Workshop on Clustering Large Data Sets (ICDM2003), Melbourne, FL, 2003

Towards Effective and Efficient Distributed Clustering

Eshref Januzaj Hans-Peter Kriegel Martin Pfeifle

Institute for Computer Science
University of Munich
Oettingenstr. 67, 80538 Munich, Germany
{januzaj, kriegel, pfeifle} @dbs.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 many of these areas, the data are originally collected at different sites. In order to extract information out of these data, they are brought together and then clustered. In this paper, we propose a different approach. We cluster the data locally and extract suitable representatives out of 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. In our experimental evaluation, we will show that we do not have to sacrifice the clustering quality in order to gain an efficiency advantage if we use distributed clustering.

1. Introduction

Knowledge Discovery in Databases (KDD) tries to identifying 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 amount 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, big companies such as Daimler-Chrysler 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 amount of data from one site to another central site is in some application areas almost impossible. In astronomy, for instance, there exist several high sophisticated space telescopes spread all over the world. These telescopes gather data unceasingly. Each of them is able to collect 1GB of data per hour [Han 00] which can only difficultly be transmitted to a central site and 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 out of 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 sent this information to a central site. We use the following terms interchangeably for this locally created information: local model, local representatives, or aggregated information of the local site. The transmission cost 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 sent 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 the 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

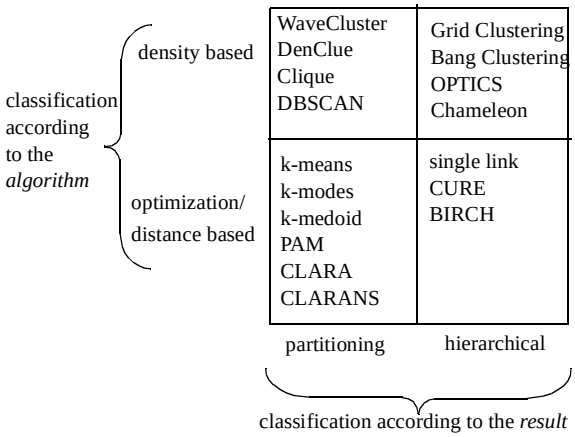

Figure 1: Classification scheme for clustering algorithms

update their clustering based on the global clustering information. In Section 8, we present the experimental evaluation of our new efficient *DBDC* (Density Based Distributed Clustering) approach and show that its use does not go along with a deterioration of quality. We conclude the paper in Section 9 with a short summary and a few remarks on future work.

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 [JMF 99] [JD 88]. 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 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* view. Here we can distinguish between *optimization or distance based* algorithms and *density based* algorithms. Distance based methods use the distances between the objects directly in order to optimize a global criterion function. 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 [JMF 99] were an excellent overview and further references can be found.

2.2. Parallel 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 [KC 00] 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 [AS 96] [HKK 97] [KHS 97] [KPHJ 00] [SAM 96] [SHKS 98] [ZHA 98] [ZPOL 97], there do not exist many algorithms for parallel and distributed clustering.

In [XJK 99] a parallel version of DBSCAN [EKSX 96] and in [DM 99] a parallel version of k-means [Har 75] were introduced. Both algorithms start with the complete data set residing on one central sever and then distribute the data among the different clients. For instance, in the case of parallel DBSCAN, the data are organized at the server site within an R*-tree [BKSS 90]. This preprocessed data is then distributed among the clients which communicate with each other via message passing.

In this paper, we propose another approach based on distributed clustering which inherently can not 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 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

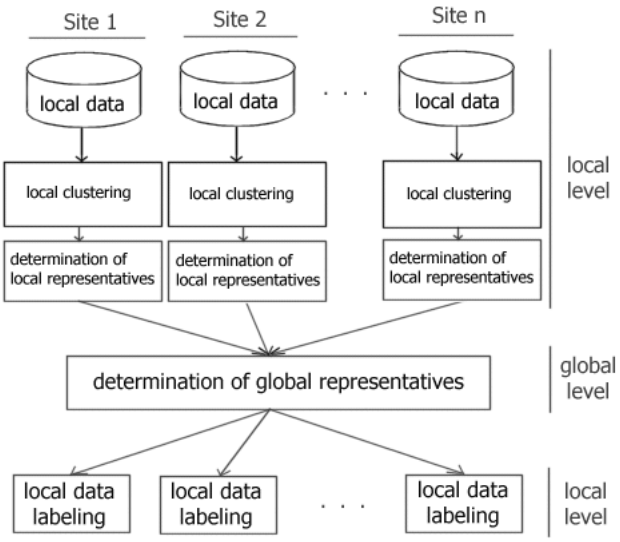

Figure 2: Distributed Clustering

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 ε–range value. Thus, a representative is a good approximation 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

In Figure 3, the different steps of our *DBDS* algorithm are depicted for an example consisting of 8700 points distributed over 4 different sites (cf. Figure 3a). In Figure 3b, the extracted local representatives are depicted. This number is much smaller than the number of all objects. In Figure 3c the union of all local representatives from all 4 sites is shown. Based on these united representatives, the DBDC algorithm determines a global clustering which is visually equivalent to the result produced by a DBSCAN algorithm applied to all 8700 points at once (cf. Figure 3d).

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 [Har 75] 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 [ESK 03]. 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 [ESK 03]. We used the density-based clustering algorithm DBSCAN [EKSX 96], because it yields the following advantages:

- DBSCAN is a very efficient and effective clustering algorithm.
- There exist 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 [EKS$^+$ 98].
- DBSCAN is rather robust concerning outliers.
- DBSCAN can be used for all kind of metric data spaces and is not confined to vector spaces.
- DBSCAN can easily be implemented.

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).

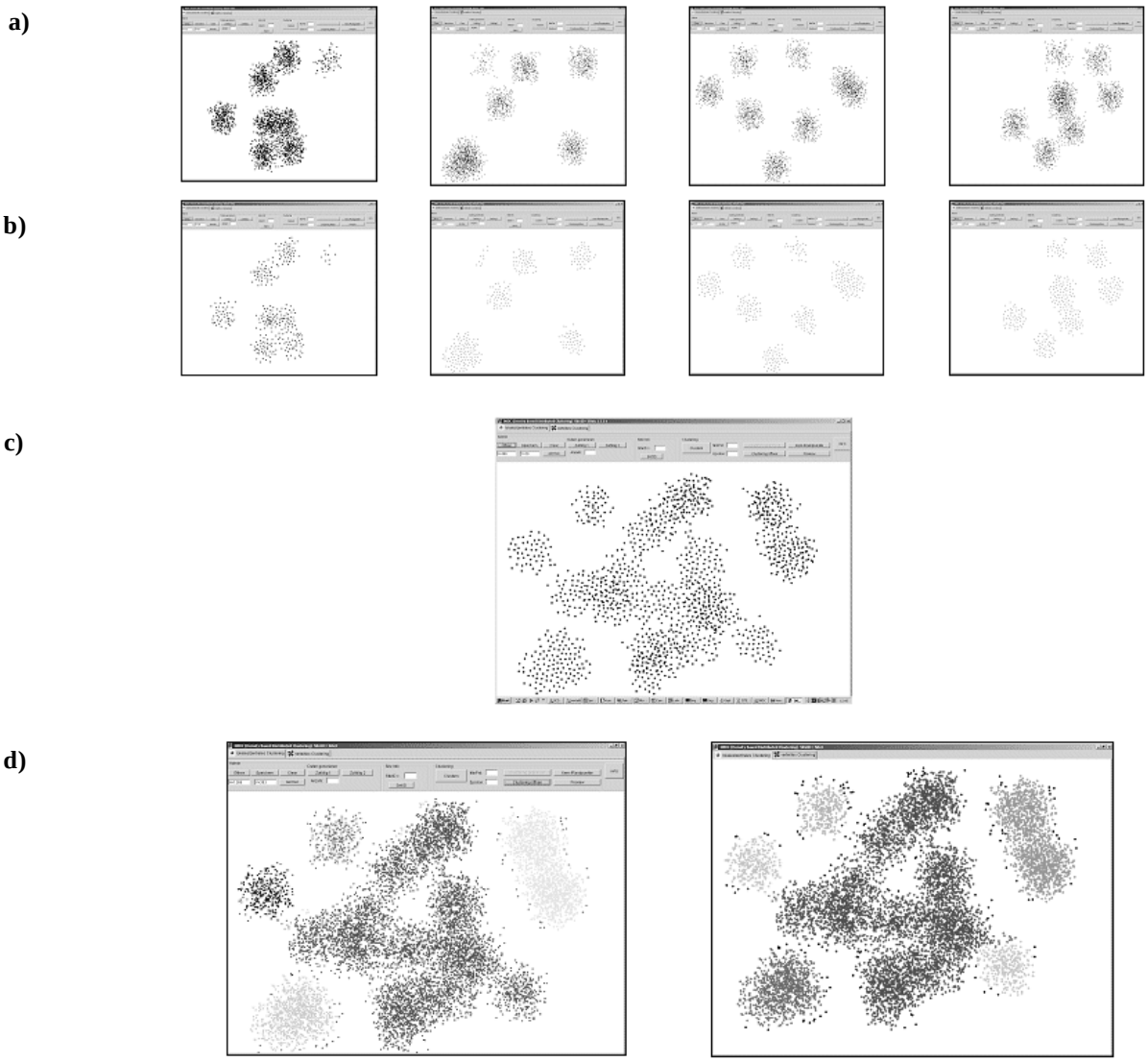

Figure 3: Screenshots from DBDC algorithm
a) 4 local sites **b)** representatives of the local sites **c)** representatives of all local sites
d) left: result from a reference clustering with *DBSCAN* where *MinPts* = 4 and *Eps* = 15
right: result yielded by *DBDC* where *MinPts = 4* and Eps_{local} = 15

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 [EKSX 96].

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.)
- $Card(N_{Eps}(q)) \geq MinPts$.

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*.

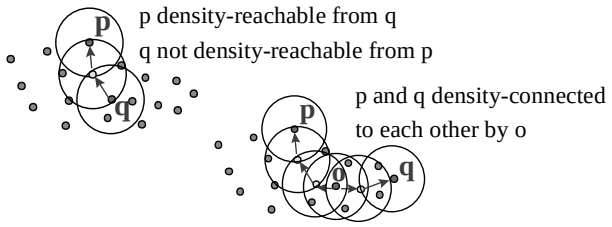

Figure 4: Density-reachability and density-connectivity

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 $Card(N_{Eps}(o)) \geq MinPts$. 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. Figure 4 illustrates the definitions on a sample database of objects from a 2-dimensional *vector* space. Note however, that the above definitions only require a distance measure and will also apply to data from a metric space.

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 $C_1,..., C_k$ 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* and $p \in C$. 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 [EKSX 96]).

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 [BKSS 90] for data from a vector space or M-trees [CPZ 97] 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 $C \subseteq$ 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.

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

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

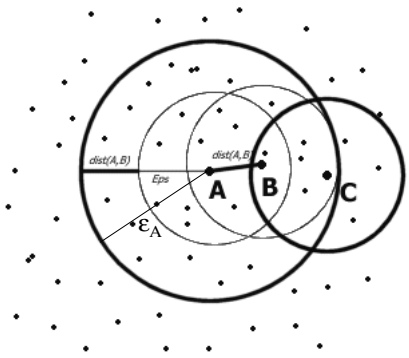

Figure 5: Specific core points and specific ε–range

The small example in Figure 5 shows that if *A* is an element of the set of specific core-points *Scor*, point *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 $C_i,..,C_n$ 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 5, 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, point *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 point $N_{Eps}(A) \cup N_{Eps}(B)$. In order for *A* to be a representative for the points $N_{Eps}(A) \cup N_{Eps}(B)$, we have to assign a new specific ε_A–range to *A* with $\varepsilon_A = Eps + dist(A,B)$ (cf. Figure 5). 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 ε_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 (cf. Section 7). The specific ε–range value is integrated into the local model of site *k* as follows:

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

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 foregoing 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.

K-means is a partitioning based clustering method which needs as input parameters the number *m* of clusters which should be detected within a point set *M*. 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 point set *M* 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 foregoing 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 formed analogously to the foregoing section as follows:

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

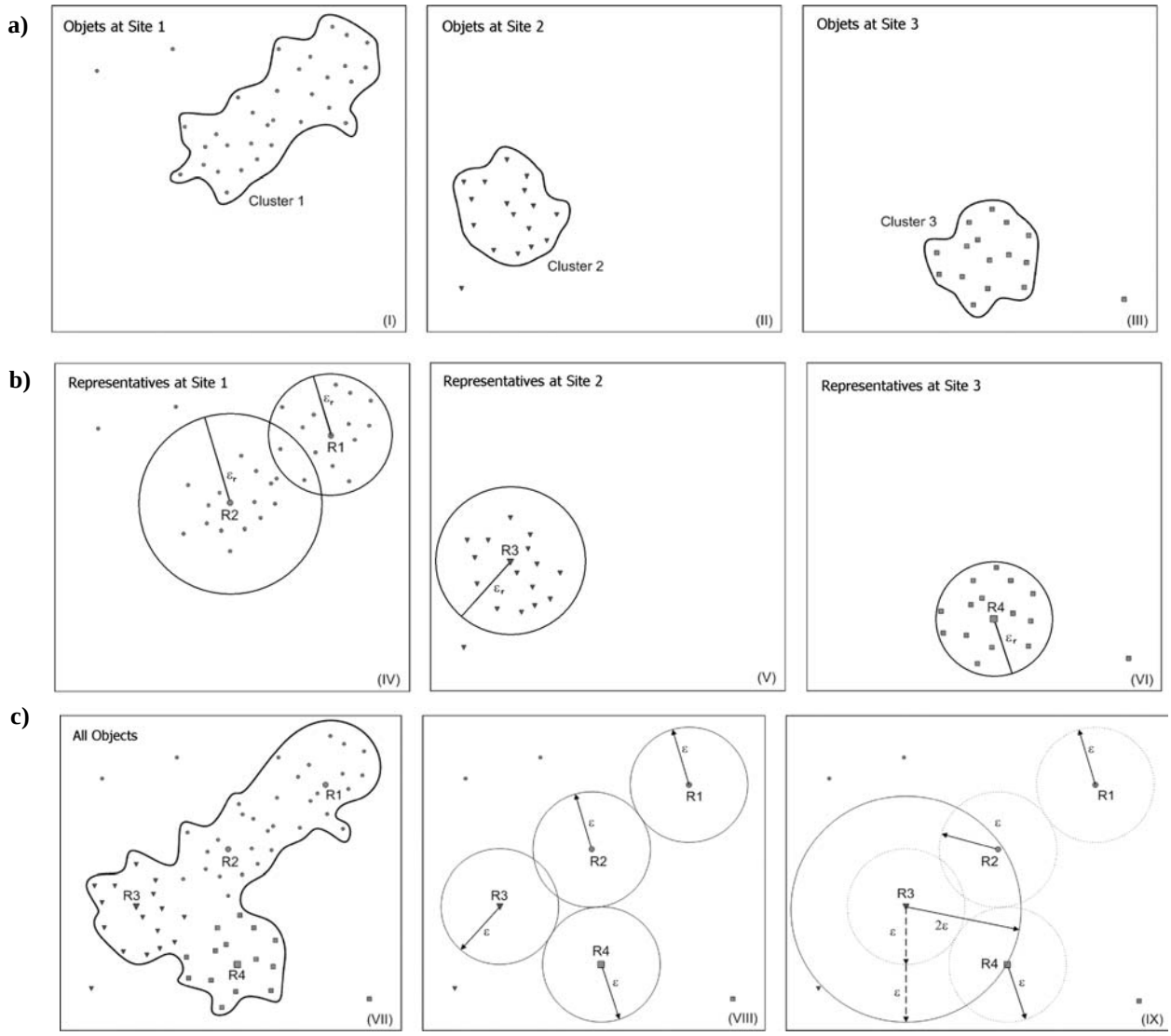

Figure 6: Determination of a global model
a) local clusters **b)** local representatives **c)** determination of a global model with $Eps_{global} = 2Eps_{local}$

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 a ε–range value ε_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:

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

Each of these pairs (r, ε_r) represent several objects which are all located in $N_{\varepsilon_r}(r)$, i.e. the ε_r-neighborhood of r. We regard all objects contained in $N_{\varepsilon_r}(r)$ as a own 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 together. 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. Thus, the question at issue is: "What are suitable $MinPts_{global}$- and Eps_{global}-parameters for a global DBSCAN run on the server site?".

As we assume that all local representatives form a cluster on their own it is enough to use a $MinPts_{global}$-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 us small Eps_{global} values, we might not be able to detect clusters belonging together. We think that the Eps_{global} parameter should be tunable by the user dependent on the ε_R values of all local representatives R. If these ε_R values are generally high it is advisable to use a high Eps_{global} value. On the other hand, if the ε_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 ε_R values of all local representatives R. This default Eps_{global} value is generally close to $2Eps_{local}$.

In Figure 6, an example for $Eps_{global}=2Eps_{local}$ is depicted. In Figure 6a the independently detected clusters on site 1,2 and 3 are depicted. The cluster on site 1 is represented by two representatives *R1* and *R2*, whereas the clusters on site 2 and site 3 are only represented by one representative as shown in Figure 6b. Figure 6c (VII) illustrates that all 4 clusters from the different sites belong to one large cluster. Figure 6c (VIII) makes clear 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 $2Eps_{local}$ the 4 representatives are merged together to one large cluster (cf. Figure 6c (IX)).

Instead of a user defined Eps_{global} parameter, we could also use a hierarchical density based clustering algorithm, e.g. OPTICS [ABKS 99], 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 refrain 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 8) of our DBDC algorithm is almost impossible. Third, the incremental version of DBSCAN allows us to start with the construction of the server model after the first representatives of any local model come in. So 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 sent the complete global model to all client sites. The client sites relable all objects located on their site independently from each other. On the client site, two former independent clusters may be merged together due to this new relabeling. Furthermore, points which were formerly assigned to local noise are now part of a global cluster. If a local point o is in the ε_r–range of a representative r, o is assigned to the same global cluster than r.

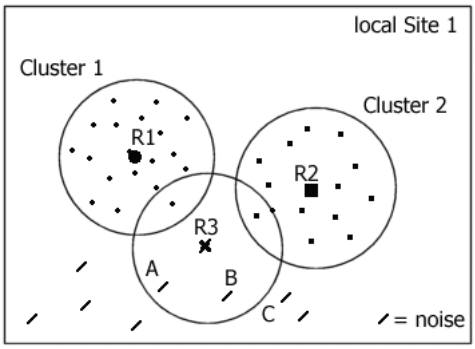

Figure 7: : Relabeling of the local clustering

Figure 7 depicts an example for this relabeling process. The points *R1* and *R2* are the local representatives. Each of them forms a cluster on its own. Point *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 ε_{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 as *"give me all objects on your site which belong to the global cluster 4711"*.

8. Experimental Evaluation

We evaluated our *DBDC*-approach based on different artificial 2-dimensional point sets. The point sets were generated on each local site independently. For the central reference clustering we used the union of the local point sets. As we suppose that this central clustering is optimal, we measure the quality of our *DBDC*-approach w.r.t. the central clustering. We varied both the number of points and the number of client sites. We compared *DBDC* to a single run of *DBSCAN* on all data points.

In order to evaluate our DBDC-algorithm, we carried out the local clusterings sequentially. We collected all representatives of all local runs, and then 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.

As there exist no general quality measure which helps to evaluate the quality of a distributed clustering, we defined one on our own. Let us assume that we have *n* points, distributed over *k* sites. Our *DBDC*-algorithm, assigned each of these *n* points, either to a cluster or to noise. We compare the result of our DBDC-algorithm to a central clustering of the *n* points with DBSCAN. Then we decide for each point *x* whether it was assigned correctly. We have to distinguish the following 5 cases where C_i and C_j denote different clusters:

Case	Clustering		quality
	distributed	central	
1	$x \in C_i$	$x \in C_i$	right
2	$x \in Noise$	$x \in Noise$	right
3	$x \in C_i$	$x \in C_j$	wrong
4	$x \in C_i$	$x \in Noise$	wrong
5	$x \in Noise$	$x \in C_i$	wrong

Let $n_{right} \leq n$ be the number of points which were assigned correctly. Then, the quality *q* of our DBDC-algorithm is computed as $q = n_{right} / n$.

In Figure 8 the overall runtime of our DBDC-algorithm is compared to the runtime of a central clustering of all points with DBSCAN. It is shown that our DBDC-approach outperforms a central clustering by far. Figure 8a shows that especially for large data sets our new approach performs much better. For instance for a point set consisting of 100,000 points, the DBDC outperforms the global DBSCAN algorithm for more than one order of magnitude. Figure 8b shows that an increasing number of sites goes hand in hand with an enhanced efficiency. From an efficiency point of view, our DBDC approach performs much better than a single clustering algorithm applied to the complete dataset.

Figure 9 shows that we do not have to sacrifice quality for this much better runtime behavior. We can see that the quality is above 98% for varying numbers of client sites and for both local models. Furthermore, the quality *q* does not considerably worsen with an increasing number of sites.

Figure 9: Quality of distributed clustering (203000 data points)

9. Conclusions and Future Work

In this paper, we first motivated the need of distributed clustering algorithms. As, to the best of our knowledge, there exist no algorithms which tackle this complex problem, we have developed an algorithm on our own 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 a ε–range value ε_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.

In an experimental evaluation we showed that our new distributed clustering approach yields almost the same clus-

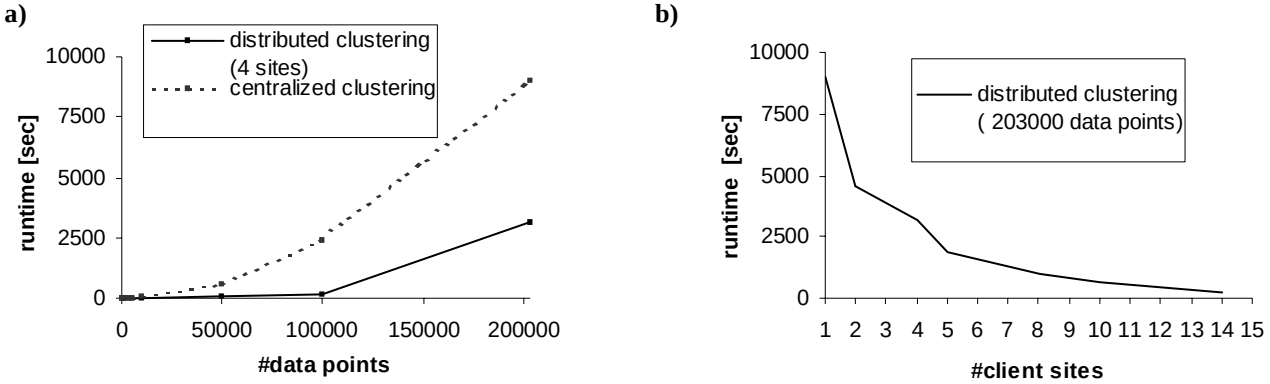

Figure 8: Overall runtime for central and distributed clustering
a) dependent on the data size **b)** dependent on the number of client sites

tering as a central clustering on all data. On the other hand, we showed that we have an enormous efficiency advantage compared to a classical clustering carried out on all data. Furthermore, 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 this data there. Therefore, we have to apply a distributed clustering algorithm, which can be very efficient and effective as shown in this paper.

Nevertheless, there are a lot of open problems which yield an enormous potential for further research activities:

- Can we produce precise error-estimations?
- How should we handle "*local noise*"? Let us assume that there exist more than *MinPts*-objects which are within an ε-distance to each other and distributed over different sites. If no site contains more than *MinPts* of these objects, a distributed algorithm might not detect a cluster.
- Do there exist better local models? Can we describe a local clustering with less points but more accurate? Such a local model should not be too complex, because we want to form a global model out of these local models efficiently.

As there are a lot of application ranges which would benefit from an efficient and effective distributed clustering algorithm, we think the open problems related to distributed clustering are worth to be carefully investigated.

References

[ABKS 99] Ankerst M., Breunig M. M., Kriegel H.-P., Sander J.: "OPTICS: Ordering Points To Identify the Clustering Structure", Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), Philadelphia, PA, 1999, pp. 49-60.

[AS 96] Agrawal R., Shafer J. C.: "Parallel mining of association rules: Design, implementation, and experience" IEEE Trans. Knowledge and Data Eng. 8 (1996) 962-969

[BKSS 90] 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.

[CPZ 97] Ciaccia P., Patella M., Zezula P.: "*M-tree: An Efficient Access Method for Similarity Search in Metric Spaces*", Proc. 23rd Int. Conf. on Very Large Data Bases, Athens, Greece, 1997, pp. 426-435.

[DM 99] Dhillon I. S., Modh Dh. S.: "A Data-Clustering Algorithm On Distributed Memory Multiprocessors", Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD 99)

[EKS+ 98] Ester M., Kriegel H.-P., Sander J., Wimmer M., Xu X.: "Incremental Clustering for Mining in a Data Warehousing Environment", VLDB 98

[EKSX 96] 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.

[ESK 03] 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)

[Han 00] 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

[Har 75] Hartigan J. A.: "Clustering Algorithms", Wiley, 1975

[HKK 97] 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

[JD 88] Jain A. K., Dubes R.C.: "Algorithms for Clustering Data", Prentice-Hall Inc., 1988.

[JMF 99] 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.

[KC 00] Kargupta H., Chan P. (editors) : "Advances in Distributed and Parallel Knowledge Discovery", AAAI/MIT Press, 2000

[KHS 97] Kargupta H., Hamzaoglu I., Stafford B.: "Scalable, Distributed Data Mining Using An Agent Based Architecture" Proceedings of Knowledge Discovery And Data Mining (1997). Eds: D. Heckerman, H. Mannila, D. Pregibon and R. Uthurusamy. AAAI Press. 211-214.

[KPHJ 00] Kargupta H., Park B., Hershberger D., Johnson E.: "Collective Data Mining: A New Perspective Toward Distributed Data Mining" Advances in Distributed and Parallel Knowledge Discovery (2000). Eds: Hillol Kargupta and Philip Chan. MIT/AAAI Press

[SAM 96] Shafer J., Agrawal R., Mehta M.: "A scalable parallel classifier for data mining" In: Proc. 22nd International Conference on VLDB, Mumbai, India. (1996)

[SHKS 98] 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)

[XJK 99] 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

[ZHA 98] Zaki M. J., Ho C.T., Agrawal R.: "Parallel classification for data mining on shared-memory multiprocessors" Technical report, IBM Almaden Research Center (1998)

[ZPOL 97] 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)