Knowledge and Information Systems (KAIS), Vol. 9, No. 3, 2006.
The original publication is available at www.springerlink.com.

Multi-Step Density-Based Clustering

Stefan Brecheisen, Hans-Peter Kriegel, and Martin Pfeifle

Institute for Informatics, University of Munich
Oettingenstr. 67, 80538 Munich, Germany
{brecheis,kriegel,pfeifle}@dbs.ifi.lmu.de

Abstract. Data mining in large databases of complex objects from scientific, engineering or multimedia applications is getting more and more important. In many areas, complex distance measures are first choice but also simpler distance functions are available which can be computed much more efficiently. In this paper, we will demonstrate how the paradigm of multi-step query processing which relies on exact as well as on lower-bounding approximated distance functions can be integrated into the two density-based clustering algorithms DBSCAN and OPTICS resulting in a considerable efficiency boost. Our approach tries to confine itself to ε-range queries on the simple distance functions and carries out complex distance computations only at that stage of the clustering algorithm where they are compulsory to compute the correct clustering result. Furthermore, we will show how our approach can be used for approximated clustering allowing the user to find an individual trade-off between quality and efficiency. In order to assess the quality of the resulting clusterings, we introduce suitable quality measures which can be used generally for evaluating the quality of approximated partitioning and hierarchical clusterings. In a broad experimental evaluation based on real-world test data sets, we demonstrate that our approach accelerates the generation of exact density-based clusterings by more than one order of magnitude. Furthermore, we show that our approximated clustering approach results in high quality clusterings where the desired quality is scalable w.r.t. the overall number of exact distance computations.

1 Introduction

In recent years, the research community spent a lot of attention to the clustering problem resulting in a large variety of different clustering algorithms [1]. One important class of clustering algorithms is density-based clustering which can be used for clustering all kinds of metric data and is not confined to vector spaces. Density-based clustering is rather robust concerning outliers [2] and is very effective in clustering all sorts of data, e.g. multi-represented objects [3]. Furthermore, the reachability plot created by the density-based hierarchical clustering algorithm OPTICS [4] serves as a starting point for an effective data mining tool which helps to visually analyze cluster hierarchies [5].

Density-based clustering algorithms like DBSCAN [2] and OPTICS [4] are based on ε-range queries for each database object. Each range query requires a

lot of distance calculations, especially when high ε-values are used. Therefore, these algorithms are only applicable to large collections of complex objects, e.g. trees, point sets, and graphs (cf. Figure 1), if those range queries are supported efficiently. When working with complex objects, the necessary distance calculations are the time-limiting factor. Thus, the ultimate goal is to save as many of these complex distance calculations as possible.

In this paper, we will present an approach which helps to compute density-based clusterings efficiently. The core idea of our approach is to integrate the multi-step query processing paradigm directly into the clustering algorithm rather than using it "only" for accelerating range queries. Our clustering approach itself exploits the information provided by simple distance measures lower-bounding complex and expensive exact distance functions. Expensive exact distance computations are only performed when the information provided by simple distance computations, which are often based on simple object representations, is not enough to compute the exact clustering. Furthermore, we show how our approach can be used for approximated clustering where the result might be slightly different from the one we compute based on the exact information. In order to measure the dissimilarity between the resulting clusterings, we introduce suitable quality measures.

The remainder of this paper is organized as follows: In Section 2, we present our new approach which integrates the multi-step query processing paradigm directly into the clustering algorithms rather than using it independently. As our approach can also be used for generating approximated clusterings, we introduce objective quality measures in Section 3 which allow us to assess the quality of approximated clusterings. In Section 4, we present a detailed experimental evaluation showing that the presented approach can accelerate the generation of density-based clusterings on complex objects by more than one order of magnitude. We show that for approximated clustering the achieved quality is scalable w.r.t. the overall runtime. We close this paper, in Section 5, with a short summary and a few remarks on future work.

2 Efficient Density-Based Clustering

In this section, we will discuss in detail how we can efficiently compute a flat (DBSCAN) and a hierarchical (OPTICS) density-based clustering. First, in Section 2.1, we present the basic concepts of density-based clustering along with the two algorithms DBSCAN and OPTICS. Then, in Section 2.2, we look at different approaches presented in the literature for efficiently computing these algorithms. We will explain why the presented algorithms are not suitable for expensive distance computations if we are interested in the exact clustering structure. In Section 2.3, we will present our new approach which tries to use lower-bounding distance functions before computing the expensive exact distances.

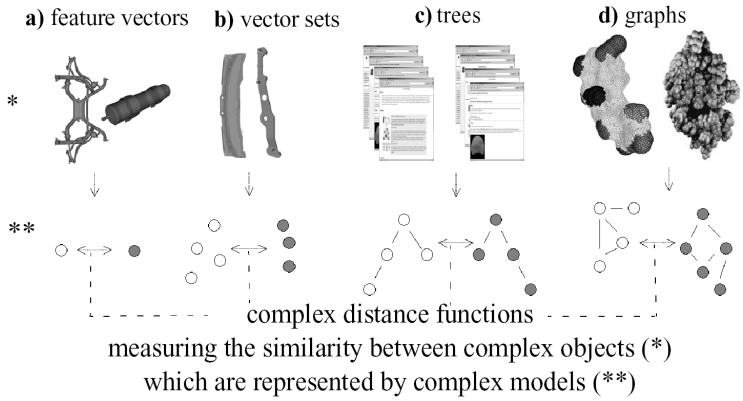


Fig. 1. Complex objects.

2.1 Density-based Clustering

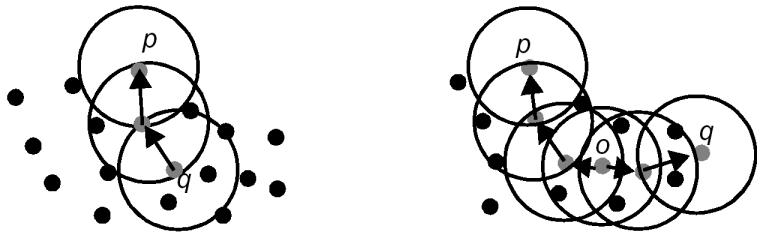
The key idea of density-based clustering is that for each object of a cluster the neighborhood of a given radius ε has to contain at least a minimum number *MinPts* of objects, i.e. the cardinality of the neighborhood has to exceed a given threshold. In the following, we will present the basic definitions of density-based clustering.

Definition 1 (directly density-reachable). *Object p is* directly density-reachable *from object q w.r.t. ε and MinPts in a set of objects DB, if $p \in N_\varepsilon(q)$ and $|N_\varepsilon(q)| \geq MinPts$, where $N_\varepsilon(q)$ denotes the subset of DB contained in the ε-neighborhood of q.*

The condition $|N_\varepsilon(q)| \geq MinPts$ is called the core object condition. If this condition holds for an object q, then we call q a core object. Other objects can be directly density-reachable only from core objects.

Definition 2 (density-reachable and density-connected). *An object p is* density-reachable *from an object q w.r.t. ε and MinPts in a set of objects DB, if there is a chain of objects $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$, such that $p_i \in DB$ and p_{i+1} is directly density-reachable from p_i w.r.t. ε and MinPts. Object p is* density-connected *to object q w.r.t. ε and MinPts in a set of objects DB, if there is an object $o \in DB$, such that both p and q are density-reachable from o in DB w.r.t. ε and MinPts.*

Density-reachability is the transitive closure of direct density-reachability and does not have to be symmetric. On the other hand, density-connectivity is symmetric (cf. Figure 2).



(a) p density-reachable from q, but q not density-reachable from p.

(b) p and q density-connected to each other by o.

Fig. 2. Density-reachability and density-connectivity.

DBSCAN A flat density-based cluster is defined as a set of density-connected objects which is maximal w.r.t. density-reachability. Then the noise is the set of objects not contained in any cluster. A cluster contains not only core objects but also objects that do not satisfy the core object condition. These border objects are directly density-reachable from at least one core object of the cluster.

The algorithm DBSCAN [2], which discovers the clusters and the noise in a database, is based on the fact that a cluster is equivalent to the set of all objects in DB which are density-reachable from an arbitrary core object in the cluster (cf. lemma 1 and 2 in [2]). The retrieval of density-reachable objects is performed by iteratively collecting directly density-reachable objects. DBSCAN checks the ε-neighborhood of each point in the database. If the ε-neighborhood $N_\varepsilon(q)$ of a point q has more than *MinPts* elements, q is a so-called core point, and a new cluster C containing the objects in $N_\varepsilon(q)$ is created. Then, the ε-neighborhood of all points p in C which have not yet been processed is checked. If $N_\varepsilon(p)$ contains more than *MinPts* points, the neighbors of p which are not already contained in C are added to the cluster and their ε-neighborhood is checked in the next step. This procedure is repeated until no new point can be added to the current cluster C. Then the algorithm continues with a point which has not yet been processed trying to expand a new cluster.

OPTICS While the partitioning density-based clustering algorithm DBSCAN [2] can only identify a "flat" clustering, the newer algorithm OPTICS [4] computes an ordering of the points augmented by additional information, i.e. the reachability-distance, representing the intrinsic hierarchical (nested) cluster structure. The result of OPTICS, i.e. the cluster ordering, is displayed by the so-called reachability plots which are 2D-plots generated as follows: the clustered objects are ordered along the x-axis according to the cluster ordering computed by OPTICS and the reachabilities assigned to each object are plotted along the abscissa. An example reachability plot is depicted in Figure 3. Valleys in this plot

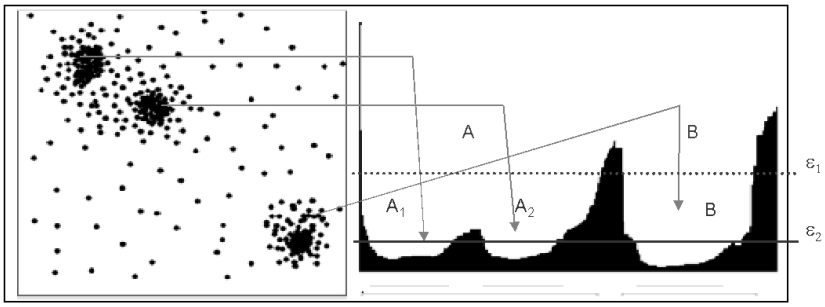


Fig. 3. Reachability plot (right) computed by OPTICS for a 2D data set (left).

indicate clusters: objects having a small reachability value are closer and thus more similar to their predecessor objects than objects having a higher reachability value. Thus, it is possible to explore interactively the clustering structure, offering additional insights into the distribution and correlation of the data.

In the following, we will shortly introduce the definitions underlying the OPTICS algorithm, the core-distance of an object p and the reachability-distance of an object p w.r.t. a predecessor object o.

Definition 3 (core-distance). *Let p be an object from a database DB, let $N_\varepsilon(p)$ be the ε-neighborhood of p, let MinPts be a natural number and let MinPts-dist(p) be the distance of p to its MinPts-th neighbor. Then, the* core-distance *of p, denoted as core-dist$_{\varepsilon,MinPts}(p)$ is defined as MinPts-dist(p) if $|N_\varepsilon(p)| \geq$ MinPts and INFINITY otherwise.*

Definition 4 (reachability-distance). *Let p and o be objects from a database DB, let $N_\varepsilon(o)$ be the ε-neighborhood of o, let dist(o,p) be the distance between o and p, and let MinPts be a natural number. Then the* reachability-distance *of p w.r.t. o, denoted as reachability-dist$_{\varepsilon,MinPts}(p,o)$, is defined as* $\max(\textit{core-dist}_{\varepsilon,MinPts}(o), \textit{dist}(o,p))$.

The OPTICS algorithm (cf. Figure 4) creates an ordering of a database, along with a reachability-value for each object. Its main data structure is a *seedlist*, containing tuples of points and reachability-distances. The seedlist is organized w.r.t. ascending reachability-distances. Initially the seedlist is empty and all points are marked as *not-done*.

The procedure update-seedlist(o_1) executes an ε-range query around the point o_1, i.e. the first object of the sorted seedlist, at the beginning of each cycle. For every point p in the result of the range query, it computes $r = \textit{reachability-dist}_{\varepsilon,MinPts}(p,o_1)$. If the seedlist already contains an entry (p,s), it is updated to $(p,\min(r,s))$, otherwise (p,r) is added to the seedlist. Finally, the order of the seedlist is reestablished.

```
Algorithm OPTICS:
  repeat {
    if the seedlist is empty {
      if all points are marked "done", terminate;
      choose "not-done" point q;
      add (q, INFINITY) to the seedlist;
    }
    (o1, r) = seedlist entry having the smallest reachability value;
    remove (o1, r) from seedlist;
    mark o1 as "done";
    output (o1, r);
    update-seedlist(o1);
  }
```

Fig. 4. The OPTICS algorithm.

2.2 Related Work

DBSCAN and OPTICS determine the local densities by *repeated range queries*. In this section, we will sketch different approaches from the literature to accelerate these density-based clustering algorithms and discuss their unsuitability for complex object representations.

Exact Clustering In the following we will present some approaches leading to exact density-based clusterings.

Multi-Dimensional Index Structures. The most common approach to accelerate each of the required single range queries is to use multi-dimensional index structures. For objects modelled by low-, medium-, or high-dimensional feature vectors there exist several specific R-tree [6] variants. For more detail we refer the interested reader to [7].

Metric Index Structures. In contrast to Figure 1a where the objects are modelled by a high-dimensional feature vector, the objects presented in the example of Figure 1b-1d are not modelled by feature vectors. Therefore, we cannot apply the index structures mentioned in the last paragraph. Nevertheless, we can use index structures, such as the M-tree [8] for efficiently carrying out range queries as long as we have a metric distance function for measuring the similarity between two complex objects. For a detailed survey on metric access methods we refer the reader to [9].

Multi-Step Query Processing. The main goal of multi-step query processing is to reduce the number of complex and, therefore, time consuming distance calculations in the query process. In order to guarantee that there occur no false drops, the used filter distances have to fulfill a lower-bounding distance criterion.

For any two objects p and q, a lower-bounding distance function d_f in the filter step has to return a value that is not greater than the exact object distance d_o of p and q, i.e. $d_f(p,q) \leq d_o(p,q)$. With a lower-bounding distance function it is possible to safely filter out all database objects which have a filter distance greater than the current query range because the exact object distance of those objects cannot be less than the query range. Using a multi-step query architecture requires efficient algorithms which actually make use of the filter step. Agrawal, Faloutsos and Swami proposed such an algorithm for range queries [10] which form the foundation of density-based clustering. For efficiency reasons, it is crucial that $d_f(p,q)$ is considerably faster to evaluate than $d_o(p,q)$ and, furthermore, in order to achieve a high selectivity $d_f(p,q)$ should be only marginally smaller than $d_o(p,q)$.

Using Multiple Similarity Queries. In [11] a schema was presented which transforms query intensive KDD algorithms into a representation using the similarity join as a basic operation without affecting the correctness of the result of the considered algorithm. The approach was applied to accelerate the clustering algorithms DBSCAN and OPTICS by using an R-tree like index structure. In [12] an approach was introduced for efficiently supporting multiple similarity queries for mining in metric databases. It was shown that many different data mining algorithms can be accelerated by multiplexing different similarity queries.

Summary. Multi-dimensional index structures based on R-tree variants and clustering based on the similarity join are restricted to vector set data. Furthermore, the main problem of all approaches mentioned above is that distance computations can only be avoided for objects located outside the ε-range of the actual query object. In order to create, for instance, a reachability plot without loss of information, the authors in [4] propose to use a very high ε-value. Therefore, all of the above mentioned approaches lead to $O(|DB|^2)$ exact distance computations for OPTICS.

Approximated Clustering Other approaches do not aim at producing the exact hierarchical clustering structure, but an approximated one.

Sampling. The simplest approach is to use sampling and apply the expensive data mining algorithms to a subset of the dataspace. Typically, if the sample size is large enough, the result of the data mining method on the sample reflects the exact result well.

Grid-Based Clustering. Another approach is based on grid cells [1] to accelerate query processing. In this case, the data space is partitioned into a number of non-overlapping regions or cells which can be used as a filter step for the range queries. All points in the result set are contained in the cells intersecting the query range. To further improve the performance of the range queries to a constant time complexity, query processing is limited to a constant number of these cells (e.g. the cell covering the query point and the direct neighbor cells) and the refinement step is dropped, thereby trading accuracy for performance.

Distance Mapping. In [13], 5 different distance-mapping algorithms were introduced to map general metric objects to Euclidean or pseudo-Euclidean spaces in such a way that the distances among the objects are approximately preserved. The approximated data mining algorithm is then performed within the Euclidean space based on rather cheap distance functions. If there already exist selective filters which can efficiently be computed, an additional mapping into a feature space is superfluous, i.e. we can carry out the approximated data mining algorithm directly on the filter information.

Data Bubbles. Finally, there exist efficient approximated versions of hierarchical clustering approaches for non-vector data which are based on Data Bubbles [14]. These approaches augment suitable representatives with additional aggregated information describing the area around the representatives.

Summary. All indicated approximated clustering approaches are able to generate efficiently the corresponding clustering structure. The question at issue is: How much quality do they have to pay for their efficiency gain?

In this paper, we will propose an approach which computes exact density-based clusterings trying to confine itself to simple distance computations lower-bounding the exact distances. Further expensive exact distance computations are postponed as long as possible, and are only carried out at that stage of the algorithm where they are compulsory to compute the correct clustering. Furthermore, we will also indicate how to use our algorithm for approximated clustering.

2.3 Accelerated Density-Based Clustering

In this section, we will demonstrate how to integrate the multi-step query processing paradigm into the two density-based clustering algorithms DBSCAN and OPTICS. We discuss in detail our approach for OPTICS and sketch how a simplified version of this extended OPTICS approach can be used for DBSCAN.

Basic Idea DBSCAN and OPTICS are both based on numerous ε-range queries. None of the approaches discussed in the literature can avoid that we have to compute the exact distance to a given query object q for all objects contained in $N_\varepsilon(q)$. Especially for OPTICS, where ε has to be chosen very high in order to create reachability plots without loss of information, we have to compute $|DB|$ many exact distance computations for each single range query, even when one of the methods discussed in Section 2.2 is used. In the case of DBSCAN, typically, the ε-values are much smaller. Nevertheless, if we apply the traditional multi-step query processing paradigm with non-selective filters, we also have to compute up to $|DB|$ many exact distance computations.

In our approach, the number of exact distance computations does not primarily depend on the size of the database and the chosen ε-value but rather on the value of *MinPts*, which is typically only a small fraction of $|DB|$, e.g.

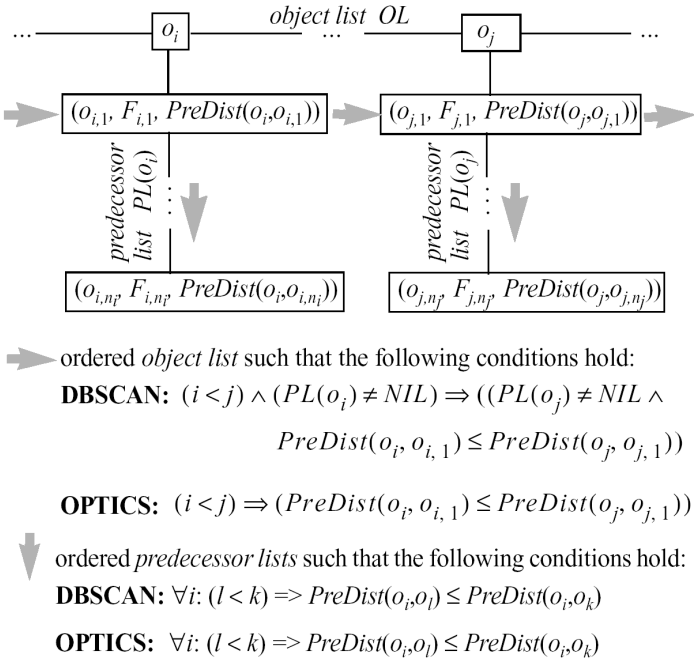


Fig. 5. Data structure Xseedlist.

$MinPts = 5$ is a suitable value even for large databases [4, 2]. Basically, we use $MinPts$-nearest neighbor queries instead of ε-range queries on the exact object representations in order to determine the "core-properties" of the objects. Further exact complex distance computations are only carried out at that stage of the algorithms where they are compulsory to compute the correct clustering result.

Extended OPTICS The main idea of our approach is to carry out the range queries based on the lower-bounding filter distances instead of using the expensive exact distances. In order to put our approach into practice, we have to slightly extend the data structure underlying the OPTICS algorithm, i.e. we have to add additional information to the elements stored in the seedlist.

The Extended Seedlist. We do not any longer use a single seedlist as in the original OPTICS algorithm (cf. Figure 4) where each list entry consists of a pair $(ObjectId, ReachabilityValue)$. Instead, we use a list of lists, called *Xseedlist*, as shown in Figure 5. The Xseedlist consists of an ordered object list OL, quite similar to the original seedlist but without any reachability information. The order of the objects o_i in OL, cf. the horizontal arrow in Figure 5, is determined

by the first element of each predecessor list $PL(o_i)$ anchored at o_i, cf. the vertical arrows in Figure 5.

An entry located at position l of the predecessor list $PL(o_i)$ belonging to object o_i consists of the following information:

- **Predecessor ID.** A processed object $o_{i,l}$ which was already added to the reachability plot which is computed from left to right.
- **Predecessor Flag.** A flag $F_{i,l}$ indicating whether we already computed the exact object distance $d_o(o_i, o_{i,l})$ between o_i and $o_{i,l}$, or whether we only computed the distance $d_f(o_i, o_{i,l})$ of these two objects based on the lower-bounding filter information.
- **Predecessor Distance.** $\mathit{PreDist}(o_i, o_{i,l})$ is equal to

$$\max(\mathit{core\text{-}dist}_{\varepsilon,\mathit{MinPts}}(o_{i,l}), d_o(o_i, o_{i,l})),$$

  if we already computed the exact object distance $d_o(o_i, o_{i,l})$, else it is equal to

$$\max(\mathit{core\text{-}dist}_{\varepsilon,\mathit{MinPts}}(o_{i,l}), d_f(o_i, o_{i,l})).$$

Throughout our new algorithm, the conditions depicted in Figure 5 belonging to this extended OPTICS algorithm are maintained. In the following, we will describe the extended OPTICS algorithm trying to minimize the number of exact distance computations.

Algorithm. The extended OPTICS algorithm exploiting the filter information is depicted in Figure 6. The algorithm always takes the first element o_1 from OL. If it is at the first position due to a filter computation, we compute the exact distance $d_o(o_1, o_{1,1})$ and reorganize the Xseedlist. The reorganization might displace $o_{1,1}$ from the first position of $PL(o_1)$. Furthermore, object o_1 might be removed from the first position of OL. On the other hand, if the filter flag $F_{1,1}$ indicates that an exact distance computation was already carried out, we add object o_1 to the reachability plot with a reachability-value equal to $\mathit{PreDist}(o_1, o_{1,1})$. Furthermore, we carry out the procedure update-Xseedlist(o_1).

Update-Xseedlist. This is the core function of our extended OPTICS algorithm. First, we carry out a range query around the query object $q := o_1$ based on the filter information, yielding the result set $N_\varepsilon^{\mathit{filter}}(q)$. Then we compute the core-distance of q by computing the *MinPts*-nearest neighbors of q as follows:

- If $|N_\varepsilon^{\mathit{filter}}(q)| < \mathit{MinPts}$, we set the core-distance of q to *INFINITY* and we are finished. Otherwise, we initialize a list $\mathit{SortList}_\varepsilon(q)$ containing tuples $(obj, flag, dist)$ which are organized in ascending order according to *dist*. For all objects $o \in N_\varepsilon^{\mathit{filter}}(q)$, we insert an entry $(o, \text{Filter}, d_f(o, q))$ into $\mathit{SortList}_\varepsilon(q)$.
- We walk through $\mathit{SortList}_\varepsilon(q)$ starting at the first element. We set

$$\mathit{SortList}_\varepsilon(q)[1].dist = d_o(\mathit{SortList}_\varepsilon(q)[1].obj, q),$$
$$\mathit{SortList}_\varepsilon(q)[1].flag = \text{Exact},$$

and reorder $SortList_\varepsilon(q)$. This step is repeated until the first *MinPts* elements of $SortList_\varepsilon(q)$ are at their final position due to an exact distance computation. The core-distance of q is equal to the distance

$$dist_{MinPts} = SortList_\varepsilon(q)[MinPts].dist,$$

if $dist_{MinPts} \leq \varepsilon$ holds, else it is set to *INFINITY*.

A tuple $(obj_j, flag_j, dist_j) \in SortList_\varepsilon(q)$ is transferred into an Xseedlist entry, if q is a core object and $dist_j \leq \varepsilon$ holds. If there exists no entry for obj_j in OL, $(obj_j, \langle(q, flag_j, \max(dist_j, core\text{-}dist_{\varepsilon,MinPts}(q)))\rangle)$ is inserted into OL, else $(q, flag_j, \max(dist_j, core\text{-}dist_{\varepsilon,MinPts}(q)))$ is inserted into $PL(obj_j)$. Note that in both cases the ordering of Figure 5 has to be maintained.

Lemma 1. *The result of the extended OPTICS algorithm is equivalent to the result of the original one.*

Proof. First, the extended OPTICS algorithm computes the correct core-distances by applying a *MinPts*-nearest neighbor search algorithm. Second, in each cycle the extended and the original OPTICS algorithm add the object o_1 having the minimum reachability-distance, w.r.t. all objects reported in the foregoing steps, to the cluster ordering. For the extended OPTICS algorithm this is true, as we have computed $d_o(o_1, o_{1,1})$ before adding it to the cluster ordering, due to the ordering conditions of Figure 5, and due to the lower-bounding filter property.

Note that this approach carries out exact distance computations only for those objects which are very close to the current query object q according to the filter information, whereas the traditional multi-step query approach would compute exact distance computations for all objects $o \in N_\varepsilon^{filter}(q)$. As ε has to be chosen very high in order to create reachability plots without loss of information [4], the traditional approach has to compute $|DB|$ many exact distance computations, even when one of the approaches discussed in Section 2.2 is used. On the other hand, the number of exact distance computations in our approach does not depend on the size of the database but rather on the value of *MinPts*, which is only a small fraction of the cardinality of the database. Note that our approach only has to compute $|DB| \cdot MinPts$, i.e. $O(|DB|)$, exact distance computations if we assume an optimal filter, in contrast to the $O(|DB|^2)$ distance computations carried out by the original OPTICS run. Only when necessary, we carry out further exact distance computations (cf. line (*) in Figure 6).

Extended DBSCAN Our extended DBSCAN algorithm is a simplified version of the extended OPTICS algorithm also using the Xseedlist as its main data structure. We carry out an ε-range query on the lower-bounding filter distances for an arbitrary database object q which has not yet been processed. Due to the lower-bounding properties of the filters, $N_\varepsilon(q) \subseteq N_\varepsilon^{filter}(q)$ holds. Therefore, if $|N_\varepsilon^{filter}(q)| < MinPts$, q is certainly no core point. Otherwise, we test whether q is a core point as follows.

```
Algorithm OPTICS:
  repeat {
    if the Xseedlist is empty {
      if all points are marked "done", terminate;
      choose "not-done" point q;
      add (q, empty_list) to the Xseedlist;
    }
    (o_1, list) = first entry in the Xseedlist;
    if list[1].PredecessorFlag == Filter{
      compute d_o(o_1, list[1].PredecessorID);                    (*)
      update list[1].PredecessorDistance;
      list[1].PredecessorFlag = Exact;
      reorganize Xseedlist according to the two conditions of Figure 5;
    }
    else {
      remove (o_1, list) from Xseedlist;
      mark o_1 as "done";
      output (o_1, list[1].PredecessorDistance);
      update-Xseedlist(o_1);
    }
  }
```

Fig. 6. The extended OPTICS algorithm.

We organize all elements $o \in N_\varepsilon^{filter}(q)$ in ascending order according to their filter distance $d_f(o,q)$ yielding a sorted list. We walk through this sorted list, and compute for each visited object o the exact distance $d_o(o,q)$ until for $MinPts$ elements $d_o(o,q) \leq \varepsilon$ holds or until we reach the end. If we reached the end, we certainly know that q is no core point. Otherwise q is a core object initiating a new cluster C.

If our current object q is a core object, some of the objects $o \in N_\varepsilon^{filter}(q)$ are inserted into the Xseedlist (cf. Figure 5). All objects for which we have already computed $d_o(o,q)$, and for which $d_o(o,q) \leq \varepsilon$ holds, certainly belong to the same cluster as the core-object q. At the beginning of OL, we add the entry (o, NIL), where $PL(o) = NIL$ indicates that o certainly belongs to the same cluster as q. Objects o for which $d_o(o,q) > e$ holds are discarded. All objects $o \in N_\varepsilon^{filter}(q)$ for which we did not yet compute $d_o(o,q)$ are handled as follows:

- If there exists no entry belonging to o in OL, $(o, \langle(q, \text{Filter}, d_f(o,q))\rangle)$ is inserted into OL and the ordering conditions of Figure 5 are reestablished.
- If there already exists an entry for o in OL and, furthermore, $PL(o) = NIL$ holds, nothing is done.
- If there already exists an entry for o in OL and, furthermore, $PL(o) \neq NIL$ holds, $(q, \text{Filter}, d_f(o,q))$ is inserted into $PL(o)$ and the ordering conditions of Figure 5 are reestablished.

DBSCAN expands a cluster C as follows. We take the first element o_1 from OL and, if $PL(o_1) = NIL$ holds, we add o_1 to C, delete o_1 from OL, carry out

a range query around o_1, and try to expand the cluster C. If $PL(o_1) \neq NIL$ holds, we compute $d_o(o_1, o_{1,1})$. If $d_o(o_1, o_{1,1}) \leq \varepsilon$, we proceed as in the case where $PL(o_1) = NIL$ holds. If $d_o(o_1, o_{1,1}) > \varepsilon$ holds and length of $PL(o_1) > 1$, we delete $(o_{1,1}, F_{1,1}, PreDist(o_1, o_{1,1}))$ from $PL(o_1)$. If $d_o(o_1, o_{1,1}) > \varepsilon$ holds and length of $PL(o_1) = 1$, we delete o_1 from OL. Iteratively, we try to expand the current cluster by examining the first entry of $PL(o_1)$ until OL is empty.

Lemma 2. *The result of the extended DBSCAN algorithm is equivalent to the result of the original one.*

Proof. First, the determination whether an object o is a core object is correct as $o' \in N_\varepsilon(o) \Rightarrow o' \in N_\varepsilon^{filter}(o)$ holds due to the lower-bounding filter property. We test as many elements $o' \in N_\varepsilon^{filter}(o)$ as necessary to decide whether $|N_\varepsilon(o)| \geq MinPts$ holds. Second, similar to the proof of Lemma 1, we can guarantee that an object o is only added to the current cluster if $d_o(o, p) \leq \varepsilon$ holds for an object p which has already been singled out as a core object of the current cluster.

Length-Limitation of the Predecessor Lists In this section, we introduce two approaches for limiting the size of the predecessor lists to a constant l_{max} trying to keep the main memory footprint as small as possible. The first approach computes additional exact distances to reduce the length of the object reachability lists, while still computing the exact clustering. On the other hand, the second approach dispenses with additional exact distance computations leading to an approximated clustering.

Exact Clustering. In the case of OPTICS, for each object o_i in OL, we store all potential predecessor objects $o_{i,p}$ along with $PreDist(o_i, o_{i,p})$ in $PL(o_i)$. Due to the lower-bounding property of d_f, we can delete all entries in $PL(o_i)$ which are located at positions $l' > l$, if we have already computed the exact distance between o_i and the predecessor object $o_{i,l}$ located at position l. So each exact distance computation might possibly lead to several delete operations in the corresponding predecessor list. In order to limit the main memory footprint, we introduce a parameter l_{max} which restricts the allowed number of elements stored in a predecessor list. If more than l_{max} elements are contained in the list, we compute the exact distance for the predecessor $o_{i,1}$ located at the first position. Such an exact distance computation between o_i and $o_{i,1}$ usually causes $o_{i,1}$ to be moved upward in the list. All elements located behind its new position l are deleted. So if $l \leq l_{max}$ holds, the predecessor list is limited to at most l_{max} entries. Otherwise, we repeat the above procedure.

For DBSCAN, if the predecessor list of o_i is not NIL, we can limit its length by starting to compute $d_o(o_i, o_{i,1})$, i.e. the exact distance between o_i and the first element of $PL(o_i)$. If $d_o(o_i, o_{i,1}) \leq \varepsilon$ holds, we set $PL(o_i) = NIL$ indicating that o_i certainly belongs to the current cluster. Otherwise, we delete $(o_{i,1}, F_{i,1}, PreDist(o_i, o_{i,1}))$ and if the length of $PL(o_i)$ is still larger than l_{max}, we iteratively repeat this limitation procedure.

Lemma 3. *The above length limitation approach does not change the result of the extended DBSCAN and OPTICS algorithms.*

Proof. The presented DBSCAN algorithm guarantees that no entries $(o_{i,l}, F_{i,l}, PreDist(o_i, o_{i,l})$ are deleted which are necessary for determining whether an object is directly density-reachable (cf. Definition 1) from a core object of the current cluster. For OPTICS we do not delete any entries which are necessary for computing the minimum reachability distance w.r.t. all already processed objects.

Approximated Clustering. In our approximated approach, we artificially limit the length of the predecessor lists by discarding all elements which are located at a position higher than l_{max} without computing any additional exact distances. This approach might not produce the same result as the original OPTICS and DBSCAN algorithms as the filter distances do not necessarily have to coincide with the exact distances. Note that if we have a very exact filter, the cutting-off of the predecessor lists will not worsen the quality heavily (cf. Section 4.2). Nevertheless, we need to know how much quality we have to pay for the achieved efficiency gain.

3 Similarity Measures for Clusterings

The similarity measures introduced in this section are suitable for generally measuring the quality between partitioning and hierarchical approximated clusterings w.r.t. a given reference clustering. Both partitioning and hierarchical clustering algorithms rely on the notion of a cluster.

Definition 5 (cluster). *A* cluster C *is a non-empty subset of objects from a database* DB*, i.e.* $C \subseteq DB$ *and* $C \neq \emptyset$*.*

Definition 6 (partitioning clustering). *Let* DB *be a database of arbitrary objects. Furthermore, let* $C_1, \ldots, C_n$ *be pairwise disjoint clusters of* DB*, i.e.* $\forall i, j \in \{1, \ldots, n\} : i \neq j \Rightarrow C_i \cap C_j = \emptyset$*. Then we call* $CL_p = \{C_1, \ldots, C_n\}$ *a* partitioning clustering *of* DB*.*

Note that due to the handling of noise, we do not demand from a partitioning clustering $CL_p = \{C_1, \ldots, C_n\}$ that $C_1 \cup \ldots \cup C_n = DB$ holds. In contrast to the partitioning structure computed by DBSCAN, OPTICS computes a hierarchical clustering order which can be transformed into a tree structure by means of suitable cluster recognition algorithms [4, 5, 15].

Definition 7 (hierarchical clustering). *Let* DB *be a database of arbitrary objects. A* hierarchical clustering *is a tree* t_{root} *where each subtree* t *represents a cluster* C_t*, i.e.* $t = (C_t, (t_1, \ldots, t_n))$*, and the* n *subtrees* t_i *of* t *represent non-overlapping subsets* C_{t_i}*, i.e.* $\forall i, j \in \{1, \ldots, n\} : i \neq j \Rightarrow C_{t_i} \cap C_{t_j} = \emptyset \wedge C_{t_1} \cup \ldots \cup C_{t_n} \subseteq C_t$*. Furthermore, the root node* t_{root} *represents the complete database, i.e.* $C_{t_{root}} = DB$*.*

Again, we do not demand from the n subtrees t_i of $t = (C_t, (t_1, \ldots, t_n))$ that $C_{t_1} \cup \ldots \cup C_{t_n} = C_t$ holds (cf. Figure 3 where $A_1 \cup A_2 \neq A$).

3.1 Similarity Measure for Clusters

As outlined in the last section, both partitioning and hierarchical clusterings consist of flat clusters. In order to compare flat clusters to each other we need a suitable distance measure between sets of objects. One possible approach is to use distance measures as used for constructing distance-based hierarchical clusterings, e.g. the distance measures used by *single-link*, *average-link* or *complete-link* [1]. Although these distance measures are used for the construction of hierarchical clusterings, these measures are not suitable when it comes to evaluating the quality of flat clusters. The similarity of two clusters w.r.t. quality solely depends on the number of identical objects contained in both clusters which is reflected by the *symmetric set difference*.

Definition 8 (symmetric set difference). *Let C_1 and C_2 be two clusters of a database DB. Then the* symmetric set difference $d_\Delta : 2^{DB} \times 2^{DB} \rightarrow [0..1]$ *and the* normalized symmetric set difference $d_\Delta^{norm} : 2^{DB} \times 2^{DB} \rightarrow [0..1]$ *are defined as follows:*

$$d_\Delta(C_1, C_2) = |C_1 \cup C_2| - |C_1 \cap C_2|,$$
$$d_\Delta^{norm}(C_1, C_2) = \frac{|C_1 \cup C_2| - |C_1 \cap C_2|}{|C_1 \cup C_2|}.$$

Note that $(2^{DB}, d_\Delta)$ and $(2^{DB}, d_\Delta^{norm})$ are metric spaces.

3.2 Similarity Measure for Partitioning Clusterings

In this section, we will introduce a suitable distance measure between sets of clusters. Several approaches for comparing two sets S and T to each other exist in the literature. In [16] the authors survey the following distance functions: the *Hausdorff distance*, the *sum of minimal distances*, the *(fair-)surjection distance* and the *link distance*. All of these approaches rely on the possibility to match several elements in one set to just one element in the compared set which is questionable when comparing the quality of an approximated clustering to a reference clustering.

A distance measure on sets of clusters that demonstrates to be suitable for defining similarity between two partitioning clusterings is based on the *minimal weight perfect matching* of sets. This well known graph problem can be applied here by building a complete bipartite graph $G = (Cl, Cl', E)$ between two clusterings Cl and Cl'. The weight of each edge $(C_i, C'_j) \in Cl \times Cl'$ in this graph G is defined by the distance $d_\Delta(C_i, C'_j)$ introduced in the last section between the two clusters C_i and C'_j. A perfect matching is a subset $M \subseteq Cl \times Cl'$ that connects each cluster $C_i \in Cl$ to exactly one cluster $C'_j \in Cl'$ and vice versa. A minimal weight perfect matching is a matching having maximum cardinality and a minimum sum of weights of its edges. Since a perfect matching can only be found for sets of equal cardinality, it is necessary to introduce weights for unmatched clusters when defining a distance measure between clusterings.

Definition 9 (minimal matching distance). *Let DB be a database and let $dist : 2^{DB} \times 2^{DB} \to \mathbb{R}$ be a distance function between two clusters. Let $Cl = \{C_1, \ldots, C_{|Cl|}\}$ and $Cl' = \{C'_1, \ldots, C'_{|Cl'|}\}$ be two clusterings. We assume w.l.o.g. $|Cl| \leq |Cl'|$. Furthermore, let $w : 2^{DB} \to \mathbb{R}$ be a weight function for the unmatched clusters, and let π be a mapping that assigns $C' \in Cl'$ a unique number $i \in \{1, \ldots, |Cl'|\}$, denoted by $\pi(Cl') = (C'_1, \ldots, C'_{|Cl'|})$. The family of all possible permutations of Cl' is called $\Pi(Cl')$. Then the* minimal matching distance *$d_{mm}^{dist,w} : 2^{2^{DB}} \times 2^{2^{DB}} \to \mathbb{R}$ is defined as follows:*

$$d_{mm}^{dist,w}(Cl, Cl') = \min_{\pi \in \Pi(Cl')} \left(\sum_{i=1}^{|Cl|} dist(C_i, C'_{\pi(i)}) + \sum_{i=|Cl|+1}^{|Cl'|} w(C'_{\pi(i)}) \right).$$

The weight function $w : 2^{DB} \to \mathbb{R}$ provides the penalty given to every unassigned cluster of the clustering having larger cardinality. Let us note that this minimal matching distance is a specialization of the *netflow distance* [17] which is shown to be a metric if the distance function *dist* is a metric and the weight function meets the following conditions for two clusters $C, C' \in 2^{DB}$:

1. $w(C) > 0$
2. $w(C) + w(C') \geq dist(C, C')$

Note that the symmetric set difference d_Δ is a metric and can be used as the underlying distance function *dist* for the minimal matching distance. Furthermore, the unnormalized symmetric set difference allows us to define a meaningful weight function based on a dummy cluster $\emptyset$ since the empty set is not included as an element in a clustering (cf. Definition 6). We propose to use the following weight function $w_\emptyset(C) = d_\Delta(C, \emptyset)$ where each unmatched cluster C is penalized with a value equal to its cardinality $|C|$. Thus the metric character of the minimal matching distance is satisfied. Furthermore, large clusters which cannot be matched are penalized more than small clusters which is a desired property for an intuitive quality measure. Based on Definition 9, we can define our final quality criterion. We compare the costs for transforming an approximated clustering $Cl^{\approx}$ into a reference clustering Cl^{ref} to the costs piling up when transforming $Cl^{\approx}$ first into $\emptyset$, i.e. a clustering consisting of no clusters, and then transforming $\emptyset$ into Cl^{ref}.

Definition 10 (quality measure Q_{APC}). *Let $Cl^{\approx}$ be an approximated partitioning clustering and Cl^{ref} the corresponding reference clustering. Then, the* approximated partitioning clustering quality $Q_{\text{APC}}(Cl^{\approx}, Cl^{ref})$ *is equal to* 100% *if $Cl^{\approx} = Cl^{ref} = \emptyset$, else it is defined as*

$$\left(1 - \frac{d_{mm}^{d_\Delta, w_\emptyset}(Cl^{\approx}, Cl^{ref})}{d_{mm}^{d_\Delta, w_\emptyset}(Cl^{\approx}, \emptyset) + d_{mm}^{d_\Delta, w_\emptyset}(\emptyset, Cl^{ref})}\right) \cdot 100\%.$$

Note that our quality measure Q_{APC} is between 0% and 100%. If $Cl^{\approx}$ and Cl^{ref} are identical, $Q_{\text{APC}}(Cl^{\approx}, Cl^{ref}) = 100\%$ holds. On the other hand, if the

clusterings are not identical and the clusters from $Cl^{\approx}$ and Cl^{ref} have no objects in common, i.e. $\forall C^{\approx} \in Cl^{\approx}, C^{ref} \in Cl^{ref} : C^{\approx} \cap C^{ref} = \emptyset$, $Q_{\text{APC}}(Cl^{\approx}, Cl^{ref})$ is equal to 0%.

3.3 Similarity Measure for Hierarchical Clusterings

In this section, we present a quality measure for approximated hierarchical clusterings. To the best of our knowledge, the only quality measure for an approximated hierarchical clustering was introduced in [14]. A simple heuristic was applied to find the "best" cut-line, i.e. the most meaningful ε_{cut}-value (cf. Figure 3), for a reachability plot resulting from an approximated OPTICS run. The number of clusters found w.r.t. ε_{cut} was compared to the maximum number of clusters found in the reachability plot resulting from an exact clustering. This quality measure has two major drawbacks. First, it does not reflect the *hierarchical* clustering structure, but compares two flat clusterings to each other. Second, the actual elements building up a cluster are not accounted for. Only the number of clusters is used for computing the quality. In the following, we will present a quality measure for hierarchical clusterings which overcomes the two mentioned shortcomings.

As already outlined, a hierarchical clustering can be represented by a tree (cf. Definition 7). In order to define a meaningful quality measure for approximated hierarchical clusterings, we need a suitable distance measure for describing the similarity between two trees $t^{\approx}$ and t^{ref}. Note that each node of the trees reflects a flat cluster, and the complete trees represent the entire hierarchical clusterings.

A common and successfully applied approach to measure the similarity between two trees is the degree-2 edit distance [18]. It minimizes the number of edit operations necessary to transform one tree into the other using three basic operations, namely the insertion and deletion of a tree node and the change of a node label. Using these operations, we can define the degree-2 edit distance between two trees.

Definition 11 (cost of an edit sequence). *An edit operation e is the insertion, deletion or relabeling of a node in a tree t. Each edit operation e is assigned a non-negative cost $c(e)$. The cost $c(S)$ of a sequence of edit operations $S = \langle e_1, \ldots, e_m \rangle$ is defined as the sum of the cost of each edit operation, i.e. $c(S) = c(e_1) + \ldots + c(e_m)$.*

Definition 12 (degree-2 edit distance). *The degree-2 edit distance is based on degree-2 edit sequences which consist only of insertions or deletions of nodes n with $degree(n) \leq 2$, or of relabelings. Then, the* degree-2 edit distance ED_2 *between two trees t and t' is the minimum cost of all degree-2 edit sequences that transform t into t' or vice versa, i.e. $ED_2(t, t') = \min\{c(S) \mid S$ is a degree-2 edit sequence transforming t into $t'\}$.*

It is important to note that the degree-2 edit distance is well defined. Two trees can always be transformed into each other using only degree-2 edit operations. This is true because it is possible to construct any tree using only degree-2

edit operations. As the same is true for the deletion of an entire tree, it is always possible to delete t completely and then build t' from scratch resulting in a distance value for this pair of trees. In [18] the authors presented an algorithm which computes the degree-2 edit distance in $O(|t| \cdot |t'| \cdot D)$ time, where D denotes the maximum fanout of the trees, and $|t|$ and $|t'|$ the number of tree nodes.

We propose to set the cost $c(e)$ for each insert and delete operation e to 1. Furthermore, we propose to use the *normalized symmetric set difference* d_{Δ}^{norm} as introduced in Definition 8 to weight the relabeling cost. Using the normalized version allows us to define a well-balanced trade-off between the relabeling cost and the other edit operations, i.e. the insert and delete operations. Based on these costs, we can define our final quality criterion. We compare the costs for transforming an approximated hierarchical clustering $Cl^{\approx}$ modelled by a tree $t^{\approx}$ into a reference clustering Cl^{ref} modelled by a tree t^{ref}, to the costs piling up when transforming $t^{\approx}$ first into an "empty" tree t^{nil} and then transforming t^{nil} into t^{ref}.

Definition 13 (quality measure Q_{AHC}). *Let t^{ref} be a tree representing a hierarchical reference clustering Cl^{ref}, and t^{nil} a tree consisting of no nodes at all, representing an empty clustering. Furthermore, let $t^{\approx}$ be a tree representing an approximated clustering $Cl^{\approx}$. Then, the* approximated hierarchical clustering quality $Q_{\mathrm{AHC}}(Cl^{\approx}, Cl^{ref})$ *is equal to*

$$\left(1 - \frac{ED_2(t^{\approx}, t^{ref})}{ED_2(t^{\approx}, t^{nil}) + ED_2(t^{nil}, t^{ref})}\right) \cdot 100\%.$$

As the degree-2 edit distance is a metric [18], the approximated hierarchical clustering quality Q_{AHC} is between 0% and 100%.

4 Evaluation

In this section, we present a detailed experimental evaluation which demonstrates the characteristics and benefits of our new approach.

4.1 Settings

Test Data Sets. As test data, we used real-world CAD data represented by 81-dimensional feature vectors [19] and vector sets where each element consists of 7 6D vectors [20]. Furthermore, we used graphs [21] to represent real-world image data. If not otherwise stated, we used 1,000 complex objects from each data set, and we employed the filter and exact object distance functions proposed in [20, 19, 21]. The used distance functions can be characterized as follows:

– The exact distance computations on the graphs are very expensive. On the other hand, the used filter is rather selective and can efficiently be computed [21].

– The exact distance computations on the feature vectors and vector sets are also very expensive as normalization aspects for the CAD objects are taken into account. We compute 48 times the distance between two 81-dimensional feature vectors, and between two vector sets, in order to determine a normalized distance between two CAD objects [20, 19]. As a filter for the feature vectors we use their Euclidean norms [22] which is not very selective, but can be computed very efficiently. The filter used for the vector sets is more selective than the filter for the feature vectors, but also computationally more expensive [20].

Implementation. The original OPTICS and DBSCAN algorithms, along with their extensions introduced in this paper and the used filter and exact object distances were implemented in Java 1.4. The experiments were run on a workstation with a Xeon 2.4 GHz processor and 2 GB main memory under Linux.

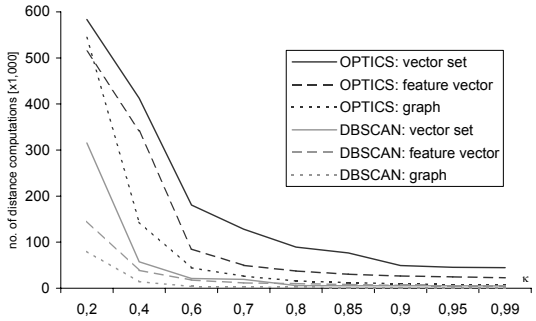
Parameter Setting. As suggested in [4], for an OPTICS run we used a maximum ε-parameter in order to create reachability plots containing the complete hierarchical clustering information. For DBSCAN, we chose an ε-parameter, based on the reachability plots (cf. the ε_{cut}-values in Figure 3), yielding as many flat clusters as possible. Furthermore, if not otherwise stated, the *MinPts*-parameter is set to 5, and the length of the predecessor lists is not limited.

Comparison Partners. As a comparison partner for extended OPTICS, we chose the full table scan based on the exact distances, because any other approach would include an unnecessary overhead and is not able to reduce the number of the required $|DB|^2$ exact distance computations. Furthermore, we compared our extended DBSCAN algorithm to the original DBSCAN algorithm based on a full table scan on the exact object distances, and we compared it to a version of DBSCAN which is based on ε-range queries efficiently carried out according to the multi-step query processing paradigm [10]. According to all our tests, this second comparison partner outperforms a DBSCAN algorithm using ε-range queries based on an M-tree [8] and the DBSCAN algorithm according to [12].

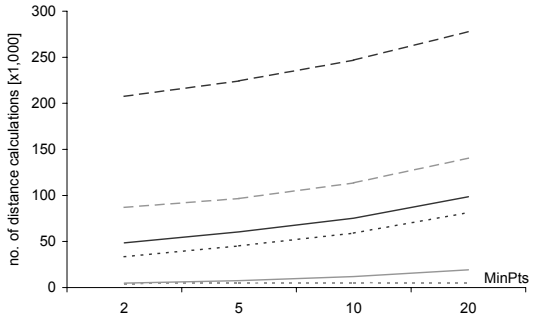
4.2 Experiments

Exact Clustering In this section, we first investigate the dependency of our approach on the filter quality, the *MinPts*-parameter, and the maximum allowed length of the predecessor lists. For these tests, we concentrate on the discussion of the overall number of distance computations. Furthermore, we investigate the influence of the ε-value in the case of DBSCAN, and, finally, we present the absolute run-times, in order to show that the required overhead of our approach is negligible compared to the saved exact distance computations.

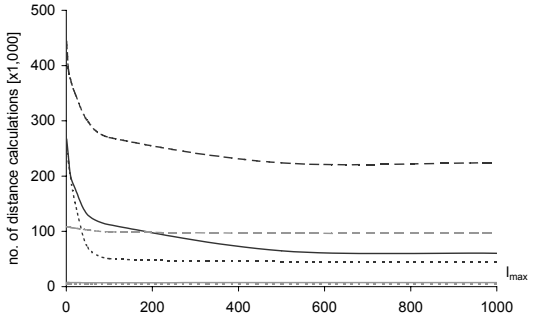
Dependency on the Filter Quality. In order to demonstrate the dependency of our approach on the quality of the filters, in a first experiment we utilized artificial filter distances d_f lower bounding the exact object distances d_o, i.e. $d_f(o_1, o_2) =$



(a) Dependency on the filter quality $d_f(o_1, o_2) = \kappa \cdot d_o(o_1, o_2)$.



(b) Dependency on the *MinPts*-parameter.



(c) Dependency on the maximum allowed length of the predecessor lists.

Fig. 7. Distance calculations for exact clusterings.

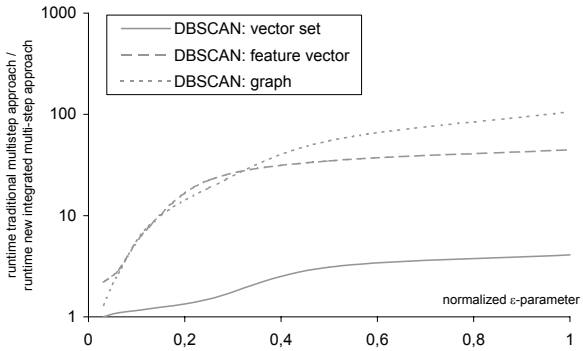


Fig. 8. Speed-up dependent on the ε-parameter.

$\kappa \cdot d_o(o_1, o_2)$ where κ is between 0 and 1. Figure 7a depicts the number of distance computations n_{dist} w.r.t. κ. In the case of DBSCAN, even rather bad filters, i.e. small values of κ, help to reduce the number of required distance computations considerably, indicating a possible high speed-up compared to both comparison partners of DBSCAN. For good filters, i.e. values of κ close to 1, n_{dist} is very small for DBSCAN and OPTICS indicating a possible high speed-up compared to a full table scan based on the exact distances d_o.

Dependency on the MinPts-Parameter. Figure 7b demonstrates the dependency of our approach for a varying *MinPts*-parameter while using the filters introduced in [22, 20, 19]. As our approach is based on *MinPts*-nearest neighbor queries, obviously the efficiency of our approach increases with a decreasing *MinPts*-parameter. Note that even for rather high *MinPts*-values around $10 = 1\% \cdot |DB|$, our approach saves up to one order of magnitude of exact distance computations compared to a full table scan based on d_o, if selective filters are used, e.g. the filters for the vector sets and the graphs. Furthermore, even for the filter of rather low selectivity used by the feature vectors, our approach needs only 1/9 of the maximum number of distance computations in the case of DBSCAN and about 1/4 in the case of OPTICS.

Dependency on the Maximum Allowed Length of the Predecessor Lists. Figure 7c depicts how the number of distance computations n_{dist} depends on the available main memory, i.e. the maximum allowed length l_{max} of the predecessor lists. Obviously, the higher the value for l_{max}, the less exact distance computations are required. The figure shows that for OPTICS we have an exponential decrease of n_{dist} w.r.t. l_{max}, and for DBSCAN n_{dist} is almost constant w.r.t. changing l_{max} parameters, indicating that small values of l_{max} are sufficient to reach the best possible runtimes.

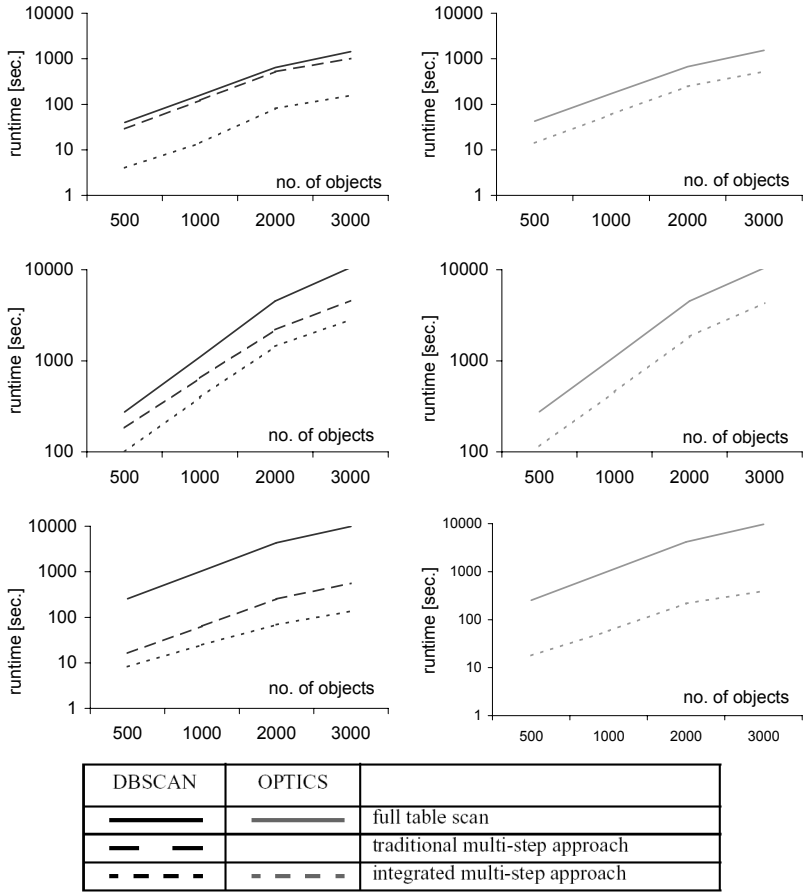


Fig. 9. Absolute runtimes w.r.t. varying database sizes.

Dependency on the ε-parameter. Figure 8 shows how the speed-up for DBSCAN between our integrated multi-step query processing approach and the traditional multi-step query processing approach depends on the chosen ε-parameter. The higher the chosen ε-parameter, the more our new approach outperforms the traditional one which has to compute the exact distances between o and q for all $o \in N_\varepsilon^{filter}(q)$. In contrast, our approach confines itself to *MinPts*-nearest neighbor queries on the exact distances and computes further distances only if compulsory to compute the exact clustering result.

Absolute Runtimes. Figure 9 presents the absolute run-times of the new extended DBSCAN and OPTICS algorithms which integrate the multi-step query processing paradigm compared to the full-table scan on the exact object representations. Furthermore, we also compare our extended DBSCAN to a DBSCAN

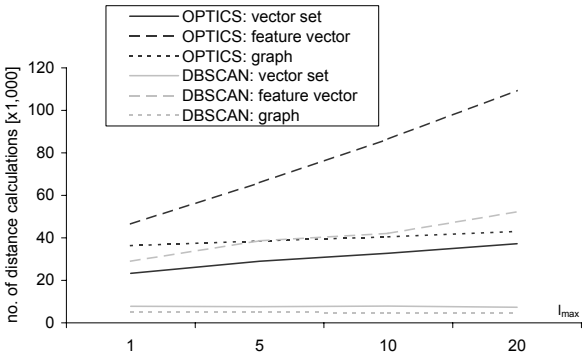


Fig. 10. Distance calculations for approximated clusterings.

variant using ε-range queries based on the traditional hmulti-step query processing paradigm. Note, that this comparison partner would induce an unnecessary overhead in the case of OPTICS where we have to use very high ε-parameters in order to detect the complete hierarchical clustering order. In all experiments, our approach was always the most efficient one. For instance, for DBSCAN on the feature vectors, our approach outperforms both comparison partners by an order of magnitude indicating that rather bad filters are already useful for our new extended DBSCAN algorithm. Note that the traditional multi-step query processing approach does not benefit much from non-selective filters even when small ε-values are used. In the case of OPTICS, the performance of our approach improves with increasing filter quality. For instance, for the graphs we achieve a speed-up factor of more than 30 indicating the suitability of our extended OPTICS algorithm.

Approximated Clustering In this section, we carry out experiments where we just cut off the predecessor lists $PL(o)$ after the l_{max}-th element without computing any additional exact distance computations between o and the discarded potential predecessor objects. Note that this approach might lead to an information loss. Figure 10 shows that the maximum number of needed distance calculations only marginally increases for higher l_{max}-values for the graphs and the vector sets indicating that we can cut off the object reachability lists at small l_{max}-values without a considerable information loss. On the other hand, for the feature vectors we have to compute more exact distance computations the higher the l_{max}-value is. The additionally needed exact distance computations (cf. line (*) in Figure 6) are due to the rather low filter selectivity of the used filter.

Next, we examine the quality of our approximated clustering algorithms by using the quality measures introduced in Section 3. For extracting the hierarchical tree structure, we used the cluster recognition algorithm presented in [5]. Figure 11 depicts the quality measures Q_{APC} for DBSCAN and Q_{AHC} for OP-

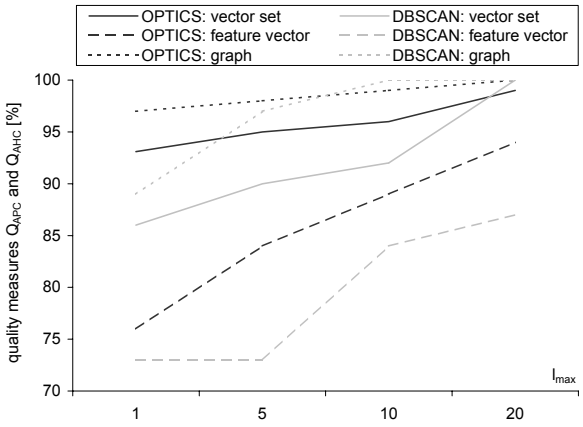


Fig. 11. Quality measures for approximated clusterings.

TICS for our three test data sets w.r.t. varying l_{max} values. Our quality measures indicate a very high quality for the graphs and the vector sets over the full range of the investigated l_{max} values. On the other hand, when using the feature vectors both quality measures Q_{APC} (for DBSCAN) and Q_{AHC} (for OPTICS) increase with increasing l_{max} values. These tests not only indicate that we can cut off the predecessor lists at small values of l_{max} without considerably worsening the clustering quality when using selective filters. The tests also demonstrate the suitability of our quality measures Q_{APC} and Q_{AHC} which indicate low quality when filters of low selectivity are combined with small l_{max} values.

5 Conclusion

In many different application areas, density-based clustering is an effective approach for mining complex data. Unfortunately, the runtime of these data-mining algorithms is rather high, as the distance functions between complex object representations are often very expensive. In this paper, we showed how to integrate the well-known multi-step query processing paradigm directly into the two density-based clustering algorithms DBSCAN and OPTICS. We replaced the expensive exact ε-range queries by *MinPts*-nearest neighbor queries which themselves are based on ε-range queries on lower-bounding filter distances. Further exact complex distance computations are only carried out at that stage of the algorithms where they are compulsory to compute the correct clustering result. Furthermore, we showed how we can use the presented approach for approximated clustering. In order to evaluate the trade-off between the achieved efficiency gain and the quality loss, we introduced suitable quality measures for comparing the partitioning and hierarchical approximated clusterings to the exact ones. In a broad experimental evaluation based on real-world test data sets

we demonstrated that our new approach leads to a significant speed-up compared to a full-table scan on the exact object representations as well as compared to an approach, where the ε-range queries are accelerated by means of the traditional multi-step query processing concept. Furthermore, we showed that for approximated clusterings we can reduce the number of required distance computations even further. Finally, we pointed out that the resulting approximated clustering quality heavily depends on the filter quality demonstrating the suitability of our introduced quality measures.

In our future work, we will demonstrate that other data mining algorithms dealing with complex object representations also benefit from a direct integration of the multi-step query processing paradigm.

References

1. Jain, A.K., Murty, M.N., Flynn, P.J.: "Data Clustering: A Review". ACM Computing Surveys **31(3)** (1999) 265–323
2. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96), Portland, OR, AAAI Press (1996) 291–316
3. Kailing, K., Kriegel, H.P., Pryakhin, A., Schubert, M.: "Clustering Multi-Represented Objects with Noise". In: Proc. 8th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'04), Sydney, Australia. (2004) 394–403
4. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: "OPTICS: Ordering Points to Identify the Clustering Structure". In: Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), Philadelphia, PA. (1999) 49–60
5. Brecheisen, S., Kriegel, H.P., Kröger, P., Pfeifle, M.: "Visually Mining Through Cluster Hierarchies". In: Proc. SIAM Int. Conf. on Data Mining (SDM'04), Lake Buena Vista, FL. (2004) 400–412
6. Guttman, A.: "R-trees: A Dynamic Index Structure for Spatial Searching". In: Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'84). (1984) 47–57
7. Gaede V., G.O.: "Multidimensional Access Methods". ACM Computing Surveys **30(2)** (1998) 170–231
8. Ciaccia, P., Patella, M., Zezula, P.: "M-Tree: An Efficient Access Method for Similarity Search in Metric Spaces". In: Proc. 23rd Int. Conf. of Very Large Data Bases, Athens, Greece. (1997) 426–435
9. Chávez, E., Navarro, G., Beaza-Yates, R., Marroquín, J.: "Searching in Metric Spaces". ACM Computing Surveyes **33(3)** (2001) 273–321
10. Agrawal, R., Faloutsos, C., Swami, A.: "Efficient Similarity Search in Sequence Databases". In: Proc. 4th. Int. Conf. on Foundations of Data Organization and Algorithms (FODO'93), Evanston, ILL. Volume 730 of Lecture Notes in Computer Science (LNCS)., Springer (1993) 69–84
11. Böhm, C., Braunmüller, B., Breunig, M., Kriegel, H.P.: "High Performance Clustering Based on the Similarity Join". In: Proc. 9th Int. Conf. on Information and Knowledge Management (CIKM 2000), Washington, DC. (2000) 298–313
12. Braunmüller, B., Ester, M., Kriegel, H.P., Sander, J.: "Efficiently Supporting Multiple Similarity Queries for Mining in Metric Databases". In: Proc. Int. Conf. on Data Engineering (ICDE 2000), San Diego, CA. (2000) 256–267

13. Wang, J.T.L., Wang, X., Lin, K.I., Shasha, D., Shapiro, B.A., Zhang, K.: "Evaluating a Class of Distance-Mapping Algorithms for Data Mining and Clustering". In: Proc. 5th Int. Conf. on Knowledge Discovery and Data Mining (KDD'99), San Diego, CA. (1999) 307–311
14. Zhou, J., Sander, S.: "Data Bubbles for Non-Vector Data: Speeding-up Hierarchical Clustering in Arbitrary Metric Spaces". In: Proc. 29th Int. Conf. on Very Large Databases (VLDB'03), Berlin, Germany. (2003) 452–463
15. Sander, J., Qin, X., Lu, Z., Niu, N., Kovarsky, A.: "Automatic Extraction of Clusters from Hierarchical Clustering Representations". In: Proc. 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2003), Seoul, Korea. (2003) 75–87
16. Eiter, T., Mannila, H.: "Distance Measures for Point Sets and Their Computation". Acta Informatica **34** (1997) 103–133
17. Ramon, J., Bruynooghe, M.: "A polynomial time computable metric between point sets". Acta Informatica **37** (2001) 765–780
18. Zhang, K., Wang, J., Shasha, D.: "On the editing distance between undirected acyclic graphs". International Journal of Foundations of Computer Science **7(1)** (1996) 43–57
19. Kriegel, H.P., Kröger, P., Mashael, Z., Pfeifle, M., Pötke, M., Seidl, T.: "Effective Similarity Search on Voxelized CAD Objects". In: Proc. 8th Int. Conf. on Database Systems for Advanced Applications (DASFAA'03), Kyoto, Japan. (2003) 27–36
20. Kriegel, H.P., Brecheisen, S., Kröger, P., Pfeifle, M., Schubert, M.: "Using Sets of Feature Vectors for Similarity Search on Voxelized CAD Objects". In: Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'03), San Diego, CA. (2003) 587–598
21. Kailing, K., Kriegel, H.P., Schönauer, S., Seidl, T.: "Efficient Similarity Search for Hierarchical Data in Large Databases". In: Proc. 9th Int. Conf. on Extending Database Technology (EDBT'04), Heraklion, Greece. (2004) 676–693
22. Fonseca, M.J., Jorge, J.A.: "Indexing High-Dimensional Data for Content-Based Retrieval in Large Databases". In: Proc. 8th Int. Conf. on Database Systems for Advanced Applications (DASFAA'03), Kyoto, Japan. (2003) 267–274