Agent-Based Distributed Data Mining: The KDEC Scheme

Matthias Klusch[1], Stefano Lodi[2,3], and Gianluca Moro[2,4]

[1] Deduction and Multiagent Systems
German Research Centre for Artificial Intelligence
Stuhlsatzenhausweg 3, 66123 Saarbruecken, Germany
klusch@dfki.de
[2] Department of Electronics, Computer Science and Systems, University of Bologna
[3] CSITE-CNR, Viale Risorgimento, 2, I-40136 Bologna, Italy
[4] Via Rasi e Spinelli, 176, I-47023 Cesena (FC), Italy
{slodi,gmoro}@deis.unibo.it

Abstract. One key aspect of exploiting the huge amount of autonomous and heterogeneous data sources in the Internet is not only how to retrieve, collect and integrate relevant information but to discover previously unknown, implicit and valuable knowledge. In recent years several approaches to distributed data mining and knowledge discovery have been developed, but only a few of them make use of intelligent agents. This paper is intended to argue for the potential added value of using agent technology in the domain of knowledge discovery. We briefly review and classify existing approaches to agent-based distributed data mining, propose a novel approach to distributed data clustering based on density estimation, and discuss issues of its agent-oriented implementation.

1 Introduction

Mining information and knowledge from huge data sources such as weather databases[5], financial data portals[6], or emerging disease information systems[7] has been recognized by industrial companies as an important area with an opportunity of major revenues from applications such as business data warehousing, process control, and personalised on-line customer services over the Internet and Web. *Knowledge discovery* (KD) is a process aiming at the extraction of previously unknown and implicit knowledge out of large databases which may potentially be of added value for some given application [1]. Among the steps of the overall KD process including preparation of the data to be analysed as well as evaluation and visualisation of the discovered knowledge, *data mining* (DM) which is devoted to automated extraction of unknown patterns from given data is a central element. The large variety of DM techniques which have been developed over the past decade includes methods for pattern-based similarity search,

[5] http://www.noaa.gov
[6] http://www.nasdaq.com
[7] http://www.cdc.gov

clustering analysis, decision-tree based classification, generalization taking the data cube or attribute-oriented induction approach, and mining of association rules [2]

Most of the existing DM techniques were originally developed for centralized data and need to be modified for handling the distributed case. The increasing demand to scale up to massive data sets inherently distributed over a network with limited bandwidth and computational resources available motivated the development of methods for parallel (PKD) and distributed knowledge discovery (DKD) [3]. The related pattern extraction problem in DKD is referred to as *distributed data mining* (DDM). DDM is expected to perform partial analysis of data at individual sites and then to send the outcome as partial result to other sites where it is sometimes required to be aggregated to the global result. In fact, quite a number of DDM solutions are available using various techniques such as distributed association rules, distributed clustering, Bayesian learning, classification (regression), and compression, but only a few of them make use of intelligent agents at all.

One of the most widely used approach to DDM in business applications is to apply DM techniques to data which has been retrieved from different sources and stored in a central data warehouse. A *data warehouse* is a collection of integrated data from distributed data sources in a single repository [4]. However, despite its commercial success, the centralised application of data warehousing technology for DDM may be impractical or even impossible for some business settings in distributed environments. The main problems any approach to DDM is challenged to cope with concern issues of autonomy, privacy, and scalability. For example, when data can be viewed at the data warehouse from many different perspectives and at different levels of abstraction, it may threaten the goal of protecting individual data and guarding against invasion of privacy. Requirements to respect strict or a certain degree of autonomy of given data sources as well as privacy restrictions on individual data may make monolithic DM infeasible. Though DDM techniques are commonly considered as an important step towards reconciliation the opposing privacy concerns of centralised DM, that is protecting individual data and reducing unwanted privacy intrusions, research in the domain is still far from having resolved this problem of secure DDM in general.

Another problem arises with the need to scale up to massive data sets which are distributed over a large number of sites. For example, the NASA Earth Observing System[8] (EOS) is a data collector for satellites producing 1450 data sets of about 350GB per day and pair of satellites at a very high rate which are stored and managed by different systems geographically located all over the USA. Any online mining of such huge and distributed data sets in a central data warehouses may be prohibitively expensive in terms of costs of both communication and computation. To date, most work on DDM and PDM use distributed processing and the decomposability of data mining problems to scale up to large data sources. One lesson from the recent research work on DDM is that cooper-

[8] http://eospso.gsfc.nasa.gov/

ation among distributed DM processes may allow effective mining even without centralised control [5].

This in turn leads us to the question whether there is any real added value of using concepts from agent technology [6, 7] for the development of advanced DDM systems. In general, the inherent feature of software agents of being autonomous, capable of adaptive and deliberative reasoning seems to fit quite well with the requirements of coping with the above mentioned problems and challenges of DDM. Autonomous data mining agents as a special kind of information agents [6] may perform various kinds of mining operations on behalf of its user(s) or in collaboration with other agents. Systems of cooperative information agents for data mining tasks in distributed, hetergoeneous and massive data environments appear to be quite a natural vision for the near future to be realised.

In this paper we briefly review and classify existing DDM systems and frameworks according to some criteria in section 2. This is followed by a brief discussion on the benefits of using agents for DDM in section 3, and the proposal of a novel scheme, named KDEC, for agent-based distributed data clustering in sections 4 and 5. We compare our approach to other related work in section 6, and conclude the paper in section 7 with an outline of ongoing and future research work.

2 State of the Art

In this section we provide a brief review of the most prominent and representative agent-based DDM systems to date, that are BODHI, PADMA, JAM, and Papyrus, according to (a) the kind, type, and used means for security of data processed; (b) used DM techniques, implementation of the system and agents; and (c) the architecture with respect to the main coordination and control, execution of data processing, and transmission of agents, data, and models in due course of the DM tasks to be pursued by the system.

BODHI [3] has been designed according to a framework for collective DM tasks on heterogeneous data sites such as supervised inductive distributed function learning and regression. This framework guarantees correct local and global data model with low network communication load. BODHI is implemented in Java; it offers message exchange and runtime environments (agent stations) for the execution of mobile agents at each local site. The mining process is distributed to the local agent stations and agents that are moving between them on demand each carrying its state, data and knowledge. A central facilitator agent is responsible for initializing and coordinating DM tasks to be pursued within the system by the agents and agent stations, as well as the communication and control flow between the agents.

PADMA [8] deals with the problem of DDM from homogeneous data sites. Partial data cluster models are first computed by stationary agents locally at different sites. All local models are collected to a central site that peforms a second-level clustering algorithm to generate the global cluster model. Individual agents perform hierarchical clustering in text document classification, and web based information visualization.

JAM [9] is a Java-based multi-agent system designed to be used for meta-learning DDM. Different learning classifiers such as Ripper, CART, ID3, C4.5, Bayes, and WEPBLS can be executed on heterogeneous (relational) databases by any JAM agent that is either residing on one site or is being imported from other peer sites in the system. Each site agent builds a classification model and different agents build classifiers using different techniques. JAM also provides a set of meta-learning agents for combining multiple models learnt at different sites into a meta-classifier that in many cases improves the overall predictive accuracy. Once the combined classifiers are computed, the central JAM system coordinates the execution of these modules to classify data sets of interest at all data sites simultaneously and independently.

Papyrus [10] is a Java-based system addressing wide-area DDM over clusters of hetereogeneous data sites and meta-clusters. It supports different task and predictive model strategies including C4.5. Mobile DM agents move data, intermediate results, and models between clusters to perform all computation locally and reduce network load, or from local sites to a central root which produces the final result. Each cluster has one distinguished node which acts as its cluster access and control point for the agents. Coordination of the overall clustering task is either done by a central root site or distributed to the (peer-to-peer) network of cluster access points. Papyrus supports various methods for combining and exchanging the locally mined predictive models and metadata required to describe them by using a special markup language.

Common to all approaches is that they aim at integrating the knowledge which is discovered out of data at different geographically distributed network sites with a minimum amount of network communication, and maximum of local computation. Agent-based DDM solutions have been applied to both heterogeneous and homogeneous data sites in both multi-database and distributed database environments

3 Why Agents for DDM?

Looking at the state of the art of agent-based DDM systems presented in the previous section we may identify following arguments in favor or against the use of intelligent agents for distributed data mining.

Autonomy of data sources. A DM agent may be considered as a modular extension of a data management system to deliberatively handle the access to the underlying data source in accordance with given constraints on the required autonomy of the system, data and model. This is in full compliance with the paradigm of cooperative information systems [11].

Interactive DDM. Pro-actively assisting agents may drastically limit the amount a human user has to supervise and interfere with the running data mining process [12]. For example, DM agents may anticipate the individual limits of the potentially large search space and proper intermediate results particularly driven by their individual users' preferences with respect to the particular type of DM task at hand.

Dynamic selection of sources and data gathering. One challenge for intelligent DM agents acting in open distributed data environments in which, for example, the DM tasks to pursue, the availability of data sites and their content may change at any time, is to discover and select relevant sources. In such settings DM agents may be applied to adaptively select data sources according to given criterias such as the expected amount, type and quality of data at the considered source, actual network and DM server load [13]. Such DM agents may be used, for example, to dynamically control and manage the process of data gathering to support any OLAP (online analytical processing) and business data warehouse application.

Scalability of DM to massive distributed data. One option to reduce network and DM application server load may be to let DM agents migrate to each of the local data sites in a DDM system on which they may perform mining tasks locally, and then either return with or send relevant pre-selected data to their originating server for further processing. Experiments in using mobile information filtering agents in distributed data environments are encouraging [14].

Multi-strategy DDM. For some complex application settings an appropriate combination of multiple data mining technique may be more beneficial than applying just one particular one. DM agents may learn in due course of their deliberative actions which one to choose depending on the type of data retrieved from different sites and mining tasks to be pursued. The learning of multi-strategy selection of DM methods is similar to the adaptive selection of coordination strategies in a multi-agent system as proposed, for example, in [15].

Collaborative DM. DM agents may operate independently on data they have gathered at local sites, and then combine their respective models. Or they may agree to share potential knowledge as it is discovered, in order to benefit from the additional opinions of other DM agents. Meta-learning techniques may be used to perform mining homogeneous, distributed data. However, naive approaches to local data analysis may produce an ambiguous and incorrect global data model if different hetereogeneous data sites are involved which store data for different sets of features, possibly with some common features among the sites. Collaborative DM agents may negotiate among each other and jointly plan a solution for the above mentioned problems at hand. The need for DM agents to collaborate is prominent, for example, in cases where credit card frauds have to be detected by scanning, analysing, and partially integrating worldwidely distributed data records in different, autonomous sources. Other applications of potential added value include the pro-active re-collection of geographically distributed patient records and mining of the corresponding data space on demand to infer implicit knowledge to support an advanced treatment of patients no matter into which and how many hospitals they have been taken into in the past. However, frameworks for agent-based collective data mining such as BODHI are still more than rare to date.

Security and trustworthiness. In fact, this may be an argument against the use of agents for DDM. Of course, any agent-based DDM system has to cope with the problem of ensuring data security and privacy. However, any failure to implement least privilege at a data source, that means endowing subjects with only enough permissions to discharge their duties, could give any mining agent unsolicited access to sensitive data. Moreover, any mining operation performed by agents of a DDM system lacking a sound security architecture could be subject to eavesdropping, data tampering, or denial of service attacks. Agent code and data integrity is a crucial issue in secure DDM: Subverting or hijacking a DM agent places a trusted piece of (mobile) software - thus any sensitive data carried or transmitted by the agent - under the control of an intruder. In cases were DM agents are even allowed to migrate to remote computing environments of the distributed data sites of the DDM system methods to ensure confidentiality and integrity of a mobile agent have to be applied. Regarding agent availability there is certainly no way to prevent malicious hosts from simply blocking or destroying the temporarily residing DM agents but selective replication in a fault tolerant DDM agent architecture may help. In addition, data integration or aggregation in a DDM process introduces concern regarding inference attacks as a potential security threat. Data mining agents may infer sensitive information even from partial integration to a certain extent and with some probability. This problem, known as the so called inference problem, occurs especially in settings where agents may access data sources across trust boundaries which enable them to integrate implicit knowledge from different sources using commonly held rules of thumb. Not any of the existing DDM systems, agent-based or not, is capable of coping with this inference problem in the domain of secure DDM.

In the following sections we are investigating how agents may be used to perform a special kind of distributed data mining, that is clustering of data at different homogeneous data sites. For this purpose, we are presenting an approach to cluster analysis based on density estimation, adopting it to the distributed case, and then briefly discuss issues of implementing the resulting scheme for distributed data clustering in an agent-based DDM system.

4 Data Clustering

4.1 The cluster analysis problem

Cluster analysis is a a descriptive data mining task which aims at decomposing or partitioning a usually multivariate data set into groups such that the data objects in one group are similar to each other and are different as possible from those in other groups. Clustering techniques inherently hinge on the notion of distance between data objects to be grouped, and all we need to know is the set of interobject distances but not the values of any of the data object variables. Several techniques for data clustering are available but must be matched by the developer to the objectives of the considered clustering task [16]. In partition-based clustering, for example, the task is to partition a given data set into multiple disjoint sets of data objects such that the objects within each set are as homogeneous as possible. Homogeneity here is captured by an appropriate cluster scoring function. Another option bases on the intuition that homogeneity is expected to be high in densely populated regions of the given data set. Consequently, searching for clusters may be reduced to searching for dense regions of the data space which are more likely to be populated by data objects. That leads us to the approach of density estimation based clustering.

4.2 Density estimation based clustering

In *density estimation* (DE) based clustering the search for densely populated regions is accomplished by estimating a so-called probability density or cumulative distribution function from which the given data set is assumed to have arisen. Many techniques for DE-based clustering are available from the vast KDD literature [17, 18, 19, 20] and statistics [21]. In both areas, the proposed clustering methods require the computation of a non-parametric estimation of the density function from the data. One important family of non-parametric estimates is known as *kernel estimators*. The idea is to estimate a density function by defining the density at any data object as being proportional to a weighted sum of all objects in the data set, where the weights are defined by an appropriately chosen kernel function. In the following we introduce our approach to kernel-based density estimation.

Let us assume a set $S = \{\boldsymbol{x}_i \mid i = 1, \ldots, N\} \subseteq \mathbb{R}^n$ of data points or objects. Kernel estimators originate from the intuition that the higher the number of neighbouring data objects $\boldsymbol{x}_i$ of some given object $\boldsymbol{x} \in \mathbb{R}^n$, the higher the density at this object $\boldsymbol{x}$. However, there can be many ways of capturing and weighting the influence of data objects. When given the distance between one data object $\boldsymbol{x}$ and another $\boldsymbol{x}_i$ as an argument, the influence of $\boldsymbol{x}_i$ may be quantified by using a so called kernel function. A *kernel function* $K(x)$ is a real-valued, non-negative function on $\mathbb{R}$ which has finite integral over $\mathbb{R}$. When computing a kernel-based density estimation of the data set S, any element $\boldsymbol{x}_i$ in S is regarded as to exert more influence on some $\boldsymbol{x} \in \mathbb{R}^n$ than elements which are farther from $\boldsymbol{x}$ than the element. Accordingly, kernel functions are often non-increasing with $|x|$. Prominent examples of kernel functions are the square pulse function $\frac{1}{4}(sign(x+1) - sign(x-1))$, and the Gaussian function $\frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}x^2)$.

A *kernel-based density estimate* $\hat{\varphi}_{K,h}[S](\cdot)\colon \mathbb{R}^n \to \mathbb{R}_+$ is defined, modulo a normalization factor, as the sum over all data objects $\boldsymbol{x}_i$ in S of the distances between $\boldsymbol{x}_i$ and $\boldsymbol{x}$, scaled by a factor h, called *window width*, and weighted by the kernel function K:

$$\hat{\varphi}_{K,h}[S](\boldsymbol{x}) = \sum_{i=1}^{N} K\left(\frac{d(\boldsymbol{x}, \boldsymbol{x}_i)}{h}\right). \tag{1}$$

The influence of data objects and the smoothness of the estimate is controlled by both the window width h and the shape of kernel K: h controls the smoothness of the estimate, whereas K determines the decay of the influence of a data object according to the distance. Even if the number N of data objects is very large, in practice it is not necessary to compute N distances for calculating the kernel density estimate at a given object $\boldsymbol{x}$. In fact, the value of commonly used kernel

Algorithm 1 DE-cluster algorithm: Clustering based on density estimation

$\underline{\textbf{funct}}\ \Phi(\boldsymbol{x}, S, K, h, I) \ \equiv\ \sum_{\boldsymbol{x}_i \in I(\boldsymbol{x})} K\left(\dfrac{d(\boldsymbol{x}, \boldsymbol{x}_i)}{h}\right).$

$\underline{\textbf{funct}}\ \mathit{Uphill}(\boldsymbol{x}, S, K, h, I) \ \equiv\ \boldsymbol{x} + \delta \dfrac{\nabla\Phi(\boldsymbol{x}, S, K, h, I)}{\|\nabla\Phi(\boldsymbol{x}, S, K, h, I)\|}.$

$\underline{\textbf{proc}}\ \mathit{DensityCluster}(S[\,], K, h, I, \mathit{Cluster}[\,]) \ \equiv$

```
for i := 1 to Len(S) do
    x := ApproxFixedPoint(Uphill(x_i, S, K, h, I), ε)
    Cluster[i] := Nearest(x)
od.
```

functions is negligible for distances larger than a few h units; it may even be zero if the kernel has bounded support, as it is the case, for example, for the square pulse. Using kernel-based density estimation, it is straightforward to decompose the clustering problem into three phases as follows.

A. Choose a window width h and a kernel function K.
B. Compute the kernel-based density estimate $\hat{\varphi}_{K,h}[S](\boldsymbol{x})$ from the given data set.
C. Detect the regions of the data space for which the value of the computed estimate exceeds a given threshold and group all data objects of these regions into corresponding clusters.

In the literature, many different definitions of cluster have been proposed formalizing the clusters referred to in step C above. A *density-based* cluster [18] collects all data objects included in a region where density exceeds a threshold. *Center-defined* clusters [19] are based on the idea that every local maximum of $\hat{\varphi}$ corresponds to a cluster including all data objects which can be connected to the maximum by a continuous, uphill path in the graph of $\hat{\varphi}$. Finally, an *arbitrary-shape* cluster [19] is the union of center-defined clusters having their maxima connected by a continuous path whose density exceeds a threshold.

The DE-cluster Algorithm 1 essentially adopted from Hinneburg and Keim [19] implements the computation of center-defined clusters by an uphill climbing procedure driven by the density estimate. Function Φ approximates $\hat{\varphi}$ by restricting the set of data objects considered in the summation to a neighbourhood of $\boldsymbol{x}$. A function I specifies the neighbourhood. *Uphill* returns an object which is δ units away from $\boldsymbol{x}$ in the steepest direction of Φ, which is given by its gradient. The approximate fixed point function *ApproxFixedPoint* stops the computation of *Uphill* when the difference between two consecutive objects returned by *Uphill* is smaller than ϵ. Finally, the cluster index of object $\boldsymbol{x}_i$ is set as the index of the nearest neighbour in the data set S of the fixed object.

The complexity of the DE-cluster algorithm is that of computing the approximate fixed point of the hill-climbing function, and executing N times a nearest neighbour query. An efficient implementation approximates the gradient by $\sum_{\boldsymbol{x}_i \in I(\boldsymbol{x})} (\boldsymbol{x}_i - \boldsymbol{x}) K\left(\frac{d(\boldsymbol{x}_i, \boldsymbol{x})}{h}\right)$ (cf. [19]) and takes the nearest data object at every step of the climbing path. Then searching for a fixed object can be stopped whenever an already clustered data object is reached, since objects in the entire path can be labeled as this clustered object. The number of visited data objects is therefore N, and for each visit a k-nearest neighbour query is executed. The function $I(\cdot)$ can be implemented as such a query to a spatial access method like KD- or MVP-tree. In summary, the total cost is then $O(Nq(N))$, where $q(N)$ is the cost of a nearest neighbour query. In many practical cases, $q(N)$ is very close to $\log N$.

5 Distributed Data Clustering

The body of work on applications of data clustering in distributed environments, the problem of so called *distributed data clustering* (DDC), is comparatively small. In this section we adopt the kernel density estimation based clustering approach presented above for the distributed case assuming homogeneous data, which means that a data object cannot be split across two sites.

5.1 The DDC Problem

Let $\mathcal{A}(\cdot)$ be a clustering algorithm mapping any data set S to a *clustering* of S, that is, a collection of pairwise disjoint subsets of S. We define the problem of *homogeneous distributed data clustering* for clustering algorithm $\mathcal{A}$ as follows. Let $S = \{\boldsymbol{x}_i \mid i = 1, \ldots, N\} \subseteq \mathbb{R}^n$ be a data set of objects. Let $L_j, j = 1, \ldots, M$, be a finite set of *sites*. Each site L_j stores one data set D_j, and it will be assumed that $S = \bigcup_{j=1}^{M} D_j$. The DDC problem then is to find for $j = 1, \ldots, M$, a site clustering $\mathcal{C}_j$ residing in the data space of L_j, such that

(i). $\mathcal{C}_j = \{C \cap D_j : C \in \mathcal{A}(S)\}$ (*correctness requirement*)
(ii). Time and communications costs are minimized (*efficiency requirement*)
(iii). At the end of the computation, the size of the subset of S which has been transferred out of the data space of any site L_j is minimized (*privacy requirement*).

The traditional solution to the homogeneous distributed data clustering problem is to simply collect all the distributed data sets D_j into one centralized repository where their union S is computed, and the clustering $\mathcal{C}$ of the union S is computed and transmitted to the sites. Such approach, however, does not satisfy our problem's requirements both in terms of privacy and efficiency. We therefore propose a different approach yielding a kernel density estimation based clustering scheme, called KDEC, which may be implemented by appropriately designed DM agents of an agent-based DDM system.

5.2 The KDEC Scheme for DDC

The key idea of the KDEC scheme is based on the observation that the density estimate computed on each local data set conceals the details of the objects of the data set. Moreover, the local density estimate can be coded to provide a more compact representation of the data set for the purpose of transmission. In the sequel, we tacitly assume that all sites L_j agree on using a global kernel function K and a global window width h. We will therefore omit K and h from our notation, and write $\hat{\varphi}[S](\boldsymbol{x})$ for $\hat{\varphi}_{K,h}[S](\boldsymbol{x})$.

Density estimates in the form of Equation (1) are additive, i.e. the global density estimate $\hat{\varphi}[S](\boldsymbol{x})$ can be decomposed into the sum of the site density estimates, one estimate for every data set D_j:

$$\hat{\varphi}[S](\boldsymbol{x}) = \sum_{j=1}^{M} \sum_{\boldsymbol{x}_i \in D_j} K\left(\frac{d(\boldsymbol{x}, \boldsymbol{x}_i)}{h}\right) = \sum_{j=1}^{M} \hat{\varphi}[D_j](\boldsymbol{x}). \tag{2}$$

Thus, the local density estimates can be transmitted to and summed up at a distinguished *helper site* yielding the global estimate which can be returned to all sites. Each site L_j may then apply, in its local data space, the hill-climbing technique of the DE-cluster Algorithm 1 to assign clusters to the local data objects. There is nevertheless a weakness in such a plan: definition of a density estimate explicitly refers to all the data objects $\boldsymbol{x}_i$. Hence, knowing how to manipulate the estimate entails knowing the data objects, which contradicts the privacy requirement. However, only an intensional, algebraic definition of the estimate includes knowledge of the data objects. Multidimensional sampling provides an alternative extensional representation of the estimate which makes no explicit reference to the data objects.

Let $Diag[\boldsymbol{u}]$, $\boldsymbol{u} = [u_1, \ldots, u_n]^T \in \mathbb{R}^n$, denote the $n \times n$ diagonal matrix having diagonal $\boldsymbol{u}$, that is, the matrix $\{a_{ij}\}$ defined by

$$a_{ij} = \begin{cases} u_i & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Further let $\tau = [\tau_1, \ldots, \tau_n]^T \in \mathbb{R}^n$ a vector of *sampling periods*. The *sampled form* of $\hat{\varphi}[S](\boldsymbol{x})$ at intervals $\tau = [\tau_1, \ldots, \tau_n]^T$ is the sequence $\{(x[S])_z\}$, $z \in \mathbb{Z}^n$, defined by

$$(x[S])_z = \hat{\varphi}[S](Diag[z] \cdot \tau), \tag{3}$$

where $\cdot$ is the inner product between vectors. Therefore, $(x[S])_z$ is the sequence of the values of $\hat{\varphi}[S](\boldsymbol{x})$ computed at all the real, n-dimensional vectors whose ith coordinates are spaced by a multiple of the ith sampling period τ_i, $i = 1, \ldots, n$. The sampled forms of the local density estimates are defined in a similar way by

$$(x[D_j])_z = \hat{\varphi}[D_j](Diag[z] \cdot \tau) \qquad j = 1, \ldots, M. \tag{4}$$

It is immediate to see by (2) that addivity holds for the sampled forms:

$$(x[S])_z = \sum_{j=1}^{M} (x[D_j])_z \qquad j = 1, \ldots, M. \tag{5}$$

Therefore, after receiving the sampled forms $(x[D_j])_z$ of the M density estimates, the helper site, can compute by (5) the sampled form of the overall estimate and transmit it to the sites L_j. Sites L_j can then apply the hill-climbing procedure of the DE-cluster Algorithm 1 to cluster local data with respect to the overall density estimate, using the interpolation formula

$$\sum_{z \in \mathbb{Z}^n} (x[S])_z \; sinc(Diag[\tau]^{-1} \cdot (\boldsymbol{x} - Diag[z] \cdot \tau)) \tag{6}$$

where

$$sinc(\boldsymbol{x}) = \prod_{i=1}^{n} \frac{\sin x_i}{x_i}, \tag{7}$$

to compute the values of the overall estimate that are needed in the hill-climbing function.

We briefly discuss the extent to which function (6), which follows from Shannon's theorem, represents $\hat{\varphi}[S](\boldsymbol{x})$. Shannon's theorem asserts that sampling a function $g(\boldsymbol{x})$ is an invertible transformation if, for every coordinate $i = 1, \ldots, n$, there is a frequency f_i such that the absolute value of the Fourier transform (the *amplitude spectrum*) of g differs from zero only in the interval $[-f_i, f_i]$ and the samples are computed with a period not greater than $\tau_i = \frac{1}{2f_i}$. Under these assumptions, the value of the interpolation formula computed at $\boldsymbol{x}$ equals $g(\boldsymbol{x})$. Unfortunately, most popular kernel functions do not satisfy this constraint since there is no upper bound to the set of frequencies at which the amplitude spectrum is not zero. That, in turn, means that any summation of such kernels does not satisfy Shannon's theorem hypothesis either. Consequently Shannon's theorem does not apply to density estimates and sampling in such a case is a lossy transformation. However, it can be shown that the kernel density estimate's amplitude spectrum is negligible outside a neighbourhood of the origin of radius not greater than a few $\frac{1}{h}$ units. Therefore, the global density estimate can be reconstructed from its samples by (6) introducing only a small error. Besides, if the used kernel function has a bounded support the same goes with the density estimate. That means there are only finitely many values in the sampled form, thus the number of terms in summation (6) is finite. If, however, the kernel function has unbounded support, like the Gaussian kernel, then the density estimate can be approximated by setting its value to zero where it is less than a threshold $\epsilon_{\hat{\varphi}}$.

According to this approach, we propose the following algorithmic KDEC scheme for computing the kernel density estimation based clusters for local data spaces at M distributed data sites L_j (see Algorithm 2). Every local site runs the procedure *DataOwner* whereas the helper site runs *Helper*. *DataOwner* is passed a reference to the helper L and the local data set $D[\,]$, and returns a clustering vector *Cluster*$[\,]$. *Helper* is passed a list of references $L[\,]$ to the local sites. Procedure *Negotiate* carries out a negotiation with the other local sites through the helper site to determine the sampling periods τ, the boundaries of the sampling rectangle $z_1, z_2 \in \mathbb{Z}^n$, the kernel K and the window width h.

Algorithm 2 KDEC: distributed clustering based on density estimation

$\underline{\textbf{funct}}\ \Phi(\boldsymbol{x}, S, K, h, I) \ \equiv\ \sum_{\boldsymbol{x}_i \in I(\boldsymbol{x})} K\left(\dfrac{d(\boldsymbol{x}, \boldsymbol{x}_i)}{h}\right)$.

$\underline{\textbf{funct}}\ \mathit{Sample}(D, \tau, z_1, z_2, K, h, I) \ \equiv$

$\quad \{\Phi(\mathit{Diag}[z] \cdot \tau, D, K, h, I) : z_1 \leq z \leq z_2\}$.

$\underline{\textbf{funct}}\ \mathit{Reconstruct}(\boldsymbol{x}, \tau, z_1, z_2, \mathit{Sam}) \ \equiv$

$\quad \underline{\textbf{for}}\ z := z_1\ \underline{\textbf{to}}\ z_2\ \underline{\textbf{do}}$

$\qquad r := r + \mathit{Sam}[z]\, \mathit{Sinc}(\mathit{Diag}[\tau]^{-1} \cdot (\boldsymbol{x} - \mathit{Diag}[z] \cdot \tau))$

$\quad \underline{\textbf{od}}$;

$\quad r$.

$\underline{\textbf{funct}}\ \mathit{Uphill}(\boldsymbol{x}, \tau, z_1, z_2, \mathit{Sam}) \ \equiv$

$\quad \boldsymbol{x} + \delta \dfrac{\nabla \mathit{Reconstruct}(\boldsymbol{x}, \tau, z_1, z_2, \mathit{Sam})}{\|\nabla \mathit{Reconstruct}(\boldsymbol{x}, \tau, z_1, z_2, \mathit{Sam})\|}$.

$\underline{\textbf{proc}}\ \mathit{DataOwner}(D[\,], L, \mathit{Cluster}[\,]) \ \equiv$

$\quad \mathit{Negotiate}(L, \tau, z_1, z_2, K, h, I))$;

$\quad \mathit{Send}(\mathit{Sample}(D, \tau, z_1, z_2, K, h, I), L)$;

$\quad \mathit{Sam} := \mathit{Receive}(L)$;

$\quad \underline{\textbf{for}}\ i := 1\ \underline{\textbf{to}}\ \mathit{Len}(D)\ \underline{\textbf{do}}$

$\qquad \boldsymbol{x} := \mathit{ApproxFixedPoint}\big(\mathit{Uphill}(\boldsymbol{x}, \tau, z_1, z_2, \mathit{Sam}), \epsilon\big)$;

$\qquad \mathit{Cluster}[i] := \mathit{Nearest}(\boldsymbol{x})$;

$\quad \underline{\textbf{od}}$.

$\underline{\textbf{proc}}\ \mathit{Helper}(L[\,]) \ \equiv$

$\quad \mathit{Negotiate}(L)$;

$\quad \boldsymbol{X} := \boldsymbol{0}$;

$\quad \underline{\textbf{for}}\ j := 1\ \underline{\textbf{to}}\ \mathit{Len}(L)\ \underline{\textbf{do}}\ \boldsymbol{X} := \boldsymbol{X} + \mathit{Receive}(L[j])\ \underline{\textbf{od}}$;

$\quad \underline{\textbf{for}}\ j := 1\ \underline{\textbf{to}}\ \mathit{Len}(L)\ \underline{\textbf{do}}\ \mathit{Send}(\boldsymbol{X}, L[j])\ \underline{\textbf{od}}$.

Negotiate is run by the local sites and the helper site, and contains appropriate handshaking primitives to ensure that all sites participate and exit *Negotiate* only if an agreement has been reached. Each local site computes the sampled form of the estimate of $D[\,]$, and sends it to the helper. The helper receives the lists of sampled estimates, sums them by order in the lists to compile a list of global sample needed to compute the global estimate, and returns this list to all sites. Procedures *Send* and *Receive* implement appropriate blocking and handshaking to ensure the transmission takes place. Each local site uses the global sample in procedure *Reconstruct* to compute the values of the global density estimate and performs a DE-cluster algorithm to compute the corresponding local data clusters.

5.3 Complexity of the KDEC scheme

In terms of the complexity in computation and communication one crucial point of the KDEC scheme is how many samples have to be computed and transferred among the sites. Since the largest frequency at which the absolute value of the Fourier transform of the kernel estimate is not negligible depends only on the window width h, and is not greater than a few $\frac{1}{h}$ units, at any site L_j, the number of samples to be computed at L_j is only a fraction of the number of data objects. The computational costs of the KDEC scheme in terms of used CPU cycles and I/O do not exceed the one in the centralized approach where clustering is performed on data collected in a single repository. The computational complexity is linear in the number of samples. Of course, the precise cost of computation of any KDEC-based DDC algorithm as an instance of the proposed scheme largely depends also on the used kernel function and local clustering algorithm. The DE-cluster algorithm we developed for the KDEC scheme in Section 4.2 is of complexity $O(Nq(N))$, where $q(N)$ is the cost of a nearest neighbour query (which in practical cases is close to $\log N$). Since the size of samples is usually much smaller than that of the corresponding data set, and the cost of communicating samples of local kernel density estimates among the local sites and the helper site is linear, the overall communication costs of our DDC approach in any case will be significantly lower than in a centralized approach.

5.4 Agent Technology for KDEC-based DDC

We assume a DDM system consisting of a set of networked, homogeneous data sites. Each data site respects its local autonomy by individually granting read-only access to internal data mining agents.

One option of agent-based implementation of the KDEC based data clustering is to implement a set of appropriate stationary DM agents, each of which is associated to one of multiple, networked local data sites (Figure 1). According to the KDEC scheme, an agent takes the role of the helper and it engages the other site agents in a negotiation (Figure 2) to agree (a) what kernel function to use for computing the local density estimate samples, and (b) the use of the DE-cluster algorithm for local clustering based on the global estimate.

The diagram in Figure 2 is a possible implementation of interactions between agents according to the communicative acts of ACL FIPA. The interactions are based on the primitives of such a standard (e.g. *call-for-proposal, accept, inform...*) which give a first level semantics to message exchanged between agents.

Another option bases on the idea that under certain constraints a mobile helper agent can perform the KDEC based DDC computation in the overall DDM system (Figure 3). In this case, the helper agent moves through a sequence $\{L_n\}$ of data sites. At each site n the agent engages the agent site, which guarantees local data access services, in the initial negotiation as depicted in Figure 4; then the visiting agent—which carries in its data space, for every sampling point, the sum of all samples of every site L_m, $(m < n)$, at that point—(a) computes the sampled form of the density estimate of local data by means of the agent site and sums the local samples, and then (b) at the end of the trip returns the global estimate's sample to agent sites interested in the cluster analysis (see the *inform* message at the end of the interaction diagram in Figure 4).

Please note that the role of the central helper site in the KDEC scheme is taken by the mobile DM agent. This agent may be initiated either by some distinguished central site, or by any of the local sites which then is in charge of coordinating the actions of its mobile agent together with its stationery agent.

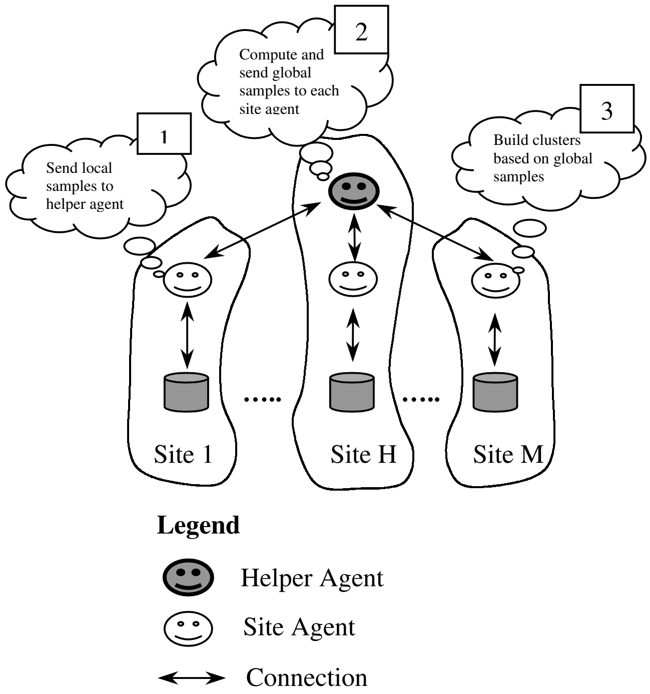

Fig. 1. Outline of agent-based implementation of the KDEC scheme for DDC

In any case, an appropriate mobile agent infrastructure, a cooperation protocol between the mobile agent and site agents, as well as a proper security architecture have to be implemented in such a system, mostly to avoid unauthorized data access and privacy violations.

As an application consider the task of market segmentation in a company structured into one headquarter and a large number of geographically distributed branches. The problem is how to effectively and efficiently partition the company's customers into classes depending on their properties in order to implement the same strategy for a given class across all branches. For this purpose, the company may launch a mobile agent which collaborates with each site agent for clustering all customers using the KDEC scheme with respect to one global set of cluster identifiers.

6 Related work

Only a few approaches to solve the problem of distributed data clustering are available to date. In [22] a tree clustering approach is taken to build a global dendrogram from individual dendrograms that are computed at local data sites subject to a given set of requirements. In contrast to the KDEC based DDC

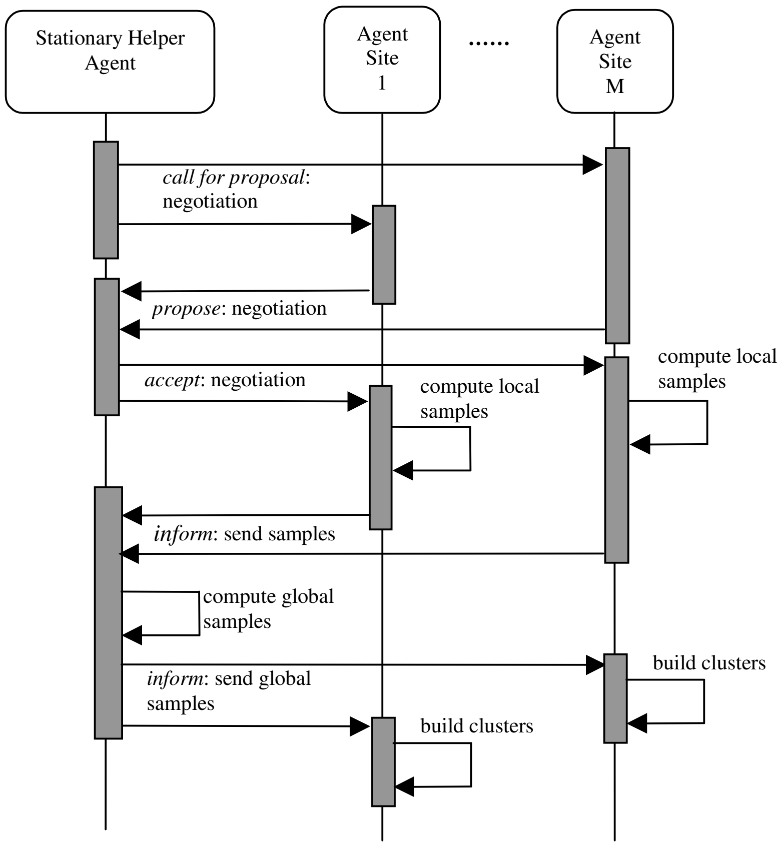

Fig. 2. Inter-agent interactions of the stationary agent-based implementation of the KDEC scheme for DDC

approach the distributed data sets are assumed to be heterogeneous, therefore every site has access only to a subset of the features of an object. The proposed solution implements a distributed version of the single link clustering algorithm which generates clusters that are substantially different from the ones generated by density-based methods like the KDEC scheme. In particular, it suffers from the so-called chaining effect, by which any of two homogeneous and well separated groups of objects connected only by a dense sequence of objects are regarded as a single cluster. [23] proposes a technique for distributed principal component analysis, Collective PCA. It is shown that the technique satisfies efficiency and data security requirements and can be integrated with existing clustering methods in order to cluster distributed, high-dimensional heterogeneous data. Since the dimensionality of the data is reduced prior to clustering by applying PCA, the approach is orthogonal to ours.

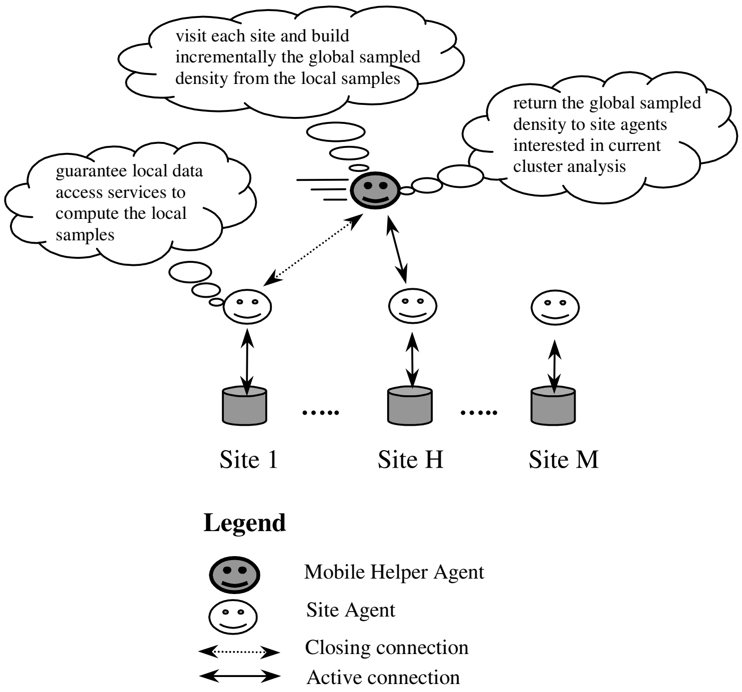

Fig. 3. Outline of the mobile agent-based implementation of the KDEC scheme for DDC

Another related research direction concerns incremental clustering algorithms. The BIRCH [24] and related BUBBLE method [25], compute the most accurate clustering, given the amount of memory available, while minimizing the number of I/O operations. It uses a dynamic index structure of nodes that store synthetic, constant-time maintainable summaries of sets of data objects. The method is sufficiently scalable requiring $O(N \log N)$ time and linear I/O. However, since it uses the centroid to incrementally aggregate object, the method exhibits a strong bias towards globular clusters. IncrementalDBSCAN [26] is a dynamic clustering method supporting both insertions and deletions, which is shown to be equivalent to the well-known static DBSCAN algorithm. Since in turn DBSCAN can be shown to be equivalent to a method based on density estimation when the kernel function is the square pulse and the clusters are density-based, IncrementalDBSCAN is less general than methods based on kernel density estimates. Its time complexity is $O(N \log N)$.

It is worth noting that, in an agent-based framework, incremental clustering techniques are potentially useful to efficiently support the incremental updates a clustering model undergoes as an agent visits data sites. However, the latter two approaches require agents to access all data available thereby violating the data privacy requirement. Our KDEC scheme shows more flexibility in that it

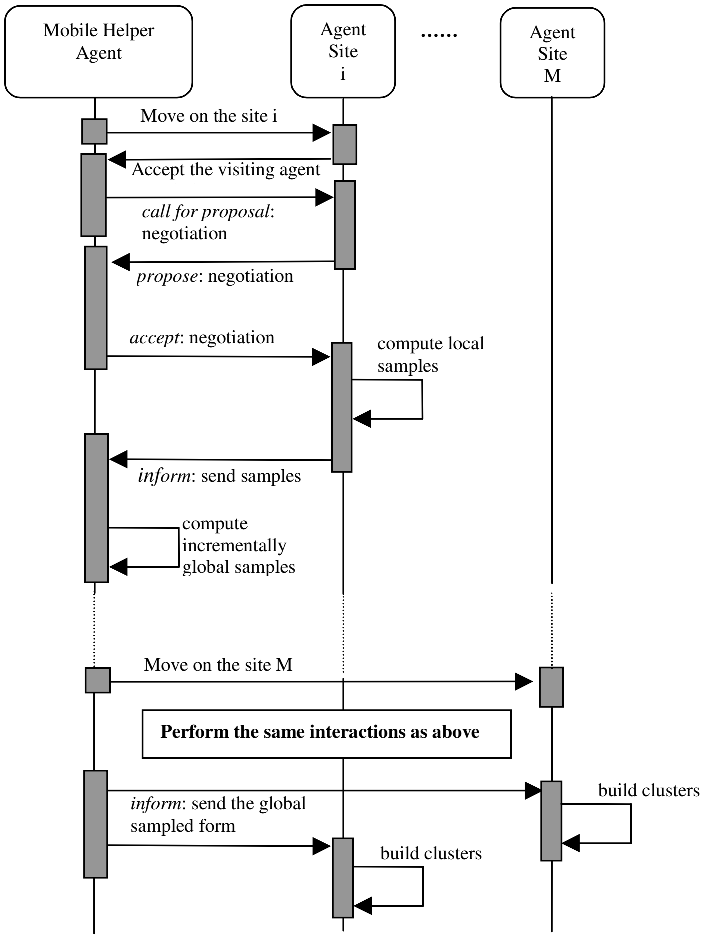

Fig. 4. Inter-agent interactions of the mobile agent-based implementation

works for any implementation where agents may be granted read-only access by autonomous data sites. However, like all other existing DDM systems the KDEC scheme fails to solve the inference problem mentioned in Section 3. In fact, from a (sampled) density estimate one can easily generate data sets which look like the original one, though only the helper site agent can do this separately for the local data sets, and the actual data objects can never be identified.

7 Conclusion

Due to the explosion in the number of autonomous data sources there is a growing need for effective approaches to distributed agent-based knowledge discovery and data mining. In this paper, we have reviewed prominent approaches in the literature and presented a novel scheme for agent-based distributed data clustering. The approach exploits statistical density estimation and information theoretic sampling to minimize communications between sites. Moreover, the privacy of data is preserved to a large extent by never transmitting data values but kernel based density estimation samples outside the site of origin. The approach does not require CPU and I/O costs significantly higher than a similar centralized approach and its communication costs may be lower. Ongoing research focus in particular on implementations of a multiagent system for KDEC-based DDC in a peer-to-peer network, and investigation on methods to mitigate the risk of security and privacy violations in DDM environments.

Acknowledgments

This work was partially supported by CSITE-CNR, project D2I of the Ministry of Education, University and Scientific Research of Italy, and project AGRICOLA under grant 032002 of the Saarland Ministry of Economics, Germany. The authors are also thankful to Claudio Sartori for valuable discussions on the subject of the paper.

References

[1] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R.: Advances in Knowledge Discovery and Data Mining. AAAI Press/MIT Press (1996)

[2] Chen, M.S., Han, J., Yu, P.S.: Data mining: an overview from a database perspective. IEEE Trans. On Knowledge And Data Engineering **8** (1996) 866–883

[3] Kargupta, H., Park, B., Hershberger, D., Johnson, E.: 5, Collective Data Mining: A New Perspective Toward Distributed Data Mining. In: Advances in Distributed and Parallel Knowledge Discovery. AAAI/MIT Press (2000) 131–178

[4] Moro, G., Sartori, C.: Incremental maintenance of multi-source views. In: M. E. Orlowska, J. Roddick Edition, Proceedings of 12th Australasian Database Conference, ADC 2001, Brisbane, Queensland, Australia, IEEE Computer Society (2001) 13–20

[5] Dhar, V., Chou, D., Provost, F.J.: Discovering interesting patterns for investment decision making with glower - a genetic learner. Data Mining and Knowledge Discovery **4** (2000) 251–280

[6] Klusch, M.: Information agent technology for the internet: A survey. Data and Knowledge Engineering, Special Issue on Intelligent Information Integration. Elsevier Science **36** (2001) 337–372

[7] Wooldridge, M.: Intelligent agents: The key concepts. In Marík, V., Stepánková, O., Krautwurmova, H., Luck, M., eds.: Multi-Agent-Systems and Applications. Volume 2322 of LNCS., Springer-Verlag (2002) 3–43

[8] Kargupta, H., Hamzaoglu, I., Stafford, B.: Scalable, distributed data mining using an agent-based architecture. In D.Heckerman, H.Mannila, D.Pregibon, R.Uthurusamy, eds.: Proc. 3rd International Conference on Knowledge Discovery and Data Mining, Newport Beach, California, USA, AAAI Press (1997) 211–214

[9] Stolfo, S.J., Prodromidis, A.L., Tselepis, S., Lee, W., Fan, D.W., Chan, P.K.: JAM: Java agents for meta-learning over distributed databases. In David Heckerman, Heikki Mannila, D.P., ed.: Proc. Third International Conference on Knowledge Discovery and Data Mining (KDD-97), Newport Beach, California, USA, AAAI Press (1997) 74–81

[10] Bailey, S., Grossman, R., Sivakumar, H., Turinsky, A.: Papyrus: a system for data mining over local and wide area clusters and super-clusters. In: Proc. Conference on Supercomputing, ACM Press (1999) 63

[11] Papazoglou, M.P., Schlageter, G.: Cooperative Information Systems - Trends and Directions. Academic Press Ltd, London, UK (1998)

[12] Zhong, N., Matsui, Y., Okuno, T., Liu, C.: Framework of a multi-agent kdd system. In Yin, H., Allinson, N.M., Freeman, R., Keane, J.A., Hubbard, S.J., eds.: Proc. of Intelligent Data Engineering and Automated Learning - IDEAL 2002, Third International Conference, Manchester, UK. Volume 2412 of Lecture Notes in Computer Science., Springer-Verlag (2002) 337–346

[13] Sen, S., Biswas, A., Gosh, S.: Adaptive choice of information sources. In: Proc. 3rd International workshop on Cooperative Information Agents, Springer (1999)

[14] Theilmann, W., Rothermel, K.: Disseminating mobile agents for distributed information filtering. In: Proc. of 1st International Sympos. on Mobile Agents, IEEE Press (1999) 152–161

[15] Prasad, M., Lesser, V.: Learning situation-specific coordinating in cooperative multi-agent systems. Autonomous Agents and Multi-Agent Systems (1999)

[16] Grabmeier, J., Rudolph, A.: Techniques of cluster algorithms in data mining. Data Mining and Knowledge Discovery **6** (2002) 303–360

[17] Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: Ordering points to identify the clustering structure. In: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, Philadephia, PA (1999) 49–60

[18] Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, OR (1996) 226–231

[19] Hinneburg, A., Keim, D.A.: An efficient approach to clustering in large multimedia databases with noise. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), New York City, New York, USA, AAAI Press (1998) 58–65

[20] Schikuta, E.: Grid-clustering: An efficient hierarchical clustering method for very large data sets. In: Proceedings of the 13th International Conference on Pattern Recognition, IEEE (1996) 101–105

[21] Silverman, B.W.: Density Estimation for Statistics and Data Analysis. Chapman and Hall, London (1986)

[22] Johnson, E., Kargupta, H.: Collective, hierarchical clustering from distributed heterogeneous data. In Zaki, M., Ho, C., eds.: Large-Scale Parallel KDD Systems. Lecture Notes in Computer Science. Springer-Verlag (1999) 221–244

[23] Kargupta, H., Huang, W., Krishnamoorthy, S., Johnson, E.: Distributed clustering using collective principal component analysis. Knowledge and Information Systems Journal **3** (2000) 422–448

[24] Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: An efficient data clustering method for very large darabases. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Canada (1996) 103–114

[25] Ganti, V., Ramakrishnan, R., Gehrke, J., Powell, A.L., French, J.C.: Clustering large datasets in arbitrary metric spaces. In: Proceedings of the 15th International Conference on Data Engineering (ICDE 1999), Sydney, Austrialia (1999) 502–511

[26] Ester, M., Kriegel, H.P., Sander, J., Wimmer, M., Xu, X.: Incremental clustering for mining in a data warehousing environment. In: Proceedings of the 24th International Conference on Very Large Data Bases (VLDB'98), New York City, NY (1998) 323–333