IASTED Int. Conf. on Databases and Applications (DBA'04)

A Quality Measure for Distributed Clustering

Hans-Peter Kriegel Eshref Januzaj Martin Pfeifle

Institute for Computer Science
University of Munich
Oettingenstr. 67, 80538 Munich, Germany
{kriegel, januzaj, pfeifle} @dbs.informatik.uni-muenchen.de

Abstract
Clustering has become an increasingly important task in modern application domains. Mostly, 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. Another approach is to cluster the data locally and extract suitable representatives from these clusters. Based on these representatives a global server tries to reconstruct the complete clustering. In this paper, we discuss the complex problem of finding a suitable quality measure for evaluating the quality of such a distributed clustering. We introduce a discrete and continuous quality criterion which we empirically compare to each other.

Keywords
Distributed Data Mining, partitioning clustering, quality measure for 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 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 unceasingly. Each of them is able to collect 1GB of data per hour [4] 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). One important part in DKDD is distributed clustering. The first step in distributed clustering consists of a local clustering of the data. Then aggregated information about the local clusters (also denoted as local representatives or local model) is sent to a central site. The transmission cost are minimal as the representatives are only a fraction of the original data. On the central site the global clustering is “reconstructed” .

A promising approach for distributed clustering is to use partitioning density-based clustering for the local and central clustering [7]. In this paper, we try to answer the complex question “how can we compare the quality of a partitioning distributed clustering to the quality of the corresponding central clustering?”.

The paper is organized as follows, in Section 2, we shortly review related work in the area of clustering. In Section 3, we sketch a Density Based Distributed Clustering (DBDC) algorithm which was used throughout the experiments. In Section 4, we introduce a discrete and continuous quality criterion allowing us to evaluate our DBDC approach. In Section 5, we present the experimental evaluation of the two introduced quality measures. We conclude the paper in Section 6 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. As, to our best knowledge, there exist no measures for evaluating the quality of distributed clusterings, we cannot present any related work regarding this topic.

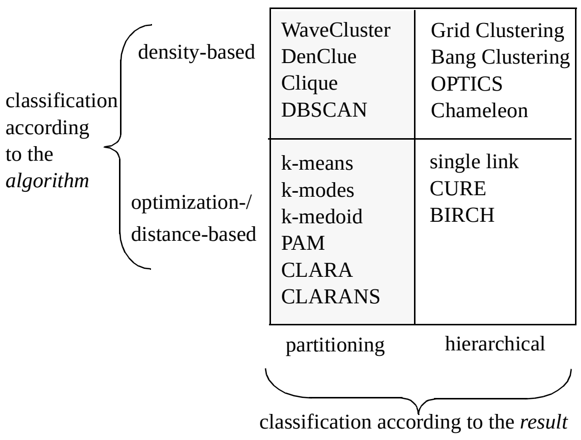


Figure 1: Classification scheme for clustering algorithms

2.1 Clustering

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 [6, 8]. 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* 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 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 [8] 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 [9] 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, there do not exist many algorithms for parallel and distributed clustering.

In [10] a parallel version of DBSCAN [3] and in [1] a parallel version of k-means [5] were introduced. Both algorithms start with the complete data set residing on one central sever and then distribute the data among the different clients.

On the other hand, distributed clustering assumes that 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. This distributed approach does usually not produce the same result as the central approach. In order to assess the quality of the distributedly produced result, we need a suitable quality criterion (cf. Section 4).

3 Distributed Clustering

In this section, we generally introduce distributed clustering and sketch our *DBDC* algorithm [7]. As this paper is about quality measures for distributed clusterings, and not about our *DBDC* algorithm, we refer the interested reader to [7] for more details.

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 local 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 clustering of the local models can be carried out much faster as the number of elements to be considered is much smaller.

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 perform 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 *DBDC* algorithm uses a local model which consists 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

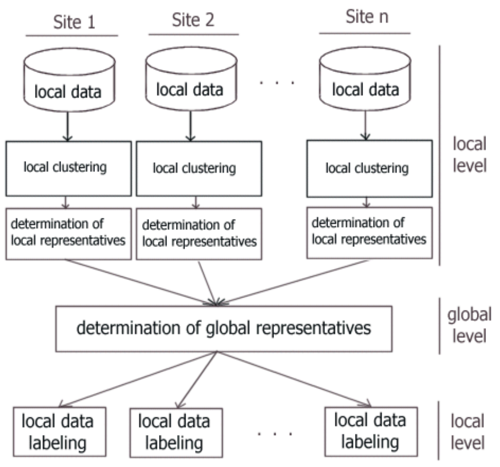


Figure 2: Distributed Clustering

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.

For our *DBDC* algorithm, we used the partitioning density-based clustering algorithm DBSCAN [3], for both the local clustering and the merging of the local models, because of the following reasons:

- 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 [2].
- 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 can easily be implemented.

4 Quality of Partitioning Distributed Clustering

To the best of our knowledge, there exists no general quality measure which helps to evaluate the quality of a distributed clustering. If we want to evaluate new partitioning distributed clustering approaches, 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, 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 partitioning clustering.

Definition 1 (partitioning clustering CL)
Let $D = \{x_1, \ldots, x_n\}$ be a database consisting of *n* objects. Then, we call any set CL a partitioning clustering of *D w.r.t. MinPts* ≥ 1, if it fulfills the following properties:

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

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

Let us assume that we have *n* objects, distributed over *k* sites. Partitioning distributed clustering algorithms assign each object *x*, either to a cluster or to noise. We compare the result of these algorithms to a central clustering of the *n* objects. 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 2 (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 partitioning 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*.

4.1 Discrete 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 3 (discrete object quality P^I)
Let $x \in D$ and let CL_{distr}, $CL_{central}$ be two clusterings. Then we can define an object quality function $P^I: D \rightarrow \{0, 1\}$ w.r.t. to a quality parameter *qp* as follows:

- If $(\exists C_d \in CL_{distr}: x \in C_d) \wedge (\exists C_c \in CL_{central}: x \in C_c) \wedge (|C_d \cap C_c| \geq qp)$, then $P^I(x) = 1$.
- If $(\exists C_d \in CL_{distr}: x \in C_d) \wedge (\exists C_c \in CL_{central}: x \in C_c) \wedge (|C_d \cap C_c| < qp)$, then $P^I(x) = 0$.
- If $(\exists C_d \in CL_{distr}: x \in C_d) \wedge (\forall C_c \in CL_{central}: x \notin C_c)$, then $P^I(x) = 0$.
- If $(\forall C_d \in CL_{distr}: x \notin C_d) \wedge (\forall C_c \in CL_{central}: x \notin C_c)$, then $P^I(x) = 1$.
- If $(\forall C_d \in CL_{distr}: x \notin C_d) \wedge (\exists C_c \in CL_{central}: x \in C_c)$, then $P^I(x) = 0$.

Table 1 explains the definition for the discrete object quality function P^I for the two cases $x \in C_d$ and $x \notin C_d$.

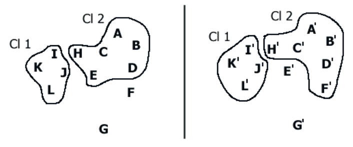
	explanation
$\in C_d$	If an object *x* is assigned to a cluster C_d during a distributed clustering algorithm, the quality P^I of this object *x* is 1 iff it is assigned to a cluster C_c by the global clustering algorithm. Furthermore, there have to be at least *qp* other objects which are also contained in C_d and C_c. If such an appropriate central cluster does not exist, the quality P^I of this object *x* is 0.
$\notin C_d$	If an object *x* is not assigned to a distributed cluster C_d during the DBDC run and furthermore $P^I(x) = 1$ holds, then it must not be contained in any central cluster C_c.

Table 1: Discrete object quality function P^I

Figure 3 shows two examples for this object quality function. The test data set were clustered with a partitioning density-based clustering algorithm where *MinPts* was set to 3. In this small example, we followed the reasoning of the beginning of this subsection and set parameter *qp* equal to *MinPts*. In Figure 3a, a "good" distributed clustering is depicted. The overall quality Q_{DBDC} w.r.t. P^I is high and thus reflects the "good" distributed clustering well. On the other hand, if we look at Figure 3b, the overall quality Q_{DBDC} w.r.t. P^I does not reflect the "bad" clustering. The distributed clustering quality Q_{DBDC} = 83,3% of Figure 3b is as high as the one for the "good" distributed clustering of Figure 3a.

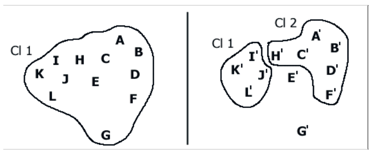
The main advantage of the object quality function P^I is that it is rather simple because it yields only a boolean return value, i.e. it tells whether an object was clustered correctly or falsely. Nevertheless, sometimes a more subtle quality measure is required which does not only assign a binary quality 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".

a) distributed clustering central clustering



	A	B	C	D	E	F	G	H	I	J	K	L	Quality Q_{DBDC}
$P^I(x)$ (*qp*=3)	1	1	1	1	0	0	1	1	1	1	1	1	83,3%
$P^{II}(x)$	$\frac{5}{7}$	$\frac{5}{7}$	$\frac{5}{7}$	$\frac{5}{7}$	0	0	1	$\frac{5}{7}$	1	1	1	1	71,4%

b) distributed clustering central clustering



	A	B	C	D	E	F	G	H	I	J	K	L	Quality Q_{DBDC}
$P^I(x)$ (*qp*=3)	1	1	1	1	0	1	0	1	1	1	1	1	83,3%
$P^{II}(x)$	$\frac{6}{12}$	$\frac{6}{12}$	$\frac{6}{12}$	$\frac{6}{12}$	0	$\frac{6}{12}$	0	$\frac{6}{12}$	$\frac{4}{12}$	$\frac{4}{12}$	$\frac{4}{12}$	$\frac{4}{12}$	36,1%

Figure 3: Quality of partitioning density-based distributed clustering with *MinPts*=3
a) "good" distributed clustering
b) "bad" distributed clustering

4.2 Continuous 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 4 (continuous object quality P^{II})
Let $x \in D$ and let CL_{distr}, $CL_{central}$ be a central and a distributed clustering. Then we define an object quality function $P^{II}: D \rightarrow [0, 1]$ as follows:

- If $(\exists C_d \in CL_{distr}: x \in C_d) \wedge (\exists C_c \in CL_{central}: x \in C_c)$, then $P^{II}(x) = \frac{|C_d \cap C_c|}{|C_d \cup C_c|}$.
- If $(\exists C_d \in CL_{distr}: x \in C_d) \wedge (\forall C_c \in CL_{central}: x \notin C_c)$, then $P^{II}(x) = 0$.
- If $(\forall C_d \in CL_{distr}: x \notin C_d) \wedge (\forall C_c \in CL_{central}: x \notin C_c)$, then $P^{II}(x) = 1$.
- If $(\forall C_d \in CL_{distr}: x \notin C_d) \wedge (\exists C_c \in CL_{central}: x \in C_c)$, then $P^{II}(x) = 0$.

Table 2 explains the definition for the continuous object quality function P^{II} for the two cases $x \in C_d$ and $x \notin C_d$.

	explanation
$\in C_d$	If an object x is assigned to a cluster C_d during a distributed clustering algorithm, the quality P^{II} of this object x is somewhere between 0 and 1. It is close to 1 iff the number of elements contained in $C_d \cap C_c$ is almost as high as the number of elements contained in $C_d \cup C_c$.
$\notin C_d$	Similar to the case for the discrete object quality function P^I.

Table 2: Continuous object quality function P^{II}

Figure 3 shows that our new object quality function P^{II} much better reflects the quality of a distributed clustering than P^I. The "good" clustering of Figure 3a yields a higher quality value than the "bad" clustering of Figure 3b. In the experimental evaluation, we will provide further evidence for the superiority of the second more sophisticated quality criterion.

5 Experimental Evaluation

We evaluated our quality criteria by applying our *DBDC*-approach to three different 2-dimensional point sets with varying number of points and characteristics. Figure 4 depicts the three used test data sets *A*, *B* and *C* on the central site. 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. If not stated otherwise, the local and the global run of DBSCAN was carried out with a *MinPts* parameter of 4 which was also used as a default parameter for the quality parameter *qp* of P^I.

In the first set of experiments, we evaluated the quality of our two introduced object quality functions P^I and P^{II}, dependent on the parameter setting of our *DBDC* approach. In numerous manually performed tests, we found out that the *DBDC* approach, works best with a "normalized parameter setting" [7] somewhere between 2 and 3. Figure 5 shows that the quality according to P^I of the DBDC approach is very high and does not change if we vary the parameters. On the other hand, we can clearly see that for a normalized parameter of 2, we get the best quality according to P^{II}. The quality graph produced by P^{II} reflects that for small parameters a lot of local clusters are not globally merged together. Furthermore, for high values it mirrors the fact that a lot of noise is assigned to clusters. To sum up, for high and low distributed clustering parameters our *DBDC* approach performs bad which is accurately pinpointed by P^{II} but not detected by P^I.

In the next set of experiments, we evaluated the quality of our *DBDC* algorithm for a noisy test data set *B* and for an extremely well clustered test data set *C*. Our DBDC approach performs very bad for the first kind of data set, whereas it has no problems to detect the clusters in the data set *C*. As the quality function P^I depends additionally on a quality parameter *qp*, we varied this parameter. Figure 6a shows that for small *qp* parameters P^I falsely assigns a very high quality to the distributed clustering of test data set B. Only, for very high *qp* parameters it detects the low quality produced by DBDC for test data set B. For test data set C,

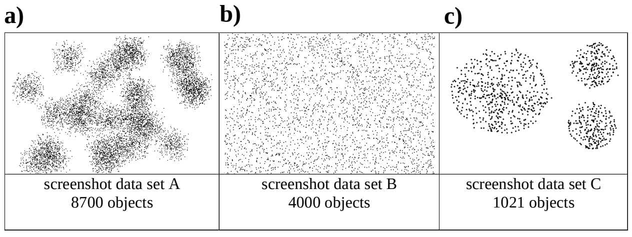


Figure 4: Used test data sets
a) test data set *A* **a)** test data set *B* **a)** test data set *C*

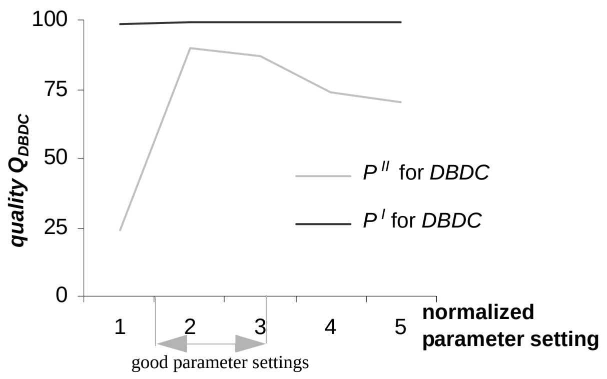
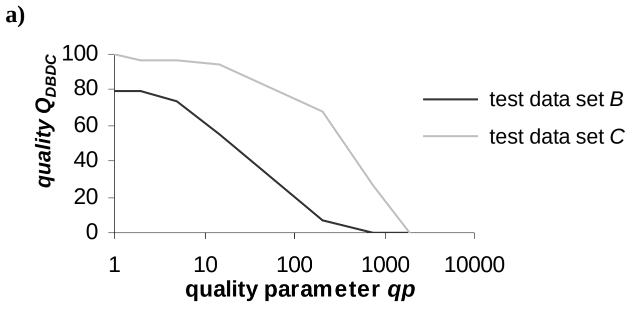


Figure 5: Evaluation of object quality functions for varying normalized parameter settings [7] for data set *A* on 4 local sites



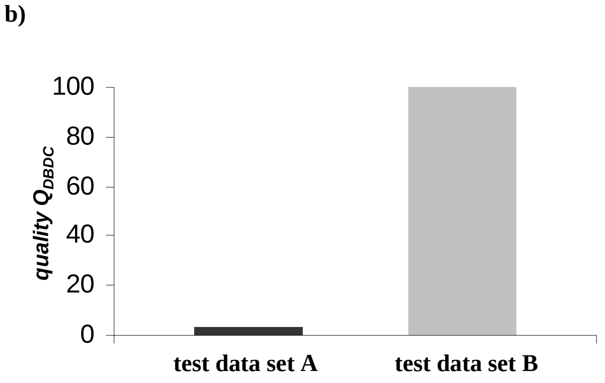


Figure 6: Quality of the DBDC-algorithm for test data set *B* and test data set *C on 4 local sites*
a) quality function P^I w.r.t. to *qp* **b)** quality function P^{II}

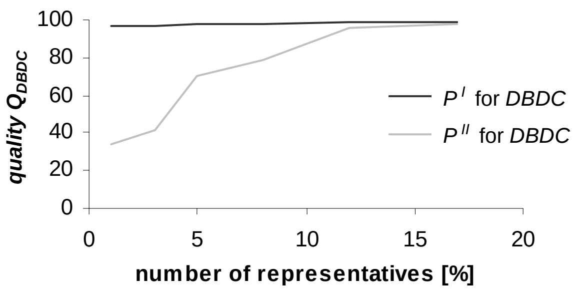


Figure 7: Quality of the *DBDC*-algorithm for a varying number of local representatives (data set A, 4 local sites)

P^I reflects the high quality of our distributed clustering only for small values, whereas for high values it malfunctions. On the other hand, P^{II} detects the correct qualities for both test data sets (cf. Figure 6b).

Figure 7 shows how the quality depends on the number of transmitted local representatives. Our DBDC approach works the more accurate, the more local representatives are transmitted. Again, our continuous object function P^{II} reflects this behavior properly, whereas the discrete object function P^I fails to point out the dependency between the quality and the number of representatives.

Figure 8 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.

To sum up, for some parameter settings and some data sets the discrete object quality function P^I produces suitable quality values. Unfortunately, the required conditions often do not hold. On the other hand, our continuous object quality function P^{II} reflects the correct quality for all kind of data sets with varying number of points and characteristics. Furthermore, it assigns the correct quality to our DBDC approach for a varying number of local representatives, local sites, and parameter settings of our distributed clustering algorithm.

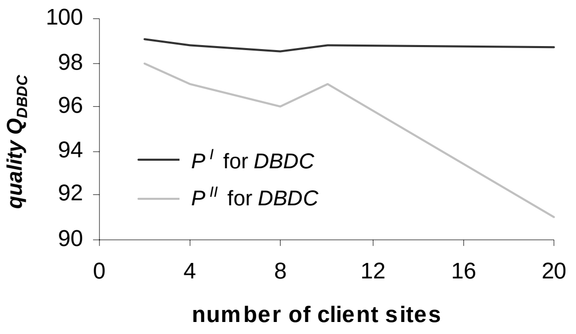


Figure 8: Quality Q_{DBDC} dependent on the number of client sites for the test data set *A*

6 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 efficient and effective distributed clustering algorithms. We sketched 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 where a global clustering was carried out. As there exist no general quality measures which help to evaluate the quality of a distributed clustering, we introduced a discrete quality function P^I and a continuous quality function P^{II}. In the experimental evaluation, we discussed the suitability of our quality functions and showed that the continuous object quality function P^{II} reflects the intuitive quality which an experienced user would assign to a distributed clustering. Based on this quality criterion it is possible to evaluate further distributed clustering algorithms from which a lot of application ranges will benefit.

References

[1] Dhillon I. S., Modh Dh. S.: "A Data-Clustering Algorithm On Distributed Memory Multiprocessors", SIGKDD 99

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

[3] 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.

[4] 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

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

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

[7] Januzaj E., Kriegel H.-P., Pfeifle M.: "DBDC: Density Based Distributed Clustering", Technical Report, University of Munich, 2003.

[8] 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.

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

[10] 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