Multi-agent Systems and Distributed Data Mining

Chris Giannella, Ruchita Bhargava, and Hillol Kargupta

Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, MD 21250 USA
{cgiannel,ruchita1,hillol}@cs.umbc.edu

Abstract. Multi-agent systems offer an architecture for distributed problem solving. Distributed data mining algorithms specialize on one class of such distributed problem solving tasks—analysis and modeling of distributed data. This paper offers a perspective on distributed data mining algorithms in the context of multi-agents systems. It particularly focuses on distributed clustering algorithms and their potential applications in multi-agent-based problem solving scenarios. It discusses potential applications in the sensor network domain, reviews some of the existing techniques, and identifies future possibilities in combining multi-agent systems with the distributed data mining technology.

Keywords: multi-agent systems, distributed data mining, clustering

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

This paper makes an effort to underscore the possible synergy between multi-agent systems and distributed data mining 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 discusses one of these application domains, illustrates the ideas, and reviews existing work in this area. Although, the power of DDM is not just restricted to clustering, this paper chooses to restrict the scope for the sake of brevity.

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 that are drawing an increasing amount of interest from the multi-agent systems community. Finally, Section 6 concludes this 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 existing domain theory and also the outcome of empirical data analysis. 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 [15] 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 site 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. Table 1 shows 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 variance one. A and B compute $\hat{a} = R\boldsymbol{a}$, $\hat{b} = R\boldsymbol{b}$, respectively.
3. A sends $\hat{a}$ to B. B computes $\hat{a}^T \hat{b} = \boldsymbol{a}^T R^T R \boldsymbol{b}$. [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 $k \ll m$ 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 Table 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.

Privacy of the data can be another reason for adopting the DDM technology. In many applications, particularly in security-related applications, data are privacy-sensitive. When the data are multi-party and privacy-sensitive, centralizing the data is usually not acceptable. Therefore, many data mining applications in such domains must analyze data in a distributed fashion without having to first download everything to a single site. There exists a growing number of DDM algorithms that address many data mining problems for distributed environments. The following section presents an overview.

3 Distributed Data Mining: A Brief Overview

Data mining [9], [10], [11],and [31] deals with the problem of analyzing data in scalable manner. Distributed data mining is a branch of the field of data mining that offers a

Table 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)

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

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 1 and 2 illustrate the homogeneously distributed case using an example from weather data. Both tables use the same set of attributes. On the other hand, Figures 3 and 4 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 may have resource constraints.
4. Sites may 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 [25], “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 [17] and [24] 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 [16] 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 [33] discusses parallel and distributed association rule mining in DDM. Survey [34] 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 [27, 18] 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 [2].

Instead of looking at the broad spectrum of different DDM algorithms, this paper restricts itself to distributed clustering methods and their applicability in multi-agent systems. The following section addresses this issue.

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. 1. 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. 2. 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. 3. Heterogeneously distributed weather data

Timestamp	Ice Cream	Pizzas	Milk
t_0	1.2 quart	9.9	14 quart
t_1	1.9 quart	7.8	12 quart
t_2	2.7 quart	6.7	13 quart
t_3	4.2 quart	5.2	14.5 quart

Fig. 4. Heterogeneously distributed grocery store data

4 Distributed Clustering Algorithms

In this section, we present an overview of various distributed clustering solutions proposed to date. We classify distributed clustering algorithms into two categories. The first group consists of methods requiring multiple rounds of message passing. These methods require a significant amount synchronization. The second group 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 Multiple Communication Round Algorithms

Dhillon and Modha [3] 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 [8] take an approach similar to the one presented in [3], but extend it to K-harmonic means. Note that the methods of [3] and [8] 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.* [19] 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.* [20] consider kernel-density based clustering over homogeneously distributed data. They adopt the definition of a density based cluster from [12] 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.* [5] 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, [19] and [20] require only two rounds. The others require as many rounds as the algorithm iterates (potentially many more than two).

4.2 Centralized Ensemble-Based Methods

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 discussed in [22].

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 [4] 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.* [21] 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.* [26] 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 [29] 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 [7] 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 [29]). 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 [14] 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 [29] and [7], 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.* [30] 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 [29]). 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 [22] 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.

Januzaj *et al.* [13] 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 [13], [22], and [26] 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.

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 four-way classification seen in Table 4.2.

	Homogeneous	Heterogeneous
Centralized Ensemble	[13], [21], [22], [26]	[4], [7], [14], [29]
Multiple Rounds of Communication	[3], [5], [8], [20]	[19]

Fig. 5. Four-way clustering algorithms classification

The following section considers a specific instance of a DDM problem—analyzing data in a sensor network with 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.

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 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 [27]. 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 [32], and an agent based auction system over a P2P network [23]. A framework for developing agent based P2P systems is described in [1]. Additional work in this area can be found elsewhere [27, 28, 6]. 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 are likely to 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 handy [18] 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. That is something that we may need to address in the immediate future. Algorithms such as [13], [22], [26] 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. The following section concludes this paper.

6 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 and their applications in sensor networks just to illustrate some of the existing challenges and weaknesses of the DDM algorithms. It noted that while these algorithms usually perform way 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. Nevertheless, 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

The authors thank the U.S. National Science Foundation for support through grants IIS-0329143 and IIS-0093353. The authors also thank Haimonti Dutta for many useful discussions.

References

1. 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.
2. 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.
3. 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.

4. 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.
5. 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.
6. 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.
7. Fred A. and Jain A. Data Clustering Using Evidence Accumulation. In *Proceedings of the International Conference on Pattern Recognition 2002*, pages 276–280, 2002.
8. Forman G. and Zhang B. Distributed Data Clustering Can Be Efficient and Exact. *SIGKDD Explorations*, 2(2):34–38, 2000.
9. Han J. and Kamber M. *Data Mining: Concepts and Techniques*. Morgan Kaufman Publishers, San Francisco, CA, 2001.
10. Hand D., Mannila H., and Smyth P. *Principals of Data Mining*. MIT press, Cambridge, Mass, 2001.
11. Hastie T., Tibshirani R., and Friedman J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, Berlin, Germany, 2001.
12. 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.
13. 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.
14. 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.
15. 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.
16. Kargupta H. and Chan P. (editors). *Advances in Distributed and Parallel Knowledge Discovery*. AAAI press, Menlo Park, CA, 2000.
17. 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.
18. 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.
19. 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.
20. 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.
21. 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.
22. Merugu S. and Ghosh J. Privacy-Preserving Distributed Clustering Using Generative Models. In *Proceedings of the IEEE Conference on Data Mining (ICDM)*, 2003.
23. 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.

24. 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.
25. 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.
26. 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.
27. Sharples S., Lindemann C., and Waldhorst O. A Multi-Agent Architecture For Intelligent Building Sensing and Control. In *International Sensor Review Journal*, 1999.
28. Soh L.-K. and Tsatsoulis C. Reflective Negotiating Agents for Real-Time Multisensor Target Tracking. In *International Joint Conference On Artificial Intelligence*, 2001.
29. Strehl A. and Ghosh J. Cluster Ensembles – A Knowledge Reuse Framework for Combining Multiple Partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.
30. Topchy A., Jain A., and Punch W. Combining Multiple Weak Clusterings. In *Proceedings of the IEEE Conference on Data Mining (ICDM)*, 2003.
31. Witten I. and Frank E. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufman Publishers, San Fransisco, 1999.
32. 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.
33. Zaki M. Parallel and Distributed Association Mining: A Survey. *IEEE Concurrency*, 7(4):14–25, 1999.
34. 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.