Privacy, Security and Data Mining*, volume 14 of *Conferences in Research and Practice in Information Technology*, pages 1–8, Maebashi City, Japan, 2002. ACS.
12. Johnson E. and Kargupta H. Collective, Hierarchical Clustering From Distributed, Heterogeneous Data. In M. Zaki and C. Ho, editors, *Large-Scale Parallel KDD Systems. Lecture Notes in Computer Science*, volume 1759, pages 221–244. Springer-Verlag, 1999.
13. Eisenhardt M., Muller W., and Henrich A. Classifying Documents by Distributed P2P Clustering. In *Proceedings of Informatik 2003, GI Lecture Notes in Informatics, Frankfort, Germany*, 2003.
14. A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proceedings of 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, Edomonton, Alberta, Canada, 2002.
15. Evfimievski A., Gehrke J., Srikant R. Limiting Privacy Breaches in Privacy Preserving Data Mining. In *Proceedings of the 2003 Symposium on the Principals of Database Systems (PODS)*, 2003.
16. Farinelli A., Grisetti G., Iocchi L.,Lo Cascio S.,Nardi D. Design and Evaluation of Multi-Agent Systems for Rescue Operations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 4, pages 3148–3143, 2003.

17. Csilla Farkas and Sushil Jajodia. The inference problem: A survey. *ACM SIGKDD Explorations Newsletter*, 4(2):6–11, 2002.
18. Fred A. and Jain A. Data Clustering Using Evidence Accumulation. In *Proceedings of the International Conference on Pattern Recognition 2002*, pages 276–280, 2002.
19. Forman G. and Zhang B. Distributed Data Clustering Can Be Efficient and Exact. *SIGKDD Explorations*, 2(2):34–38, 2000.
20. Han J. and Kamber M. *Data Mining: Concepts and Techniques*. Morgan Kaufman Publishers, San Francisco, CA, 2001.
21. Hand D., Mannila H., and Smyth P. *Principals of Data Mining*. MIT press, Cambridge, Mass, 2001.
22. Hastie T., Tibshirani R., and Friedman J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, Berlin, Germany, 2001.
23. Hinneburg A. and Keim D. An Efficient Approach to Clustering in Large Multimedia Databases with Noise. In *Proceedings of the 1998 International Confernece on Knowledge Discovery and Data Mining (KDD)*, pages 58–65, 1998.
24. S. Jajodia and C. Meadows. Inference problems in multilevel secure database management systems. In Marshall D. Abrams, Sushil Jajodia, and Harold J. Podell, editors, *Information Security: An Integrated Collection of Essays*, chapter 24. IEEE Computer Society Press, Los Alamitos, California, USA, 1995.
25. Januzaj E., Kriegel H.-P., and Pfeifle M. DBDC: Density Based Distributed Clustering. In *Proceedings of EDBT in Lecture Notes in Computer Science 2992*, pages 88–105, 2004.
26. Tom Johnsten and Vijay V. Raghavan. A methodology for hiding knowledge in databases. In Chris Clifton and Vladimir Estivill-Castro, editors, *IEEE ICDM Workshop on Privacy, Security and Data Mining*, volume 14 of *Conferences in Research and Practice in Information Technology*, pages 9–17, Maebashi City, Japan, 2002. ACS.
27. Jouve P. and Nicoloyannis N. A New Method for Combining Partitions, Applications for Distributed Clustering. In *Proceedings of Workshop on Parallel and Distributed Computing for Machine Learning as part of the 14th European Conference on Machine Learning*, 2003.
28. Kahn J., Katz R., and Pister K. Mobile networking for smart dust. In *ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 99)*, 1999.
29. Murat Kantarcioglu and Chris Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *The ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'02)*, June 2002.
30. Kargupta H. and Chan P. (editors). *Advances in Distributed and Parallel Knowledge Discovery*. AAAI press, Menlo Park, CA, 2000.
31. Kargupta H. and Sivakumar K. Existential Pleasures of Distributed Data Mining. In *Data Mining: Next Generation Challenges and Future Directions, edited by H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha, MIT/AAAI Press*, 2004.
32. Kargupta H., Bhargava R., Liu K., Powers M., Blair P., and Klein M. VEDAS: A Mobile Distributed Data Stream Mining System for Real-Time Vehicle Monitoring. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, 2004.
33. Kargupta H., Datta S., Wang Q., and Sivakumar K. Random Data Perturbation Techniques and Privacy-Preserving Data Mining. *Knowledge and Information Systems*, 7(4):in press, 2005.
34. Kargupta H., Huang W., Sivakumar K., and Johnson E. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems Journal*, 3:422–448, 2001.
35. Matthias Klusch, Stefano Lodi, and Gianluca Moro. Agent-based distributed data mining: the KDEC scheme. In Matthias Klusch, Sonia Bergamaschi, Pete Edwards, and Paolo Petta, editors, *Intelligent Information Agents: the AgentLink perspective*, volume 2586 of *Lecture Notes in Computer Science*. Springer, 2003.

36. Klusch M., Lodi S., and Moro G. Distributed Clustering Based on Sampling Local Density Estimates. In *Proceedings of the Joint International Conference on AI (IJCAI 2003)*, 2003.
37. N. Labroche, N. Monmarché, and G. Venturini. A new clustering algorithm based on the chemical recognition system of ants. In F. van Harmelen, editor, *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 345–349, Lyon, France, july 2002. IOS Press.
38. Lazarevic A., Pokrajac D., and Obradovic Z. Distributed Clustering and Local Regression for Knowledge Discovery in Multiple Spatial Databases. In *Proceedings of the 8th European Symposium on Artificial Neural Networks*, pages 129–134, 2000.
39. Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. *Lecture Notes in Computer Science*, 1880:36–54, 2000.
40. Merugu S. and Ghosh J. Privacy-Preserving Distributed Clustering Using Generative Models. In *Proceedings of the IEEE Conference on Data Mining (ICDM)*, 2003.
41. Ogston E. and Vassiliadis, S. . A Peer-to-Peer Agent Auction. In *First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 150–159, 2002.
42. Stanley Oliveira and Osmar R. Zaiane. Privacy preserving clustering by data transformation. In *Proc. of SBBD 2003*, Manaus, AM, Brasil, 2003.
43. Stanley R. M. Oliveira and Osmar R. Zaiane. Privacy preserving frequent itemset mining. In Chris Clifton and Vladimir Estivill-Castro, editors, *IEEE ICDM Workshop on Privacy, Security and Data Mining*, volume 14 of *Conferences in Research and Practice in Information Technology*, pages 43–54, Maebashi City, Japan, 2002. ACS.
44. Oliveira S. and Zaïane O. Privacy-Preserving Clustering by Object Similarity-Based Representation and Dimensionality Reduction Transformation. In *Proceedings of the Workshop on Privacy and Security Aspects of Data Mining (PSDM) as part of ICDM*, pages 21–30, 2004.
45. Park B. and Kargupta H. Distributed Data Mining: Algorithms, Systems, and Applications. In *The Handbook of Data Mining, edited by N. Ye, Lawrence Erlbaum Associates*, pages 341–358, 2003.
46. Benny Pinkas. Cryptographic techniques for privacy-preserving data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):12–19, 2002.
47. Provost F. Distributed Data Mining: Scaling Up and Beyond. In *Advances in Distributed and Parallel Knowledge Discovery, edited by H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha, MIT/AAAI Press*, pages 3–27, 2000.
48. Shariq J. Rizvi and Jayant R. Haritsa. Maintaining data privacy in association rule mining. In *Proceedings of the 28th VLDB – Very Large Data Base Conference*, pages 682–693, Hong Kong, China, 2002.
49. Samatova N., Ostrouchov G., Geist A., and Melechko A. RACHET: An Efficient Cover-Based Merging of Clustering Hierarchies from Distributed Datasets. *Distributed and Parallel Databases*, 11(2):157–180, 2002.
50. Yucel Saygin, Vassilios S. Verykios, and Chris Clifton. Using unknowns to prevent discovery of association rules. *ACM SIGMOD Record*, 30:45–54, December 2001.
51. Yucel Saygin, Vassilios S. Verykios, and Ahmed K. Elmagarmid. Privacy preserving association rule mining. In *Reseach Issues in Data Engineering (RIDE)*, 2002.
52. Sharples S., Lindemann C., and Waldhorst O. A Multi-Agent Architecture For Intelligent Building Sensing and Control. In *International Sensor Review Journal*, 1999.
53. Soh L.-K. and Tsatsoulis C. Reflective Negotiating Agents for Real-Time Multisensor Target Tracking. In *International Joint Conference On Artificial Intelligence*, 2001.
54. Strehl A. and Ghosh J. Cluster Ensembles – A Knowledge Reuse Framework for Combining Multiple Partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.
55. Bhavani Thuraisingham. Data mining, national security, privacy and civil liberties. *ACM SIGKDD Explorations Newsletter*, 4(2):1–5, 2002.

56. Topchy A., Jain A., and Punch W. Combining Multiple Weak Clusterings. In *Proceedings of the IEEE Conference on Data Mining (ICDM)*, 2003.
57. Jaideep Vaidya and Chris Clifton. Secure set intesection cardinality with application to association rule mining, March 2003. Submited to ACM Transactions on Information and Systems Security.
58. Jaydeep Vaidya and Chris Clifton. Privacy preserving association rules mining in vertically partitioned data. In *Proceedings of 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 639–644, Edomonton, Alberta, Canada, 2002.
59. Vaidya J. and Clifton C. Privacy-Preserving K-means Clustering Over Vertically Partitioned Data. In *Proceedings of the SIGKDD*, pages 206–215, 2003.
60. V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *In SIGMOD Record*, 33(1):50–57, March 2004.
61. Witten I. and Frank E. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufman Publishers, San Fransisco, 1999.
62. Yu B. and Singh M. Emergence of Agent-Based Referral Networks. In *Proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1208–1209, 2002.
63. Zaki M. Parallel and Distributed Association Mining: A Survey. *IEEE Concurrency*, 7(4):14–25, 1999.
64. Zaki M. Parallel and Distributed Data Mining: An Introduction. In *Large-Scale Parallel Data Mining (Lecture Notes in Artificial Intelligence 1759), edited by Zaki M. and Ho C.-T., Springer-Verlag, Berlin*, pages 1–23, 2000.