# DBDC: Density Based Distributed Clustering

Eshref Januzaj, Hans-Peter Kriegel, Martin Pfeifle

University of Munich, Institute for Computer Science
http://www.dbs.informatik.uni-muenchen.de
{januzaj,kriegel,pfeifle}@informatik.uni-muenchen.de

**Abstract.** Clustering has become an increasingly important task in modern application domains such as marketing and purchasing assistance, multimedia, molecular biology as well as many others. In most of these areas, the data are originally collected at different sites. In order to extract information from these data, they are merged at a central site and then clustered. In this paper, we propose a different approach. We cluster the data locally and extract suitable representatives from these clusters. These representatives are sent to a global server site where we restore the complete clustering based on the local representatives. This approach is very efficient, because the local clustering can be carried out quickly and independently from each other. Furthermore, we have low transmission cost, as the number of transmitted representatives is much smaller than the cardinality of the complete data set. Based on this small number of representatives, the global clustering can be done very efficiently. For both the local and the global clustering, we use a density based clustering algorithm. The combination of both the local and the global clustering forms our new DBDC (Density Based Distributed Clustering) algorithm. Furthermore, we discuss the complex problem of finding a suitable quality measure for evaluating distributed clusterings. We introduce two quality criteria which are compared to each other and which allow us to evaluate the quality of our DBDC algorithm. In our experimental evaluation, we will show that we do not have to sacrifice clustering quality in order to gain an efficiency advantage when using our distributed clustering approach.

## 1 Introduction

Knowledge Discovery in Databases (KDD) tries to identify valid, novel, potentially useful, and ultimately understandable patterns in data. Traditional KDD applications require full access to the data which is going to be analyzed. All data has to be located at that site where it is scrutinized. Nowadays, large amounts of heterogeneous, complex data reside on different, independently working computers which are connected to each other via local or wide area networks (LANs or WANs). Examples comprise distributed mobile networks, sensor networks or supermarket chains where check-out scanners, located at different stores, gather data unremittingly. Furthermore, international companies such as DaimlerChrysler have some data which is located in Europe and some data in the US. Those companies have various reasons why the data cannot be transmitted to a central site, e.g. limited bandwidth or security aspects.

The transmission of huge amounts of data from one site to another central site is in some application areas almost impossible. In astronomy, for instance, there exist several highly sophisticated space telescopes spread all over the world. These telescopes gather data un-

ceasingly. Each of them is able to collect 1GB of data per hour [10] which can only, with great difficulty, be transmitted to a central site to be analyzed centrally there. On the other hand, it is possible to analyze the data locally where it has been generated and stored. Aggregated information of this locally analyzed data can then be sent to a central site where the information of different local sites are combined and analyzed. The result of the central analysis may be returned to the local sites, so that the local sites are able to put their data into a global context.

The requirement to extract knowledge from distributed data, without a prior unification of the data, created the rather new research area of Distributed Knowledge Discovery in Databases (DKDD). In this paper, we will present an approach where we first cluster the data locally. Then we extract aggregated information about the locally created clusters and send this information to a central site. The transmission costs are minimal as the representatives are only a fraction of the original data. On the central site we "reconstruct" a global clustering based on the representatives and send the result back to the local sites. The local sites update their clustering based on the global model, e.g. merge two local clusters to one or assign local noise to global clusters.

The paper is organized as follows, in Section 2, we shortly review related work in the area of clustering. In Section 3, we present a general overview of our distributed clustering algorithm, before we go into more detail in the following sections. In Section 4, we describe our local density based clustering algorithm. In Section 5, we discuss how we can represent a local clustering by relatively little information. In Section 6, we describe how we can restore a global clustering based on the information transmitted from the local sites. Section 7 covers the problem how the local sites update their clustering based on the global clustering information. In Section 8, we introduce two quality criteria which allow us to evaluate our new efficient *DBDC* (Density Based Distributed Clustering) approach. In Section 9, we present the experimental evaluation of the *DBDC* approach and show that its use does not suffer from a deterioration of quality. We conclude the paper in Section 10.

## 2 Related Work

In this section, we first review and classify the most common clustering algorithms. In Section 2.2, we shortly look at parallel clustering which has some affinity to distributed clustering.

### 2.1 Clustering

Given a set of objects with a distance function on them (i.e. a feature database), an interesting data mining question is, whether these objects naturally form groups (called clusters) and what these groups look like. Data mining algorithms that try to answer this question are called *clustering algorithms*. In this section, we classify well-known clustering algorithms according to different categorization schemes.

Clustering algorithms can be classified along different, independent dimensions. One well-known dimension categorizes clustering methods according to the *result* they produce. Here, we can distinguish between *hierarchical* and *partitioning clustering* algorithms [13, 15]. Partitioning algorithms construct a flat (single level) partition of a database $D$ of $n$ objects into a set of $k$ clusters such that the objects in a cluster are more similar to each other

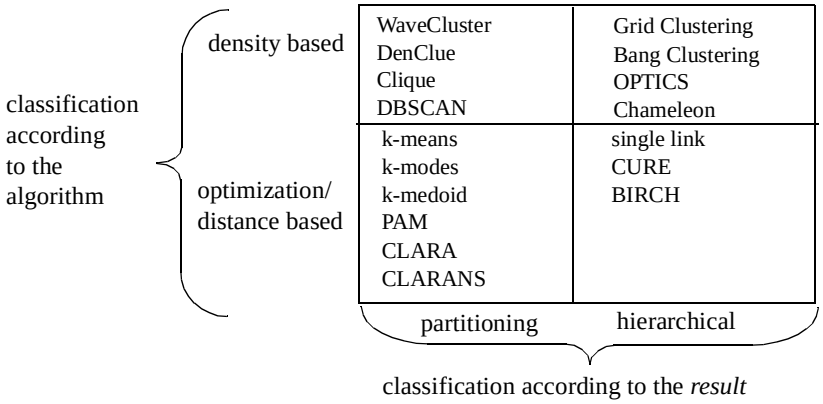


**Fig. 1.** Classification scheme for clustering algorithms

than to objects in different clusters. Hierarchical algorithms decompose the database into several levels of nested partitionings (clusterings), represented for example by a dendrogram, i.e. a tree that iteratively splits *D* into smaller subsets until each subset consists of only one object. In such a hierarchy, each node of the tree represents a cluster of *D*.

Another dimension according to which we can classify clustering algorithms is from an *algorithmic* point of view. Here we can distinguish between *optimization based or distance based* algorithms and *density based* algorithms. Distance based methods use the distances between the objects directly in order to optimize a global cluster criterion. In contrast, density based algorithms apply a local cluster criterion. Clusters are regarded as regions in the data space in which the objects are dense, and which are separated by regions of low object density (noise).

An overview of this classification scheme together with a number of important clustering algorithms is given in Figure 1. As we do not have the space to cover them here, we refer the interested reader to [15] were an excellent overview and further references can be found.

## 2.2 Parallel Clustering and Distributed Clustering

Distributed Data Mining (DDM) is a dynamically growing area within the broader field of KDD. Generally, many algorithms for distributed data mining are based on algorithms which were originally developed for parallel data mining. In [16] some state-of-the-art research results related to DDM are resumed.

Whereas there already exist algorithms for distributed and parallel classification and association rules [2, 12, 17, 18, 20, 22], there do not exist many algorithms for parallel and distributed clustering.

In [9] the authors sketched a technique for parallelizing a family of center-based data clustering algorithms. They indicated that it can be more cost effective to cluster the data in-place using an exact distributed algorithm than to collect the data in one central location for clustering. In [14] the “collective hierarchical clustering algorithm” for vertically distributed data sets was proposed which applies single link clustering. In contrast to this approach, we concentrate in this paper on horizontally distributed data sets and apply a

partitioning clustering. In [19] the authors focus on the reduction of the communication cost by using traditional hierarchical clustering algorithms for massive distributed data sets. They developed a technique for centroid-based hierarchical clustering for high dimensional, horizontally distributed data sets by merging clustering hierarchies generated locally. In contrast, this paper concentrates on density based partitioning clustering.

In [21] a parallel version of DBSCAN [7] and in [5] a parallel version of k-means [11] were introduced. Both algorithms start with the complete data set residing on one central server and then distribute the data among the different clients.

The algorithm presented in [5] distributes $N$ objects onto $P$ processors. Furthermore, $k$ initial centroids are determined which are distributed onto the $P$ processors. Each processor assigns each of its objects to one of the $k$ centroids. Afterwards, the global centroids are updated (reduction operation). This process is carried out repeatedly until the centroids do not change any more. Furthermore, this approach suffers from the general shortcoming of k-means, where the number of clusters has to be defined by the user and is not determined automatically.

The authors in [21] tackled these problems and presented a parallel version of DBDSAN. They used a 'shared nothing'-architecture, where several processors where connected to each other. The basic data-structure was the dR*-tree, a modification of the R*-tree [3]. The dR*-tree is a distributed index-structure where the objects reside on various machines. By using the information stored in the dR*-tree, each local site has access to the data residing on different computers. Similar, to parallel k-means, the different computers communicate via message-passing.

In this paper, we propose a different approach for distributed clustering assuming we cannot carry out a preprocessing step on the server site as the data is not centrally available. Furthermore, we abstain from an additional communication between the various client sites as we assume that they are independent from each other.

## 3 Density Based Distributed Clustering

Distributed Clustering assumes that the objects to be clustered reside on different sites. Instead of transmitting all objects to a central site (also denoted as server) where we can apply standard clustering algorithms to analyze the data, the data are clustered independently on the different local sites (also denoted as clients). In a subsequent step, the central site tries to establish a global clustering based on the local models, i.e. the representatives. This is a very difficult step as there might exist dependencies between objects located on different sites which are not taken into consideration by the creation of the local models. In contrast to a central clustering of the complete dataset, the central clustering of the local models can be carried out much faster.

Distributed Clustering is carried out on two different levels, i.e. the local level and the global level (cf. Figure 2). On the local level, all sites carry out a clustering independently from each other. After having completed the clustering, a local model is determined which should reflect an optimum trade-off between complexity and accuracy. Our proposed local models consist of a set of representatives for each locally found cluster. Each representative is a concrete object from the objects stored on the local site. Furthermore, we augment each representative with a suitable $\varepsilon$–range value. Thus, a representative is a good approximation

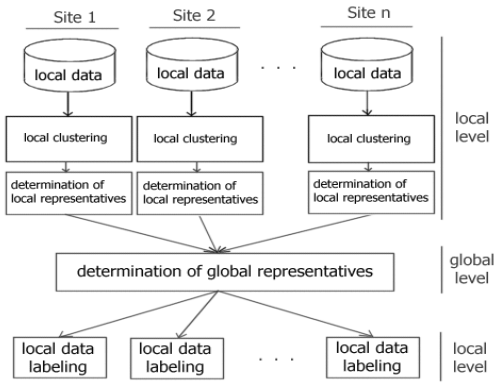


**Fig. 2.** Distributed Clustering

for all objects residing on the corresponding local site which are contained in the ε–range around this representative.

Next the local model is transferred to a central site, where the local models are merged in order to form a global model. The global model is created by analyzing the local representatives. This analysis is similar to a new clustering of the representatives with suitable global clustering parameters. To each local representative a global cluster-identifier is assigned. This resulting global clustering is sent to all local sites.

If a local object belongs to the ε-neighborhood of a global representative, the cluster-identifier from this representative is assigned to the local object. Thus, we can achieve that each site has the same information as if their data were clustered on a global site, together with the data of all the other sites.

To sum up, distributed clustering consists of four different steps (cf. Figure 2):

- Local clustering
- Determination of a local model
- Determination of a global model, which is based on all local models
- Updating of all local models

## 4 Local Clustering

As the data are created and located at local sites we cluster them there. The remaining question is “which clustering algorithm should we apply”. K-means [11] is one of the most commonly used clustering algorithms, but it does not perform well on data with outliers or with clusters of different sizes or non-globular shapes [8]. The single link agglomerative clustering method is suitable for capturing clusters with non-globular shapes, but this approach is very sensitive to noise and cannot handle clusters of varying density [8]. We used the density-based clustering algorithm DBSCAN [7], because it yields the following advantages:

- DBSCAN is rather robust concerning outliers.
- DBSCAN can be used for all kinds of metric data spaces and is not confined to vector spaces.
- DBSCAN is a very efficient and effective clustering algorithm.

- There exists an efficient incremental version, which would allow incremental clusterings on the local sites. Thus, only if the local clustering changes "considerably", we have to transmit a new local model to the central site [6].

We slightly enhanced DBSCAN so that we can easily determine the local model after we have finished the local clustering. All information which is comprised within the local model, i.e. the representatives and their corresponding ε–ranges, is computed on-the-fly during the DBSCAN run.

In the following, we describe DBSCAN in a level of detail which is indispensable for understanding the process of extracting suitable representatives (cf. Section 5).

### 4.1 The Density-Based Partitioning Clustering-Algorithm DBSCAN

The key idea of density-based clustering is that for each object of a cluster the neighborhood of a given radius (*Eps*) has to contain at least a minimum number of objects (*MinPts*), i.e. the cardinality of the neighborhood has to exceed some threshold. Density-based clusters can also be significantly generalized to density-connected sets. Density-connected sets are defined along the same lines as density-based clusters.

We will first give a short introduction to DBSCAN. For a detailed presentation of DBSCAN see [7].

**Definition 1 (directly density-reachable).** An object $p$ is *directly density-reachable* from an object $q$ wrt. *Eps* and *MinPts* in the set of objects $D$ if

- $p \in N_{Eps}(q)$ ($N_{Eps}(q)$ is the subset of $D$ contained in the *Eps*-neighborhood of $q$)
- $|N_{Eps}(q)| \geq MinPts$ (core-object condition)

**Definition 2 (density-reachable).** An object $p$ is *density-reachable* from an object $q$ wrt. *Eps* and *MinPts* in the set of objects $D$, denoted as $p >_D q$, if there is a chain of objects $p_1, ..., p_n, p_1 = q, p_n = p$ such that $p_i \in D$ and $p_{i+1}$ is directly density-reachable from $p_i$ wrt. *Eps* and *MinPts.*

Density-reachability is a canonical extension of direct density-reachability. This relation is transitive, but it is not symmetric. Although not symmetric in general, it is obvious that density-reachability is symmetric for objects $o$ with $|N_{Eps}(o)| \geq MinPts$. Two "border objects" of a cluster are possibly not density-reachable from each other because there are not enough objects in their *Eps*-neighborhoods. However, there must be a third object in the cluster from which both "border objects" are density-reachable. Therefore, we introduce the notion of density-connectivity.

**Definition 3 (density-connected).** An object $p$ is *density-connected* to an object $q$ wrt. *Eps* and *MinPts* in the set of objects $D$ if there is an object $o \in D$ such that both, $p$ and $q$ are density-reachable from $o$ wrt. *Eps* and *MinPts* in $D$.

Density-connectivity is a symmetric relation. A *cluster* is defined as a set of density-connected objects which is maximal wrt. density-reachability and the *noise* is the set of objects not contained in any cluster.

**Definition 4 (cluster).** Let $D$ be a set of objects. A *cluster* $C$ wrt. *Eps* and *MinPts* in $D$ is a non-empty subset of $D$ satisfying the following conditions:

- Maximality: $\forall p,q \in D$: if $p \in C$ and $q >_D p$ wrt. *Eps* and *MinPts*, then also $q \in C$.
- Connectivity: $\forall p,q \in C$: $p$ is density-connected to $q$ wrt. *Eps* and *MinPts* in $D$.

**Definition 5 (noise).** Let $C_1,\ldots, C_k$ be the clusters wrt. *Eps* and *MinPts* in *D*. Then, we define the *noise* as the set of objects in the database *D* not belonging to any cluster $C_i$, i.e. *noise* = $\{p \in D \mid \forall$ i: p $\notin C_i\}$.

We omit the term "wrt. *Eps* and *MinPts"* in the following whenever it is clear from the context. There are different kinds of objects in a clustering: *core objects* (satisfying condition 2 of definition 1) or *non-core objects* otherwise. In the following, we will refer to this characteristic of an object as the *core object property* of the object. The non-core objects in turn are either *border objects* (no core object but density-reachable from another core object) or *noise objects* (no core object and not density-reachable from other objects).

The algorithm DBSCAN was designed to efficiently discover the clusters and the noise in a database according to the above definitions. The procedure for finding a cluster is based on the fact that a cluster as defined is uniquely determined by any of its core objects: first, given an arbitrary object *p* for which the core object condition holds, the set $\{o \mid o >_D p\}$ of all objects *o* density-reachable from *p* in *D* forms a complete cluster *C*. Second, given a cluster *C* and an arbitrary core object $p \in C$, *C* in turn equals the set $\{o \mid o >_D p\}$ (c.f. lemma 1 and 2 in [7]).

To find a cluster, DBSCAN starts with an arbitrary core object *p* which is not yet clustered and retrieves all objects density-reachable from *p*. The retrieval of density-reachable objects is performed by successive region queries which are supported efficiently by spatial access methods such as R*-trees [3] for data from a vector space or M-trees [4] for data from a metric space.

## 5 Determination of a Local Model

After having clustered the data locally, we need a small number of representatives which describe the local clustering result accurately. We have to find an optimum trade-off between the following two opposite requirements:

- We would like to have a small number of representatives.
- We would like to have an accurate description of a local cluster.

As the core points computed during the DBSCAN run contain in its *Eps*-neighborhood at least *MinPts* other objects, they might serve as good representatives. Unfortunately, their number can become very high, especially in very dense areas of clusters. In the following, we will introduce two different approaches for determining suitable representatives which are both based on the concept of *specific core-points*.

**Definition 6 (specific core points).** Let *D* be a set of objects and let $C \in 2^D$ be a cluster wrt. *Eps* and *MinPts*. Furthermore, let $Cor_C \subseteq C$ be the set of core-points belonging to this cluster. Then $Scor_C \subseteq C$ is called a *complete set of specific core points of C* iff the following conditions are true.

- $Scor_C \subseteq Cor_C$
- $\forall s_i, s_j \in Scor_C: s_i \neq s_i \Rightarrow s_i \notin N_{Eps}(s_j)$
- $\forall c \in Cor_C \; \exists s \in Scor_C: c \in N_{Eps}(s)$

There might exist several different sets $Scor_C$ which fulfil Definition 6. Each of these sets $Scor_C$ usually consists of several specific core points which can be used to describe the cluster *C*.

The small example in Figure 3a shows that if *A* is an element of the set of specific core-points *Scor*, object *B* can not be included in *Scor* as it is located within the *Eps*-neighborhood of *A*. *C* might be contained in *Scor* as it is not in the *Eps*-neighborhood of *A*. On the other hand, if *B* is within *Scor*, A and *C* are not contained in *Scor* as they are both in the *Eps*-neighborhood of *B*. The actual processing order of the objects during the DBSCAN run determines a concrete set of specific core points. For instance, if the core-point *B* is visited first during the DBSCAN run, the core-points *A* and *C* are not included in *Scor*.

In the following, we introduce two local models called, $REP_{Scor}$ (cf. Section 5.1) and $REP_{k\text{-}Means}$ (cf. Section 5.2) which both create a local model based on the complete set of specific core points.

## 5.1 Local Model: $REP_{Scor}$

In this model, we represent each local cluster $C_i$ by a complete set of specific core points $Scor_{C_i}$. If we assume that we have found *n* clusters $C_1,..,C_n$ on a local site *k*, the local model $LocalModel_k$ is formed by the union of the different sets $Scor_{C_i}$.

In the case of density-based clustering, very often several core points are in the *Eps*-neighborhood of another core point. This is especially true, if we have dense clusters and a large *Eps*-value. In Figure 3a, for instance, the two core points *A* and *B* are within the *Eps*-range of each other as *dist*(*A*, *B*) is smaller than *Eps*.

Assuming core point *A* is a specific core point, i.e. $A \in Scor$, than $B \notin Scor$ because of condition 2 in Definition 6. In this case, object *A* should not only represent the objects in its own neighborhood, but also the objects in the neighborhood of *B*, i.e. *A* should represent all objects of $N_{Eps}(A) \cup N_{Eps}(B)$. In order for *A* to be a representative for the objects $N_{Eps}(A) \cup N_{Eps}(B)$, we have to assign a new specific $\varepsilon_A$–range to *A* with $\varepsilon_A = Eps + dist(A,B)$ (cf. Figure 3a). Of course we have to assign such a specific ε–range to all specific core points, which motivates the following definition:

**Definition 7 (specific ε–ranges).** Let $C \subseteq D$ be a cluster wrt. *Eps* and *MinPts*. Furthermore, let $Scor \subseteq C$ be a complete set of specific core-points. Then we assign to each $s \in Scor$ an $\varepsilon_s$–range indicating the represented area of *s*:

$$\varepsilon_s := Eps + max\{dist(s,s_i) | s_i \in Cor \wedge s_i \in N_{Eps}(s)\} .$$

This specific ε–range value is part of the local model and is evaluated on the server site to develop an accurate global model. Furthermore, it is very important for the updating process of the local objects. The specific ε–range value is integrated into the local model of site *k* as follows:

$$LocalModel_k := \bigcup_{i \in 1..n} \{(s,\varepsilon_s) | s \in Scor_{C_i}\}.$$

## 5.2 Local Model: $REP_{k\text{-}Means}$

This approach is also based on the complete set of specific core-points. In contrast to the previous approach, the specific core points are not directly used to describe a cluster. Instead, we use the number $|Scor_C|$ and the elements of $Scor_C$ as input parameters for a further "clustering step" with an adapted version of *k-means*. For each cluster *C*, found by *DBSCAN*, k-means yields $|Scor_C|$ centroids within *C*. These centroids are used as representatives. The small example in Figure 3b shows that if object *A* is a specific core point, and

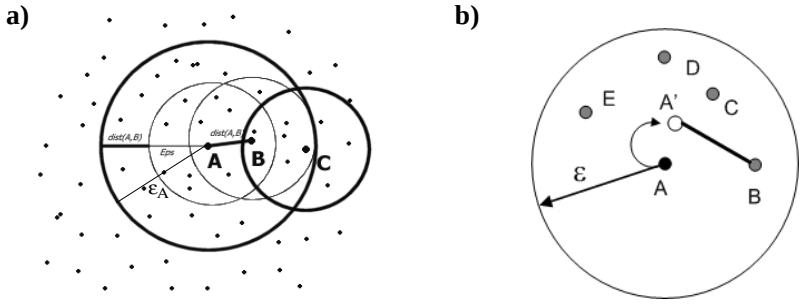


**Fig. 3.** Local models
**a)** $REP_{Scor}$: specific core points and specific ε–range **b)** $REP_{k\text{-}Means}$: representatives by using k-means

we apply an additional clustering step by using k-means, we get a more appropriate representative A'.

K-means is a partitioning based clustering method which needs as input parameters the number *m* of clusters which should be detected within a set *M* of objects. Furthermore, we have to provide *m* starting points for this algorithm, if we want to find *m* clusters. We use k-means as follows:

- Each local cluster *C* which was found throughout the original DBSCAN run on the local site forms a set *M* of objects which is again clustered with k-means.
- We ask k-means to find $|Scor_C|$ (sub)clusters within *C*, as all specific core points together yield a suitable number of representatives. Each of the centroids found by k-means within cluster *C* is then used as a new representative. Thus the number of representatives for each cluster is the same as in the previous approach.
- As initial starting points for the clustering of *C* with k-means, we use the set of complete specific core points $Scor_C$.

Again, let us assume that there are *n* clusters $C_1,..,C_n$ on a local site *k*. Furthermore, let $c_{i,1}..c_{i,|Scor_{C_i}|}$ be the $|Scor_{C_i}|$ centroids found by the clustering of $C_i$ with k-means. Let $O_{i,j} \subseteq C_i$ be the set of objects which are assigned to the centroid $c_{i,j}$. Then we assign to each *centroid* $c_{i,j}$ an $\varepsilon_{c_{i,j}}$–range, indicating the represented area by $c_{i,j}$, as follows:

$$\varepsilon_{c_{i,j}} := max\{dist(o,c_{i,j}) | o \in O_{i,j}\}.$$

Finally, the local model, describing the *n* clusters on site *k*, can be generated analogously to the previous section as follows:

$$LocalModel_k := \bigcup_{i \in 1..n} \bigcup_{j \in 1..|Scor_{C_i}|} (c_{i,j}, \varepsilon_{c_{i,j}}).$$

# 6 Determination of a Global Model

Each local model $LocalModel_k$ consists of a set of $m_k$ pairs, consisting of a representative *r* and an ε–range value $\varepsilon_r$. The number *m* of pairs transmitted from each site *k* is determined by the number *n* of clusters $C_i$ found on site *k* and the number $|Scor_{C_i}|$ of specific core-points for each cluster $C_i$ as follows:

$$m = \sum_{i=1..n} |Scor_{C_i}| \quad .$$

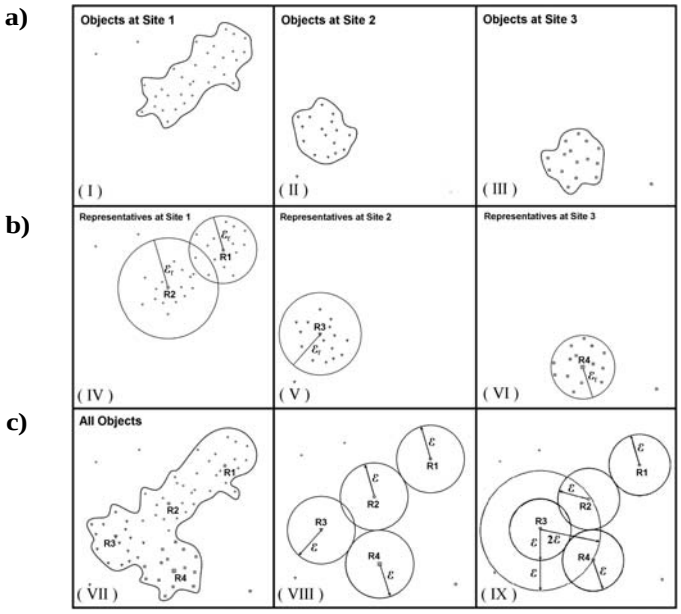


**Fig. 4.** Determination of a global model **a)** local clusters **b)** local representatives **c)** determination of a global model with $Eps_{global} = 2 \cdot Eps_{local}$

Each of these pairs $(r, \varepsilon_r)$ represent several objects which are all located in $N_{\varepsilon_r}(r)$, i.e. the $\varepsilon_r$-neighborhood of $r$. All objects contained in $N_{\varepsilon_r}(r)$ belongs to the same cluster. To put it another way, each specific local representative forms a cluster on its own. Obviously, we have to check whether it is possible to merge two or more of these clusters. These merged local representatives together with the unmerged local representatives form the global model. Thus, the global model consist of clusters consisting of one or of several local representatives.

To find such a global model, we use the density based clustering algorithm DBSCAN again. We would like to create a clustering similar to the one produced by DBSCAN if applied to the complete dataset with the local parameter settings. As we have only access to the set of all local representatives, the global parameter setting has to be adapted to this aggregated local information.

As we assume that all local representatives form a cluster on their own it is enough to use a $Min\text{-}Pts_{global}$-parameter of 2. If 2 representatives, stemming from the same or different local sites, are density connected to each other wrt. $MinPts_{global}$ and $Eps_{global}$, then they belong to the same global cluster.

The question for a suitable $Eps_{global}$ value, is much more difficult. Obviously, $Eps_{global}$ should be greater than the Eps-parameter $Eps_{local}$ used for the clustering on the local sites. For high $Eps_{global}$ values, we run the risk of merging clusters together which do not belong together. On the other hand, if we use small $Eps_{global}$ values, we might not be able to detect clusters belonging together. Therefore, we suggest that the $Eps_{global}$ parameter should be tunable by the user dependent on the $\varepsilon_R$ values of all local representatives $R$. If these $\varepsilon_R$ values are generally high it is advisable to use a high $Eps_{global}$ value. On the other hand, if the $\varepsilon_R$ values are low, a small $Eps_{global}$ value is better. The default value which we propose is

equal to the maximum value of all $\varepsilon_R$ values of all local representatives *R*. This default $Eps_{global}$ value is generally close to $2 \cdot Eps_{local}$ (cf. Section 9).

In Figure 4, an example for $Eps_{global}=2 \cdot Eps_{local}$ is depicted. In Figure 4a the independently detected clusters on site 1,2 and 3 are depicted. The cluster on site 1 is characterized by two representatives *R1* and *R2*, whereas the clusters on site 2 and site 3 are only characterized by one representative as shown in Figure 4b. Figure 4c (VII) illustrates that all 4 clusters from the different sites belong to one large cluster. Figure 4c (VIII) illustrates that an $Eps_{global}$ equal to $Eps_{local}$ is insufficient to detect this global cluster. On the other hand, if we use an $Eps_{global}$ parameter equal to $2 \cdot Eps_{local}$ the 4 representatives are merged together to one large cluster (cf. Figure 4c (IX)).

Instead of a user defined $Eps_{global}$ parameter, we could also use a hierarchical density based clustering algorithm, e.g. OPTICS [1], for the creation of the global model. This approach would enable the user to visually analyze the hierarchical clustering structure for several $Eps_{global}$-parameters without running the clustering algorithm again and again. We refine from this approach because of several reasons. First, the relabeling process discussed in the next section would become very tedious. Second, a quantitative evaluation (cf. Section 9) of our DBDC algorithm is almost impossible. Third, the incremental version of DBSCAN allows us to start with the construction of the global model after the first representatives of any local model come in. Thus we do not have to wait for all clients to have transmitted their complete local models.

# 7 Updating of the Local Clustering based on the Global Model

After having created a global clustering, we send the complete global model to all client sites. The client sites relabel all objects located on their site independently from each other. On the client site, two former independent clusters may be merged due to this new relabeling. Furthermore, objects which were formerly assigned to local noise are now part of a global cluster. If a local object *o* is in the $\varepsilon_r$–range of a representative *r*, *o* is assigned to the same global cluster as *r*.

Figure 5 depicts an example for this relabeling process. The objects *R1* and *R2* are the local representatives. Each of them forms a cluster on its own. Objects *A* and *B* have been classified as noise. Representative *R3* is a representative stemming from another site. As *R1*, *R2* and *R3* belong to the same global cluster all Objects from the local clusters *Cluster 1* and *Cluster 2* are assigned to this global cluster. Furthermore, the objects *A* and *B* are assigned to this global cluster as they are within the $\varepsilon_{R3}$–neighborhood of *R3*, i.e. $A, B \in N_{\varepsilon_{R3}}(R3)$. On the other hand, object *C* still belongs to noise as $C \notin N_{\varepsilon_{R3}}(R3)$.

These updated local client clusterings help the clients to answer server questions efficiently, e.g. questions such as "*give me all objects on your site which belong to the global cluster 4711*".

# 8 Quality of Distributed Clustering

There exist no general quality measure which helps to evaluate the quality of a distributed clustering. If we want to evaluate our new *DBDC* approach, we first have to tackle the problem of finding a suitable quality criterion. Such a suitable quality criterion should yield a high quality value if we compare a "good" distributed clustering to a central clustering,

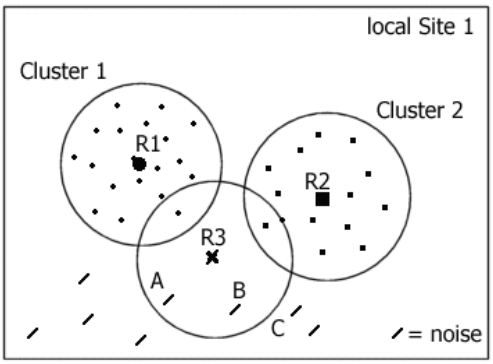


**Fig. 5.**Relabeling of the local clustering

i.e. reference clustering. On the other hand, it should yield a low value if we compare a "bad" distributed clustering to a central clustering. Needless to say, if we compare a reference clustering to itself, the quality should be 100%. Let us first formally introduce the notion of a clustering.

**Definition 8 (clustering $CL$).** Let $D = \{x_1, \ldots, x_n\}$ be a database consisting of *n* objects. Then, we call any set $CL$ a clustering of *D w.r.t. MinPts*, if it fulfils the following properties:

- $CL \subseteq 2^D$
- $\forall C \in CL: (|C| \geq MinPts)$
- $\forall C_1, C_2 \in CL: C_1 \neq C_2 \Rightarrow C_1 \cap C_2 = \emptyset$

In the following we denote by $CL_{distr}$ a clustering resulting from our distributed approach and by $CL_{central}$ our central reference clustering. We will define two different quality criterions which measure the similarity between $CL_{distr}$ and $CL_{central}$. We compare the two introduced quality criterions to each other by discussing a small example.

Let us assume that we have *n* objects, distributed over *k* sites. Our *DBDC*-algorithm, assigns each object *x*, either to a cluster or to noise. We compare the result of our *DBDC*-algorithm to a central clustering of the *n* objects using DBSCAN. Then we assign to each object *x* a numerical value *P* (*x*) indicating the quality for this specific object. The overall quality of the distributed clustering is the mean of the qualities assigned to each object.

**Definition 9 (distributed clustering quality $Q_{DBDC}$).** Let $D = \{x_1, \ldots, x_n\}$ be a database consisting of *n* objects. Let *P* be an object quality function $P: D \rightarrow [0, 1]$. Then the quality $Q_{DBDC}$ of our distributed clustering w.r.t. *P* is computed as follows:

$$Q_{DBDC} = \frac{\sum_{i=1\ldots n} P(x_i)}{n}$$

The crucial question is "what is a suitable object quality function?". In the following two subsections, we will discuss two different object functions *P*.

## 8.1 First Object Quality Function $P^I$

Obviously, $P(x)$ should yield a rather high value, if an object $x$ together with many other objects is contained in a distributed cluster $C_d$ and a central cluster $C_c$. In the case of density-based partitioning clustering, a cluster might consist of only *MinPts* elements. Therefore, the number of objects contained in two identical clusters might be not higher than *MinPts*. On the other hand, each cluster consists of at least *MinPts* elements. Therefore, asking for less than *MinPts* elements in both clusters would weakening the quality criterion unnecessarily.

If $x$ is included in a distributed cluster $C_d$ but is assigned to noise by the central clustering, the value of $P(x)$ should be 0. If $x$ is not contained in any distributed cluster, i.e. it is assigned to noise, a high object quality value requires that it is also not contained in a central cluster. In the following, we will define a discrete object quality function $P^I$ which assigns either 0 or 1 to an object $x$, i.e. $P^I(x) = 0$ or $P^I(x) = 1$.

**Definition 10 (discrete object quality $P^I$).** Let $x \in D$ and let $C_d$, $C_c$ be two cluster. Then we can define an object quality function $P^I: D \rightarrow \{0, 1\}$ w.r.t. to a quality parameter $qp$ as follows:

$$P^I(x) = \begin{cases} 0, & x \in Noise_{distr} \wedge x \notin Noise_{central} \\ 0, & x \notin Noise_{distr} \wedge x \in Noise_{central} \\ 1, & x \in Noise_{distr} \wedge x \in Noise_{central} \\ 1, & x \notin Noise_{distr} \wedge x \notin Noise_{central} \wedge (|C_d \cap C_c| \geq qp) \\ 0, & x \notin Noise_{distr} \wedge x \notin Noise_{central} \wedge (|C_d \cap C_c| < qp) \end{cases}$$

The main advantage of the object quality function $P^I$ is that it is rather simple because it yields only a boolean return value, i.e. it tells whether an object was clustered correctly or falsely. Nevertheless, sometimes a more subtle quality measure is required which does not only assign a binary quality value to an object. In the following section, we will introduce a new object quality function which is not confined to the two binary quality values 0 and 1. This more sophisticated quality function can compute any value in between 0 and 1 which much better reflects the notion of "correctly clustered".

## 8.2 Second Object Quality Function $P^{II}$

The main idea of our new quality function is to take the number of elements which were clustered together with the object $x$ during the distributed and the central clustering into consideration. Furthermore, we decrease the quality of $x$ if there are objects which have been clustered together with $x$ in only one of the two clusterings.

**Definition 11 (continuous object quality $P^{II}$).** Let $x \in D$ and let $C_d$, $C_c$ be a central and a distributed cluster. Then we define an object quality function $P^{II}: D \rightarrow [0, 1]$ as follows:

$$P^{II}(x) = \begin{cases} 1, & x \in Noise_{distr} \wedge x \notin Noise_{central} \\ 0, & x \notin Noise_{distr} \wedge x \in Noise_{central} \\ 1, & x \in Noise_{distr} \wedge x \in Noise_{central} \\ \dfrac{|C_d \cap C_c|}{|C_d \cup C_c|}, & otherwise \end{cases}$$

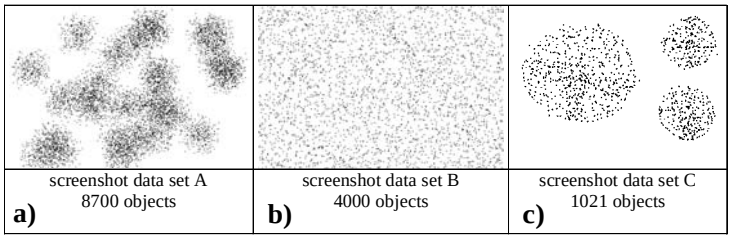


**Fig. 6.** Used test data sets
**a)** test data set *A* **a)** test data set *B* **a)** test data set *C*

# 9 Experimental Evaluation

We evaluated our *DBDC*-approach based on three different 2-dimensional point sets where we varied both the number of points and the characteristics of the point sets. Figure 6 depicts the three used test data sets *A* (8700 objects, randomly generated data/cluster), *B* (4000 objects, very noisy data) and *C* (1021 objects, 3 clusters) on the central site. In order to evaluate our *DBDC*-approach, we equally distributed the data set onto the different client sites and then compared *DBDC* to a single run of *DBSCAN* on all data points. We carried out all local clusterings sequentially. Then, we collected all representatives of all local runs, and applied a global clustering on these representatives. For all these steps, we always used the same computer. The overall runtime was formed by adding the time needed for the global clustering to the maximum time needed for the local clusterings. All experiments were performed on a Pentium III/700 machine.

In a first set of experiments, we consider efficiency aspects, whereas in the following sections we concentrate on quality aspects.

## 9.1 Efficiency

In Figure 7, we used test data sets with varying cardinalities to compare the overall runtime of our *DBDC*-algorithm to the runtime of a central clustering. Furthermore, we compared our two local models w.r.t. efficiency to each other. Figure 7a shows that our *DBDC*-approach outperforms a central clustering by far for large data sets. For instance, for a point set consisting of 100,000 points, both DBDC approaches, i.e. $DBDC(Rep_{SCor})$ and $DBDC(Rep_{K\text{-}Means})$, outperform the central DBSCAN algorithm by more than one order of magnitude independent of the used local clustering. Furthermore, Figure 7a shows that the local model for $REP_{Scor}$ can more efficiently be computed than the local model for $REP_{k\text{-}Means}$.

Figure 7b shows that for small data sets our *DBDC*-approach is slightly slower than the central clustering approach. Nevertheless, the additional overhead for distributed clustering is almost negligible even for small data sets.

In Figure 8 it is depicted in what way the overall runtime depends on the number of used sites. We compared DBDC based on $REP_{Scor}$ to a central clustering with DBSCAN. Our experiments show that we obtain a speed-up factor which is somewhere between $O(n)$ and $O(n^2)$. This high speed-up factor is due to the fact that DBSCAN has a runtime complexity somewhere between $O(n\log n)$ and $O(n^2)$ when using a suitable index structure, e.g. an R*-tree [3].

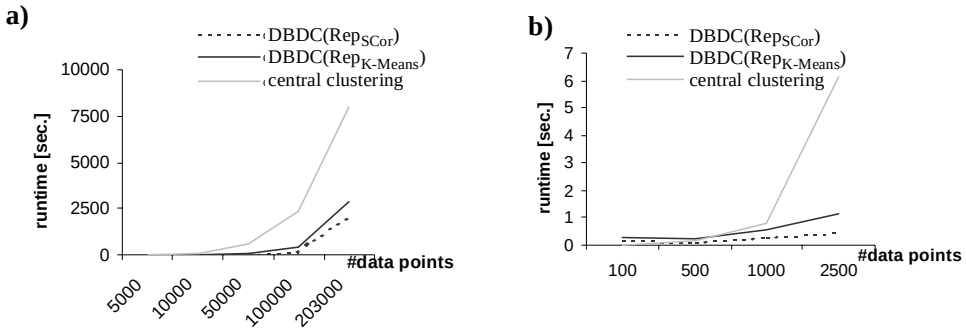


**Fig. 7.** Overall runtime for central and distributed clustering dependent on the cardinality of the data set *A*. **a)** high number of data objects **b)** small number of data objects.

### 9.2 Quality

In the next set of experiments we evaluated the quality of our two introduced object quality functions $P^I$ and $P^{II}$ together with the quality of our DBDC-approach. Figure 9a shows that the quality according to $P^I$ of both local models is very high and does not change if we vary the $Eps_{global}$ parameter during global clustering. On the other hand, if we look at Figure 9b, we can clearly see that for $Eps_{global}$ parameters equal to $2 \cdot Eps_{local}$, we get the best quality for both local models. This is equal to the default value for the server site clustering which we derived in Section 6. Furthermore, the quality worsens for very high and very small $Eps_{global}$ parameters, which is in accordance to the quality which an experienced user would assign to those clusterings.

To sum up, these experiments yield two basic insights:

- The object quality function $P^{II}$ is more suitable than $P^I$.
- A good $Eps_{global}$ parameter is around $2 \cdot Eps_{local}$

Furthermore, the experiments indicate that the local model $REP_{k\text{-}Means}$ yields slightly higher quality.

For the following experiments, we used an $Eps_{global}$ parameter of $2 \cdot Eps_{local}$.

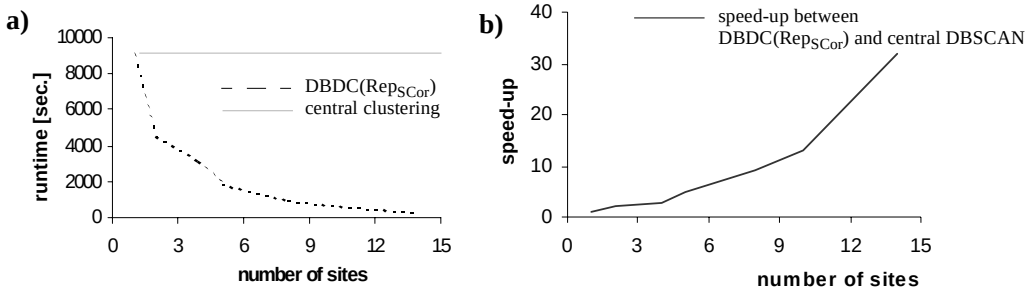


**Fig. 8.** Overall runtime for central and distributed clustering $DBDC(Rep_{SCor})$ for a data set of 203,000 points. **a)** dependent on the number of sites **b)** speed-up of *DBDC* compared to central *DBSCAN*

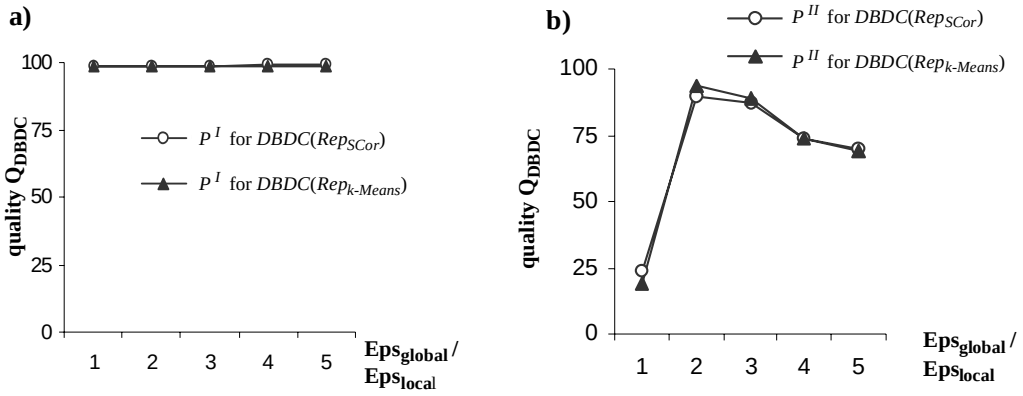


**Fig. 9.** Evaluation of object quality functions for varying $Eps_{global}$ parameters for data set *A* on 4 local sites. **a)** object quality function $P^I$ **b)** object quality function $P^{II}$

Figure 10 shows how the quality of our *DBDC*-approach depends on the number of client-sites. We can see that the quality according to $P^I$ is independent of the number of client sites which indicates again that this quality measure is unsuitable. On the other hand, the quality computed by $P^{II}$ is in accordance with the intuitive quality which an experienced user would assign to the distributed clusterings on the varying number of sites. Although, we have a slight decreasing quality for an increasing number of sites, the overall quality for both local models $REP_{k\text{-}Means}$ and $REP_{Scor}$ is very high.

Figure 11 shows that for the three different data sets *A*, *B* and *C* our *DBDC*-approach yields good results for both local models. The more accurate quality measure $P^{II}$ indicates that the *DBDC*-approach based on $REP_{k\text{-}Means}$ yields a quality which reflects more ade-

| number of sites | number of local repr.[%] | $DBDC(REP_{k\text{-}Means})$ | | $DBDC(REP_{Scor})$ | |
|---|---|---|---|---|---|
| | | $P^I$ | $P^{II}$ | $P^I$ | $P^{II}$ |
| 2 | 16 | 99 | 98 | 99 | 97 |
| 4 | 16 | 98 | 97 | 98 | 96 |
| 5 | 17 | 98 | 97 | 98 | 96 |
| 8 | 17 | 98 | 96 | 98 | 96 |
| 10 | 17 | 98 | 97 | 98 | 96 |
| 14 | 17 | 98 | 89 | 98 | 89 |
| 20 | 17 | 98 | 91 | 98 | 91 |

**Fig. 10.** Quality $Q_{DBDC}$ dependent on the number of client sites, the local models $REP_{k\text{-}Means}$ and $REP_{Scor}$, and the object quality functions $P^I$ and $P^{II}$ for test data set *A* and $Eps_{global} = 2 \cdot Eps_{local}$

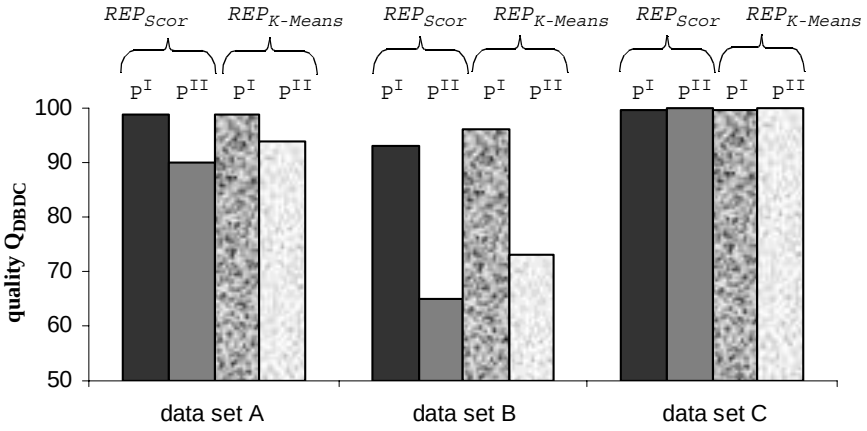


**Fig. 11.** Quality for data sets *A*, *B* and *C*

quately the user's expectations. This is especially true for the rather noisy data set *B*, where $P^{II}$ yields the lower quality corresponding to the user's intuition.

To sum up, our new DBDC-approach based on $REP_{k\text{-}Means}$ efficiently yields a very high quality even for a rather high number of local sites and data sets of various cardinalities and characteristics.

## 10 Conclusions

In this paper, we first motivated the need of distributed clustering algorithms. Due to technical, economical or security reasons, it is often not possible to transmit all data from different local sites to one central server site and then cluster the data there. Therefore, we have to apply an efficient and effective distributed clustering algorithm from which a lot of application ranges will benefit. We developed a partitioning distributed clustering algorithm which is based on the density-based clustering algorithm DBSCAN. We clustered the data locally and independently from each other and transmitted only aggregated information about the local data to a central server. This aggregated information consists of a set of pairs, comprising a representative $r$ and an $\varepsilon$–range value $\varepsilon_r$, indicating the validity area of the representative. Based on these local models, we reconstruct a global clustering. This global clustering was carried out by means of standard DBSCAN where the two input-parameters $Eps_{global}$ and $MinPts_{global}$ were chosen such that the information contained in the local models are processed in the best possible way. The created global model is sent to all clients, which use this information to relable their own objects.

As there exists no general quality measures which helps to evaluate the quality of a distributed clustering, we introduced suitable quality criteria on our own. In the experimental evaluation, we discussed the suitability of our quality criteria and our density-based distributed clustering approach. Based on the quality criteria, we showed that our new distributed clustering approach yields almost the same clustering quality as a central clustering on all data. On the other hand, we showed that we have an enormous efficiency advantage compared to a central clustering carried out on all data.

## References


1. Ankerst M., Breunig M. M., Kriegel H.-P., Sander J.: "OPTICS: Ordering Points To Identify the Clustering Structure", Proc. ACM SIGMOD, Philadelphia, PA, 1999, pp. 49-60.
2. Agrawal R., Shafer J. C.: "Parallel mining of association rules: Design, implementation, and experience" IEEE Trans. Knowledge and Data Eng. 8 (1996) 962-969
3. Beckmann N., Kriegel H.-P., Schneider R., Seeger B.: "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles", Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'90), Atlantic City, NJ, ACM Press, New York, 1990, pp. 322?331.
4. Ciaccia P., Patella M., Zezula P.: "*M-tree: An Efficient Access Method for Similarity Search in Metric Spaces*", Proc. 23rd Int. VLDB, Athens, Greece, 1997, pp. 426-435.
5. Dhillon I. S., Modh Dh. S.: "A Data-Clustering Algorithm On Distributed Memory Multiprocessors", SIGKDD 99
6. Ester M., Kriegel H.-P., Sander J., Wimmer M., Xu X.: "Incremental Clustering for Mining in a Data Warehousing Environment", VLDB 98
7. Ester M., Kriegel H.-P., Sander J., Xu X.: "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96), Portland, OR, AAAI Press, 1996, pp.226-231.
8. Ertöz L., Steinbach M., Kumar V.: "Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data", SIAM International Conference on Data Mining (2003)
9. Forman G., Zhang B. : "Distributed Data Clustering Can Be Efficient and Exact". SIGKDD Explorations 2(2): 34-38 (2000)
10. Hanisch R. J.: "Distributed Data Systems and Services for Astronomy and the Space Sciences", in ASP Conf. Ser., Vol. 216, Astronomical Data Analysis Software and Systems IX, eds. N. Manset, C. Veillet, D. Crabtree (San Francisco: ASP) 2000
11. Hartigan J. A.: "Clustering Algorithms", Wiley, 1975
12. Han E. H., Karypis G., Kumar V.: "Scalable parallel data mining for association rules" In: SIGMOD Record: Proceedings of the 1997 ACM-SIGMOD Conference on Management of Data, Tucson, AZ, USA. (1997) 277-288
13. Jain A. K., Dubes R.C.: "Algorithms for Clustering Data", Prentice-Hall Inc., 1988.
14. Johnson E., Kargupta H.: "Hierarchical Clustering From Distributed, Heterogeneous Data." In Zaki M. and Ho C., editors, Large-Scale Parallel KDD Systems. Lecture Notes in Computer Science, colum 1759, 221-244. Springer-Verlag, 1999
15. Jain A. K., Murty M. N., Flynn P. J.:"Data Clustering: A Review", ACM Computing Surveys, Vol. 31, No. 3, Sep. 1999, pp. 265-323.
16. Kargupta H., Chan P. (editors) : "Advances in Distributed and Parallel Knowledge Discovery", AAAI/MIT Press, 2000
17. Shafer J., Agrawal R., Mehta M.: "A scalable parallel classifier for data mining" In: Proc. 22nd International Conference on VLDB, Mumbai, India. (1996)
18. Srivastava A., Han E. H., Kumar V., Singh V.: "Parallel formulations of decision-tree classification algorithms" In: Proc. 1998 International Conference on Parallel Processing. (1998)
19. Samatova N.F., Ostrouchov G., Geist A., Melechko A.V.: "RACHET: An Efficient Cover-Based Merging of Clustering Hierarchies from Distributed Datasets, Distributed and Parallel Databases, 11(2): 157-180; Mar 2002"
20. Sayal M., Scheuermann P.: "A Distributed Clustering Algorithm for Web-Based Access Patterns", in Proceedings of the 2nd ACM-SIGMOD Workshop on Distributed and Parallel Knowledge Discovery, Boston, August 2000"
21. Xu X., Jäger J., H.-P. Kriegel.: "A Fast Parallel Clustering Algorithm for Large Spatial Databases", Data Mining and Knowledge Discovery, 3, 263-290 (1999), Kluwer Academic Publisher
22. Zaki M. J.., Parthasarathy S., Ogihara M., Li W.: "New parallel algorithms for fast discovery of association rule" Data Mining and Knowledge Discovery, 1, 343-373 (1997)