In Proc. 4th SIAM Int. Conf. on Data Mining, pp. 400-412, Lake Buena Vista, FL, 2004

Visually Mining Through Cluster Hierarchies

Stefan Brecheisen Hans-Peter Kriegel Peer Kröger Martin Pfeifle

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

Abstract

Similarity search in database systems is becoming an increasingly important task in modern application domains such as multimedia, molecular biology, medical imaging, computer aided engineering, marketing and purchasing assistance as well as many others. In this paper, we show how visualizing the hierarchical clustering structure of a database of objects can aid the user in his time consuming task to find similar objects. We present related work and explain its shortcomings which led to the development of our new methods. Based on reachability plots, we introduce approaches which automatically extract the significant clusters in a hierarchical cluster representation along with suitable cluster representatives. These techniques can be used as a basis for visual data mining. We implemented our algorithms resulting in an industrial prototype which we used for the experimental evaluation. This evaluation is based on real world test data sets and points out that our new approaches to automatic cluster recognition and extraction of cluster representatives create meaningful and useful results in comparatively short time.

Keywords

similarity search, visual data mining, clustering

1 Introduction

In the last ten years, an increasing number of database applications has emerged for which efficient and effective support for similarity search is substantial. The importance of similarity search grows in application areas such as multi-media, medical imaging, molecular biology, computer aided engineering, marketing and purchasing assistance, etc. [12, 1, 10, 11, 2, 5, 6, 13]

Hierarchical clustering was shown to be effective for evaluating similarity models [16, 14]. Especially, the reachability plot generated by *OPTICS* [4] is suitable for assessing the quality of a similarity model. Furthermore, visually analyzing cluster hierarchies helps the user, e.g. an engineer, to find and group similar objects. Solid cluster extraction and meaningful cluster representatives form the foundation for providing the user with significant and quick information.

In this paper, we introduce algorithms for automatically detecting hierarchical clusters along with their corresponding representatives. In order to evaluate our ideas, we developed a prototype called *BOSS* (*B*rowsing *O*PTICS Plots for *S*imilarity *S*earch). BOSS is based on techniques related to *visual data mining*. It helps to visually analyze cluster hierarchies by providing meaningful cluster representatives.

To sum up, the main contributions of this paper are as follows:

- We explain how different important application ranges would benefit from a tool which allows visually mining through cluster hierarchies.
- We reason why the hierarchical clustering algorithm *OPTICS* forms a suitable foundation for such a browsing tool.
- We introduce a new cluster recognition algorithm for the reachability plots generated by *OPTICS*. This algorithm generalizes all the other known cluster recognition algorithms for reachability plots. Although our new algorithm does not need a sophisticated and extensive parameter setting, it outperforms the other cluster recognition algorithms wrt. the quality and number of recognized clusters and subclusters.
- We tackle the complex problem of finding suitable cluster representatives. We introduce two new approaches and show that they yield more intuitive representatives than the known medoid-approach.
- We describe a new browsing tool which comprises algorithms for cluster recognition and representation. This tool, called BOSS, is based on a client-server architecture which allows the user to get a quick and meaningful overview over large data sets.

The remainder of the paper is organized as follows: After briefly introducing reachability plots, we present in Section 2 the application areas of hierachical clustering along with the corresponding requirements in the industrial and in the scientific community which motivated the development of BOSS. In Sections 3 and 4, we introduce suitable algorithms for cluster recognition and cluster representatives respectively. In Section 5, we describe the actual industrial

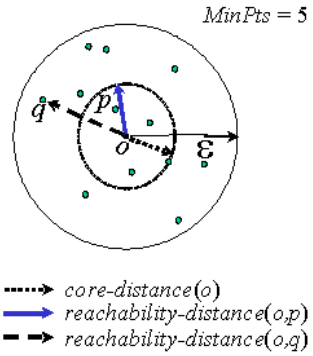

Figure 1: Illustration of core-level and reachability-distance.

prototype we developed and used it, in Section 6, to evaluate our new cluster recognition and representation algorithms. The paper concludes in Section 7 with a short summary and a few remarks on future work.

2 Hierachical Clustering

In this section, we outline the application ranges which led to the development of our interactive browsing tool, called BOSS. In order to understand the connection between BOSS and the application requirements we first introduce the major concepts of the hierarchical clustering algorithm OPTICS and its output, the so called reachability plots, which served as a starting point for BOSS. The technical aspects related to BOSS are described later in Section 6.

2.1 Major Concepts of OPTICS

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. Using the density-based hierarchical clustering algorithm OPTICS yields several advantages due to the following reasons.

- OPTICS is – in contrast to most other algorithms – relatively insensitive to its two input parameters, ε and *MinPts*. The authors in [4] state that the input parameters just have to be large enough to produce good results.
- OPTICS is a hierarchical clustering method which yields more information about the cluster structure than a method that computes a flat partitioning of the data (e.g. k-means[17]).
- There exists a very efficient variant of the OPTICS algorithm which is based on a sophisticated data compression technique called "Data Bubbles" [8], where we have to trade only very little quality of the clustering result for a great increase in performance.
- There exists an efficient incremental version of the OPTICS algorithm [15].

OPTICS emerges from the algorithm DBSCAN [9] which computes a flat partitioning of the data. The clustering notion underlying DBSCAN is that of density-connected sets (cf. [9] for more details). It is assumed that there is a metric distance function on the objects in the database (e.g. one of the L_p-norms for a database of feature vectors).

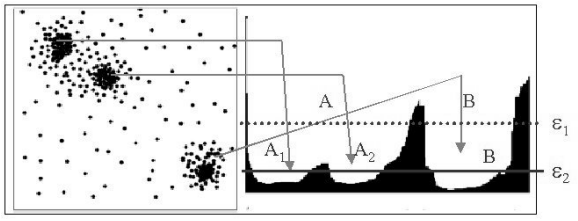

Figure 2: Reachability plot (right) computed by OPTICS for a sample 2-D dataset (left).

In contrast to DBSCAN, OPTICS does not assign cluster memberships but computes an *ordering* in which the objects are processed and additionally generates the information which would be used by an extended DBSCAN algorithm to assign cluster memberships. This information consists of only two values for each object, the *core-distance* and the *reachability-distance*.

Definition 1. (core-distance) *Let $o \in DB$, $MinPts \in \mathbb{N}$, $\varepsilon \in \mathbb{R}$, and MinPts-dist(o) be the distance from o to its MinPts-nearest neighbor. The* core-distance *of o wrt. ε and MinPts is defined as follows:*

$$\mathit{Core\text{-}Dist}(o) := \begin{cases} \infty & \text{if } |\mathcal{N}_\varepsilon(o)| < \mathit{MinPts} \\ \mathit{MinPts\text{-}dist}(o) & \text{otherwise.} \end{cases}$$

Definition 2. (reachability-distance) *Let $o \in DB$, $MinPts \in \mathbb{N}$ and $\varepsilon \in \mathbb{R}$. The* reachability distance *of o wrt. ε and MinPts from an object $p \in DB$ is defined as follows:*

$$\mathit{Reach\text{-}Dist}(p, o) := \max(\mathit{Core\text{-}Dist}(p), \mathit{distance}(p, o)).$$

Figure 1 illustrates both concepts: The reachability-distance of p from o equals to the core-distance of o and the reachability-distance of q from o equals to the distance between q and o.

The original output of OPTICS is an ordering of the objects, a so called *cluster ordering*:

Definition 3. (cluster ordering) *Let $MinPts \in \mathbb{N}$, $\varepsilon \in \mathbb{R}$, and CO be a totally ordered permutation of the database objects. Each $o \in D$ has additional attributes $o.P$, $o.C$ and $o.R$, where $o.P \in \{1, \ldots, |CO|\}$ symbolizes the position of o in CO. We call CO a* cluster ordering *wrt. ε and MinPts if the following three conditions hold:*

(1) $\forall p \in CO : p.C = \mathit{Core\text{-}Dist}(p)$

(2) $\forall o, x, y \in CO : o.P < x.P \wedge x.P < y.P \Rightarrow \mathit{Reach\text{-}Dist}(o, x) \leq \mathit{Reach\text{-}Dist}(o, y)$

(3) $\forall p, o \in CO : R(p) = \min\{\mathit{Reach\text{-}Dist}(o, p) \mid o.P < p.P\}$, *where* $\min \emptyset = \infty$.

Intuitively, Condition (2) states that the order is built on selecting at each position i in CO that object o having the minimum reachability to any object before i. $o.C$ symbolizes the core-distance of an object o in CO whereas $o.R$ is the reachability-distance assigned to object o during the generation of CO. We call $o.R$ the *reachablity* of object o throughout the paper. Note that $o.R$ is only well-defined in the context of a cluster ordering.

The cluster structure can be visualized through 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 2. Valleys in this plot 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.

The reachability plot generated by OPTICS can be cut at any level ε_{cut} parallel to the abscissa. It represents the density-based clusters according to the density threshold ε_{cut}: A consecutive subsequence of objects having a smaller reachability value than ε_{cut} belongs to the same cluster. An example is presented in Figure 2: For a cut at the level ε_1 we find two clusters denoted as A and B. Compared to this clustering, a cut at level ε_2 would yield three clusters. The cluster A is split into two smaller clusters denoted by A_1 and A_2 and cluster B decreased its size. Usually, for evaluation purposes, a good value for ε_{cut} would yield as many clusters as possible.

2.2 Application Ranges

BOSS was designed for three different purposes: visual data mining, similarity search and evaluation of similarity models. For the first two applications, the choice of the representative objects of a cluster is the key step. It helps the user to get a meaningful and quick overview over a large existing data set. Furthermore, BOSS helps scientists to evaluate new similarity models.

2.2.1 Visual Data Mining

As defined in [3], visual data mining is a step in the KDD process that utilizes visualization as a communication channel between the computer and the user to produce novel and interpretable patterns. Based on the balance and sequence of the automatic and the interactive (visual) part of the KDD process, three classes of visual data mining can be identified.

- Visualization of the data mining result:
  An algorithm extracts patterns from the data. These patterns are visualized to make them interpretable. Based on the visualization, the user may want to return to the data mining algorithm and run it again with different input parameters (cf. Figure 3a).
- Visualization of an intermediate result:
  An algorithm performs an analysis of the data not producing the final patterns but an intermediate result which can be visualized. Then the user retrieves the interesting patterns in the visualization of the intermediate result (cf. Figure 3b).
- Visualization of the data:
  Data is visualized immediately without running a sophisticated algorithm before. Patterns are obtained by the user by exploring the visualized data (cf. Figure 3c).

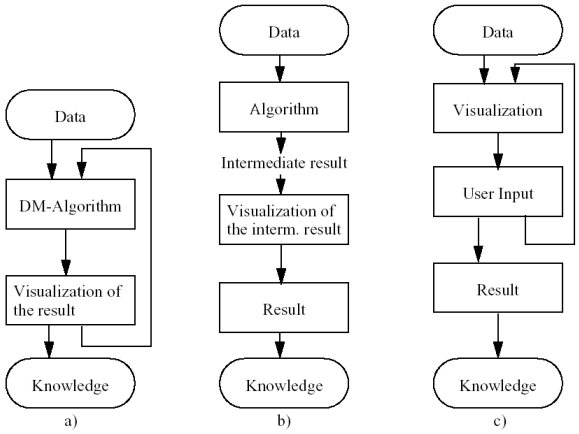

Figure 3: Different approaches to visual data mining [3].

The approach presented in this paper belongs to the second class. A hierarchical clustering algorithm is applied to the data, which extracts the clustering structure as an intermediate result. There is no meaning associated with the generated clusters. However, our approach allows the user to visually analyze the contents of the clusters. The clustering algorithm used in the algorithmic part is independent from an application. It performs the core part of the data mining process and its result serves as a multi-purpose basis for further analysis directed by the user. This way the user may obtain novel information which was not even known to exist in the data set. This is in contrast to similarity search where the user is restricted to find similar parts respective to a query object and a predetermined similarity measure.

2.2.2 Similarity Search

The development, design, manufacturing and maintenance of modern engineering products is a very expensive and complex task. Effective similarity models are required for two- and three-dimensional CAD applications to cope with rapidly growing amounts of data. Shorter product cycles and a greater diversity of models are becoming decisive competitive factors in the hard-fought automobile and aircraft market. These demands can only be met if the engineers have an overview of already existing CAD parts. It would be desirable to have an interactive data browsing tool which depicts the reachability plot computed by OPTICS in a user friendly way together with appropriate representatives of the clusters. This clear illustration would support

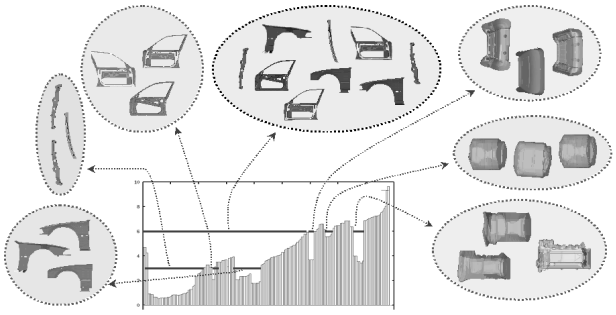

Figure 4: Browsing through reachability plots with different density thresholds ε_{cut}.

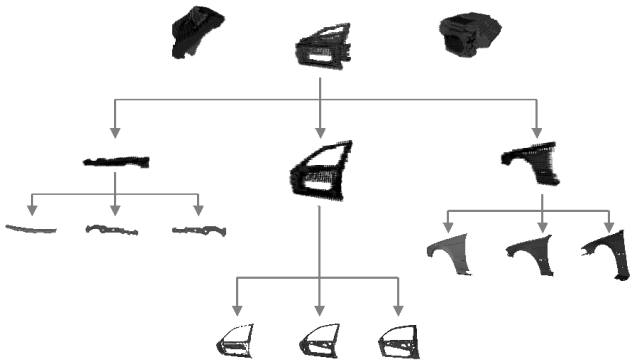

Figure 5: Hierarchically ordered representatives.

the user in his time-consuming task to find similar parts. From the industrial user's point of view, this browsing tool should meet the following two requirements:

- The hierarchical clustering structure of the dataset is revealed at a glance. The reachability plot is an intuitive visualization of the clustering hierarchy which helps to assign each object to its corresponding cluster or to noise. Furthermore, the hierarchical representation of the clusters using the reachability plot helps the user to get a quick overview over all clusters and their relation to each other. As each entry in the reachabiltity plot is assigned to one object, we can easily illustrate some representatives of the clusters belonging to the current density threshold ε_{cut} (cf. Figure 4).
- The user is not only interested in the shape and the number of the clusters, but also in the specific objects building up a cluster. As for large clusters it is rather difficult to depict all objects, representatives of each cluster should be displayed. To follow up a first idea, these representatives could be simply constructed by superimposing all parts belonging to the regarded cluster (cf. Figure 5). We can browse through the hierarchy of the representatives in the same way as through the OPTICS plots.

This way, the cost of developing and producing new parts could be reduced by maximizing the reuse of existing parts, because the user can browse through the hierarchical structure of the clusters in a top-down way. Thus the engineers get an overview of already existing parts and are able to navigate their way through the diversity of existing variants of products, such as cars.

2.2.3 Evaluation of Similarity Models

In general, similarity models can be evaluated by computing k-nearest neighbour queries (k-nn queries). As shown in [16], this evaluation approach is subjective and error-prone because the quality measure of the similarity model depends on the results of a few similarity queries and, therefore, on the choice of the query objects. A model may perfectly reflect the intuitive similarity according to the chosen query objects and would be evaluated as "good" although it produces disastrous results for other query objects.

A better way to evaluate and compare several similarity models is to apply a clustering algorithm. Clustering groups a set of objects into classes where objects within one class are similar and objects of different classes are dissimilar to each other. The result can be used to evaluate which model is best suited for which kind of objects. It is more objective since each object of the data set is taken into account to evaluate the data models.

3 Cluster Recognition

In this section, we address the first task of automatically extracting clusters from the reachability plots. After a brief discussion of recent work in that area, we propose a new approach for hierarchical cluster recognition based on reachability plots called *Gradient Clustering*.

3.1 Recent Work

To the best of our knowledge, there are only two methods for automatic cluster extraction from hierarchical representations such as reachability plots or dendrograms – both are also based on reachability plots. Since clusters are represented as valleys (or dents) in the reachability plot, the task of automatic cluster extraction is to identify significant valleys.

The first approach proposed in [4] called ξ-clustering is based on the steepness of the valleys in the reachability plot. The steepness is defined by means of an input parameter ξ. The method suffers from the fact that this input parameter is difficult to understand and hard to determine. Rather small variations of the value ξ often lead to drastic changes of the resulting clustering hierarchy. As a consequence, this method is unsuitable for our purpose of automatic cluster extraction.

The second approach was proposed recently by Sander et al. [18]. The authors describe an algorithm called `cluster_tree` that automatically extracts a hierarchical clustering from a reachability plot and computes a cluster tree. It is based on the idea that *significant* local maxima in

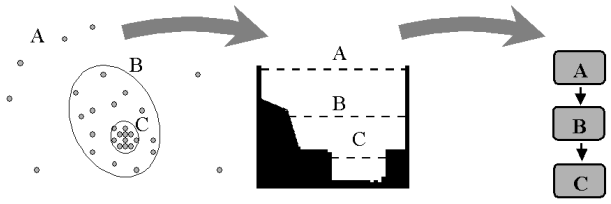

Figure 6: Sample narrowing clusters: data space (left); reachability plot (midle); cluster hierarchy (right)

the reachability plot separate clusters. Two parameters are introduced to decide whether a local maximum is significant: The first parameter specifies the minimum cluster size, i.e. how many objects must be located between two significant local maxima. The second parameter specifies the ratio between the reachability of a significant local maximum m and the average reachabilities of the regions to the left and to the right of m. The authors in [18] propose to set the minimum cluster size to 0.5% of the data set size and the second parameter to 0.75. They empirically show, that this default setting approximately represents the requirements of a typical user.

Although the second method is rather suitable for automatic cluster extraction from reachability plots, it has one major drawback. Many real-world data sets consist of narrowing clusters, i.e. clusters each consisting of exactly one smaller sub-cluster (cf. Figure 6).

Since the algorithm `cluster_tree` runs through a list of all local maxima (sorted in descending order of reachability) and decides at each local maximum m, whether m is significant to split the objects to the left of m and to the right of m into two clusters, the algorithm cannot detect such narrowing clusters. These clusters cannot be split by a significant maximum. Figure 6 illustrates this fact. The narrowing cluster A consists of one cluster B which is itself narrowing consisting of one cluster C (the clusters are indicated by dashed lines). The algorithm `cluster_tree` will only find cluster A since there are no local maxima to split clusters B and C. The ξ-clustering will detect only one of the clusters A, B or C depending on the ξ-parameter but also fails to detect the cluster hierarchy.

A new cluster recognition algorithm should meet the following requirements:

- It should detect all kinds of subclusters, including narrowing subclusters.
- It should create a clustering structure which is close to the one which an experienced user would manually extract from a given reachability plot.
- It should allow an easy integration into the OPTICS algorithm. We do not want to apply an additional cluster recognition step after the OPTICS run is completed. In contrast, the hierarchical clustering structure should be created on-the-fly during the OPTICS run without causing any noteworthy additional cost.
- It should be integrable into the incremental version of OPTICS [15], as most of the discussed application ranges benefit from such a incremental version.

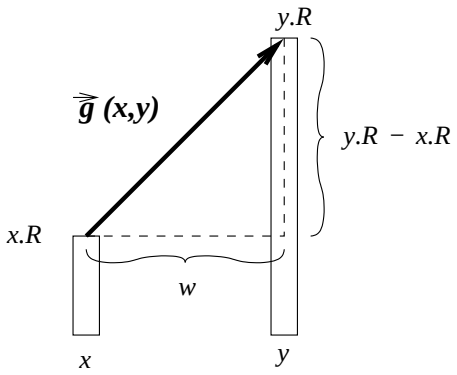

Figure 7: Gradient vector $\vec{g}(x, y)$ of two objects x and y adjacent in the cluster ordering.

3.2 Gradient Clustering

In this section, we introduce our new *gradient-clustering* algorithm which fulfills all of the above mentioned requirements. The idea behind our new cluster extraction algorithm is based on the concept of *inflexion points*. During the OPTICS run, we decide for each point added to the result set, i.e. the reachability plot, whether it is an inflexion point or not. If it is an inflexion point we might be at the start or at the end of a new subcluster. We store the possible starting points of the sublcusters in a list, called $startPts$. This stack consists of pairs $(o.P, o.R)$. Our *gradient-clustering* algorithm can easily be intergrated into OPTICS and is described in full detail, after we have formally introduced the new concept of *inflexion points*.

In the following, we assume that CO is a cluster ordering as defined in Definition 3. We call two objects $o_1, o_2 \in CO$ *adjacent in* CO if $o_2.P = o_1.P + 1$. Let us recall, that $o.R$ is the reachability of $o \in CO$ assigned by OPTICS while generating CO. For any two objects $o_1, o_2 \in CO$ adjacent in the cluster ordering, we can determine the gradient of the reachability values $o_1.R$ and $o_2.R$. The gradient can easily be modelled as a 2D vector where the y-axis measures the reachability values ($o_1.R$ and $o_2.R$) in the ordering, and the x-axis represent the ordering of the objects. If we assume that each object in the ordering is seperated by width w, the gradient of o_1 and o_2 is the vector

$$\vec{g}(o_1, o_2) = \begin{pmatrix} w \\ o_2.R - o_1.R \end{pmatrix}.$$

An example for a gradient vector of two objects x and y adjacent in a cluster ordering is depicted in Figure 7.

Intuitively, an inflexion point should be an object in the cluster ordering where the gradient of the reachabilities changes significantly. This significant change indicates a starting or an end point of a cluster.

Let $x, y, z \in CO$ be adjacent, i.e. $x.P + 1 = y.P = z.P - 1$. We can now measure the differences between the gradient vector $\vec{g}(x, y)$ and $\vec{g}(y, z)$ be computing the cosinus function of the angle between these two vectors. The cosinus of this angle is equal to -1 if the angle is $180°$, i.e. the vectors have the same direction. On the other hand, if the gradient vectors differ a lot, the angle between them will be clearly smaller than $180°$ and thus the cosinus will be significantly greater than -1. This observation motivates the concepts of inflection index and inflexion points:

DEFINITION 4. (INFLEXION INDEX) *Let CO be a cluster ordering and $x, y, z \in CO$ be objects adjacent in CO. The* inflexion index *of y, denoted by $II(y)$, is defined as the cosinus of the angle between the gradient vector of x, y ($\vec{g}(x, y)$) and the gradient vector of y, z ($\vec{g}(y, z)$), formally:*

$$II(y) = \cos \varphi_{(\vec{g}(x,y), \vec{g}(y,z))} = \frac{-w^2 + (x.R - y.R)(z.R - y.R)}{\|\vec{g}(x, y)\| \, \|\vec{g}(y, z)\|},$$

where $\|\vec{v}\| := \sqrt{v_1^2 + v_2^2}$ is the length of the vector $\vec{v}$.

DEFINITION 5. (INFLEXION POINT) *Let CO be a cluster ordering and $x, y, z \in CO$ be objects adjacent in CO and let $t \in \mathbb{R}$. Object y is an* inflexion point *iff*

$$II(y) > t.$$

The concept of inflexion points is suitable to detect objects in CO which are interesting for extracting clusters.

DEFINITION 6. (GRADIENT DETERMINANT) *Let CO be a cluster ordering and $x, y, z \in CO$ be objects adjacent in CO. The* gradient determinant *of the gradients $\vec{g}(x, y)$ and $\vec{g}(y, z)$ is defined as*

$$gd(\vec{g}(x, y), \vec{g}(y, z)) := \begin{vmatrix} w & w \\ y.R - x.R & z.R - y.R \end{vmatrix}$$

If x, y, z are clear from the context, we use the short form $gd(y)$ for the gradient determinant $gd(\vec{g}(x, y), \vec{g}(y, z))$.

The sign of $gd(y)$ indicates whether $y \in CO$ is a starting point or end point of a cluster. In fact, we can distinguish the following two cases which are visualized in Figure 8:

- $II(y) > t$ and $gd(y) > 0$:
  Object y is either a starting point of a cluster (e.g. object a in Figure 8) or the first object outside of a cluster (e.g. object z in Figure 8).
- $II(y) > t$ and $gd(y) < 0$:
  Object y is either an end point of a cluster (e.g. object n in Figure 8) or the second object inside a cluster (e.g. object b in Figure 8).

Let us note that a local maximum $m \in CO$ which is the cluster seperation point in [18] is a special form of the first case (i.e. $II(m) > t$ and $gd(m) > 0$).

The threshold t is independent from the absolut reachability values of the objects in CO. The influence of t is also very comprehensible because if we know which values for the angles between gradients are interesting, we can easily compute t. For example, if we are interested in angles $< 120°$ and $> 240°$ we set $t = \cos 120° = -0.5$.

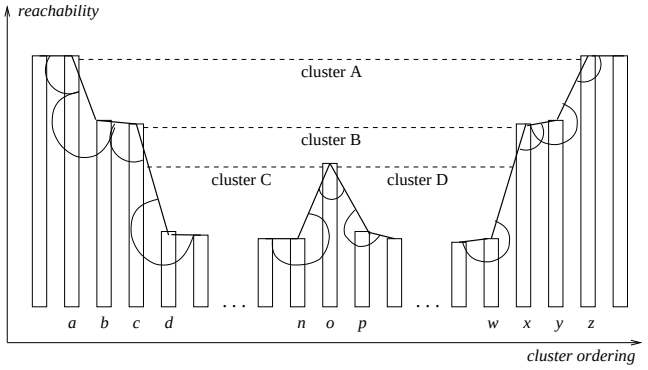

Figure 8: Illustration of inflexion points measuring the angle between the gradient vectors of objects adjacent in the ordering.

Obviously, the gradient clustering algorithm is able to extract narrowing clusters. First experimental comparisons with the methods in [18] and [4] are presented in Section 6.

The pseudo code of the Gradient Clustering algorithm is depicted in Figure 9, which works like this. Initially, the first object of the cluster ordering CO is pushed to the stack of starting points $startPts$. Whenever a new starting point is found, it is pushed to the stack. If the current object is an end point, a new cluster is created containing all objects between the starting point on top of the stack and the current end point. Starting points are removed from the stack if their reachablity is lower than the reachability of the current object. Clusters are created as described above for all removed starting points as well as for the starting point which remains in the stack. The input parameter $MinPts$ determines the minimum cluster size and the parameter t was discussed above. Finally the parameter w influences the gradient vectors and proportionally depends on the reachability values of the objects in CO.

4 Cluster Representatives

In this section, we present three different approaches to determine representative objects for clusters computed by OPTICS. A simple approach could be to superimpose all objects of a cluster to build the representative as it is depicted in Figure 5. However, this approach has the huge drawback that the representatives on a higher level of the cluster hierarchy become rather unclear. Therefore, we choose real objects of the data set as cluster representatives.

In the following, CO denotes the cluster ordering from which we want to extract clusters. A cluster $C \subseteq CO$ will be represented by a set of k objects of the cluster, denoted as $\text{REP}(C)$. The number of representatives k can be a user defined number or a number which depends on the size and data distribution of the cluster C.

```
algorithm gradient_clustering(ClusterOrdering CO, Integer MinPts, Real t)

   startPts := emptyStack;
   setOfClusters := emptySet;
   currCluster := emptySet;

   o := CO.getFirst();                          // first object is a starting point
   startPts.push(o);

   WHILE o.hasNext() DO                         // for all remaining objects
      o := o.next;

      IF o.hasNext() THEN
         IF II(o) > t THEN                      // inflexion point
            IF gd(o) > 0 THEN
               IF currCluster.size() >= MinPts THEN
                  setOfClusters.add(currCluster);
               ENDIF
               currCluster := emptySet;
               IF startPts.top().R <= o.R THEN
                  startPts.pop();
               ENDIF
               WHILE startPts.top().R < o.R DO
                  setOfClusters.add(set of objects from startPts.top() to last end point);
                  startPts.pop();
               ENDDO
               setOfClusters.add(set of objects from startPts.top() to last end point);
               IF o.next.R < o.R THEN           // o is a starting point
                  startPts.push(o);
               ENDIF
            ELSE
               IF o.next.R > o.R THEN           // o is an end point
                  currCluster := set of objects from startPts.top() to o;
               ENDIF
            ENDIF
         ENDIF
      ELSE                                      // add clusters at end of plot
         WHILE NOT startPts.isEmpty() DO
             currCluster := set of objects from startPts.top() to o;
             IF (startPts.top().R > o.R) AND (currCluster.size() >= MinPts) THEN
                setOfClusters.add(currCluster);
             ENDIF
             startPts.pop();
         ENDDO
      ENDIF
   ENDDO

   RETURN setOfClusters;
END. // gradient_clustering
```

Figure 9: Pseudo code of the Gradient Clustering algorithm.

4.1 The Extended Medoid Approach

Many partitioning clustering algorithms are known to use medoids as cluster representatives. The medoid of a cluster C is the closest object to the mean of all objects in C.

The mean of C is also called centroid. For $k > 1$ we could choose the k closest objects to the centroid of C as representatives.

The choice of medoids as cluster representative is somehow questionable. Obviously, if C is not of convex shape, the medoid is not really meaningful.

An extension of this approach coping with the problems of clusters with non-convex shape is the computation of k medoids by applying a k-medoid clustering algorithm to the objects in C. The clustering using a k-medoid algorithm is rather efficient due to the expectation that the clusters are much smaller than the whole data set. This approach can also be easily extended to cluster hierarchies. At any level we can apply the k-medoid clustering algorithm to the merged set of objects from the child clusters or – due to performance reasons – merge the medoids of child clusters and apply k-medoid clustering on this merged set of medoids.

4.2 Minimizing the Core-Distance

The second approach to choose representative objects of hierarchical clusters uses the density-based clustering notion of OPTICS. The core-distance $o.C$ = Core-Dist(o) of an object $o \in CO$ (cf. Definition 1) indicates the density of the surrounding region. The smaller the core-distance of o, the denser the region surrounding o. This observation led us to the choice of the object having the minimum core-distance as representative of the respective cluster. Formally, Rep(C)

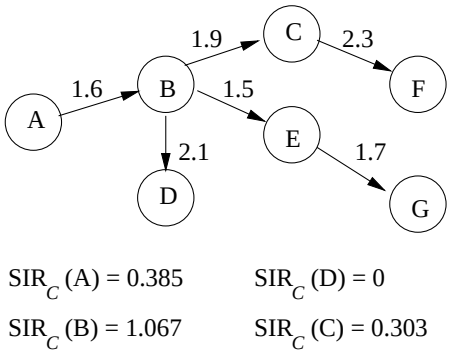

Figure 10: Sample successor graph for a cluster of seven objects. Some of the according SIR-values are depicted.

can be computed as:

$$\text{REP}(C) := \{o \in C \mid \forall x \in C : o.C \leq x.C\}.$$

We choose the k objects with the minimum core-distances of the cluster as representatives.

The straightforward extension for cluster hierarchies is to choose the k objects from the merged child clusters having the minimum core-distances.

4.3 Maximizing the Successors

The third approach to choose representative objects of hierarchical clusters also uses the density-based clustering notion of OPTICS but in a more sophisticated way. In fact, it makes use of the density-connected relationships underlying the OPTICS algorithm.

As mentoined above, the result of OPTICS is an ordering of the database minimizing the reachability relation. At each step of the ordering, the object o having the minimum reachability wrt. the already processed objects occurring before o in the ordering is choosen. Thus, if the reachability of object o is not ∞, it is determined by Reach-Dist(p, o) where p is an unique object located before o in the cluster ordering. We call p the *predecessor* of o, formally:

DEFINITION 7. (PREDECESSOR) *Let CO be a cluster ordering. For each entry $o \in CO$ the* predecessor *is defined as*

$$Pre(o) = \begin{cases} p & \text{if} \quad o.R = \textit{Reach-Dist}(p, o) \\ \text{UNDEFINED} & \text{if} \quad p.R = \infty. \end{cases}$$

Intuitively, $Pre(o)$ is the object in CO from which o has been reached. Let us note, that an object and its predecessor need not to be adjacent in the cluster ordering.

DEFINITION 8. (SUCCESSOR) *Let DB be a database of objects. For each object $o \in DB$ in a cluster ordering computed by OPTICS, the set of* successors *is defined as $S(o) := \{s \in DB \mid Pre(s) = o\}$.*

Let us note, that objects may have no predecessor, e.g. each object having a reachability of ∞ does not have a predecessor, including the first object in the ordering. On the other hand, some objects may have more than one successor. In that case, some other objects have no successors. Again, an object and its successors need not to be adjacent in the ordering.

We can model this succsessor-relationship within each cluster as a directed *successor graph* where the nodes are the objects of one cluster and a directed edge from object o to s represents the relationship $s \in S(o)$. Each edge (x, y) can further be labeled by Reach-Dist(x, y) $(= y.R)$. A sample successor graph is illustrated in Figure 10.

For the purpose of computing representatives of a cluster, the objects having many successors are interesting. Roughly speaking, these objects are responsible for the most density-connections within a cluster. The reachability values of these "connections" further indicate the distance between the objects. For example, for the objects in the cluster visualized in Figure 10, object B is responsible for the most density-connections since its node in the successor graph has the most out-going edges.

Our third strategy selects the representatives of clusters by maximizing the number of successors and minimizing the according reachabilities. For this purpose, we compute for each object o of a cluster C, the *S*um of the *I*nvers *R*eachability distances of the successors of o within C, denoted by $\text{SIR}_C(o)$:

$$\text{SIR}_C(o) := \begin{cases} 0 & \text{if} \quad S(o) = \emptyset \\ \sum\limits_{\substack{s \in S(o), \\ s \in C}} \frac{1}{1+\text{Reach-Dist}(o,s)} & \text{otherwise.} \end{cases}$$

We add 1 to Reach-Dist(o, s) in the denominator to weight the impact of the number of successors over the significance of the reachability values. Based on $\text{SIR}_C(o)$, the representatives can be computed as follows:

$$\text{REP}(C) := \{o \in C \mid \forall x \in C : \text{SIR}_C(o) \geq \text{SIR}_C(x)\}.$$

In Figure 10, the SIR-values of some objects of the depicted successor graph for a cluster of seven objects are computed. Since D has no successors, $\text{SIR}_C(D)$ is zero. In fact object B has the highest SIR-value indicating the central role of B in the cluster: B has three successors with relatively low reachability distance values. Our third strategy would select object B as representative for the cluster.

Let us note, that there is no additional overhead to compute the reachability distances Reach-Dist$(o, S(o))$ for each $o \in CO$ since these values have been computed by OPTICS during the generation of CO and Reach-Dist$(o, S(o))$ $= S(o).R$.

If we want to select k representatives for C we simply have to choose the k objects with the maximum SIR_C values.

5 System Architecture

The development of the industrial prototype BOSS is a first step towards developing a comprehensive, scalable and distributed computing solution designed to make the efficiency of OPTICS and the analytical capabilities of BOSS

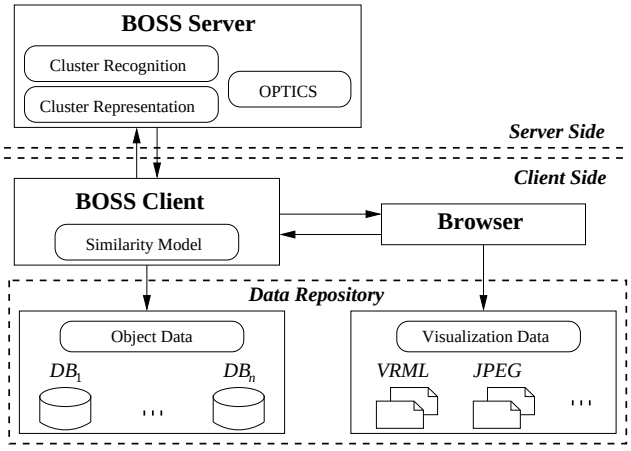

Figure 11: BOSS distributed architecture

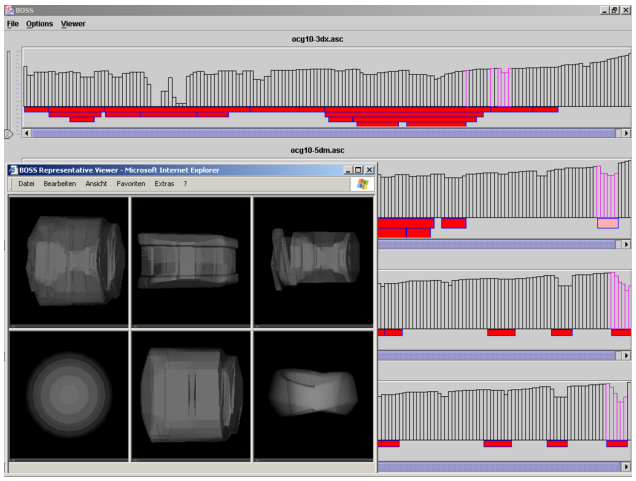

Figure 12: BOSS screenshot.

available to a broader audience. BOSS is a client/server system allowing users to provide their own data locally, along with an appropriate similarity model (cf. Figure 11).

The data provided by the user will be comprised of the objects to be clustered, as well as a data set to visualize these objects, e.g. VRML files for CAD data (cf. Figure 12) or JPEG images for multi-media data. Since this data resides on the user's local computer and is not transmitted to the server, heavy network traffic can be avoided. In order for BOSS to be able to interpret this data, the user must supply his own similarity model with which the reachability data can be calculated.

The independence of the data processing and the data specification enables maximum flexibility. Further flexibility is introduced through the support of external visual representation. As long as the user is capable of displaying the visualization data in a browser, e.g. by means of a suitable plug-in, the browser will then load web pages generated by BOSS displaying the appropriate data. Thus, multimedia data such as images or VRML files can easily be displayed (cf. Figure 12). By externalizing the visualization procedure, we can resort to approved software components, which have been specifically developed for displaying objects which are of the same type as the objects within our clusters.

6 Evaluation

We evaluated both the effectiveness and efficiency of our approaches using two real-world test data sets. The first one contains approximately 200 CAD objects from a German car manufacturer, and the second one is a sample of the Protein Databank [7] containing approximately 5000 protein structures. We tested on a workstation featuring a 1.7 GHz CPU and 2 GB RAM.

In the following, the three cluster recognition algorithms will vie among themselves, after which the three approaches for generating representatives will be evaluated.

6.1 Cluster Recognition

Automatic cluster recognition is clearly very desirable when analyzing large sets of data. In this section, we will first discuss the quality of our three cluster recognition algorithms. For this evaluation we use the Car and the Protein dataset. Secondly, we discuss the efficiency by using the Car and the Plane data set.

6.1.1 Effectivity

Both the Car and the Protein data set exhibit the commonly seen quality of unpronounced but nevertheless to the observer clearly visible clusters. The corresponding reachability plots of the two data sets are depicted in Figure 13.

Figure 13c shows that the `cluster_tree`-algorithm does not find any clusters at all in the Car data set, with the suggested default ratio-parameter of 75% [18]. In order to detect clusters in the CAR data set, we had to adjust the ratio-input parameter to 95%. In this case the `cluster_tree`-algorithm detected some clusters but missed out on some other important clusters and did not detect any cluster hierarchies at all. If we have rather high reachability values, e.g. values between 5-7 as in Figure 13 for the Car data set, the ratio-parameter for the `cluster_tree`-algorithm should be higher than for smaller values. In the case of the Protein data set we detected three clusters with the default parameter setting, but again missed out on some important clusters. Generally, in cases where a reachability graph consists of rather high reachability values or does not present spikes at all, but clusters are formed by smooth troughs in the waveform, this cluster recognition algorithm is unsuitable. Furthermore, it is inherently unable to detect narrowing clusters where a cluster has one sub-cluster of increased density (cf. Figure 6).

On the other hand, the ξ-clustering approach successfully recognizes some clusters while also missing out on significant subclusters (cf. Figure 13b). This algorithm has some trouble recognizing cluster structures with a significant differential of "steepness". For instance, in Figure 6 it does

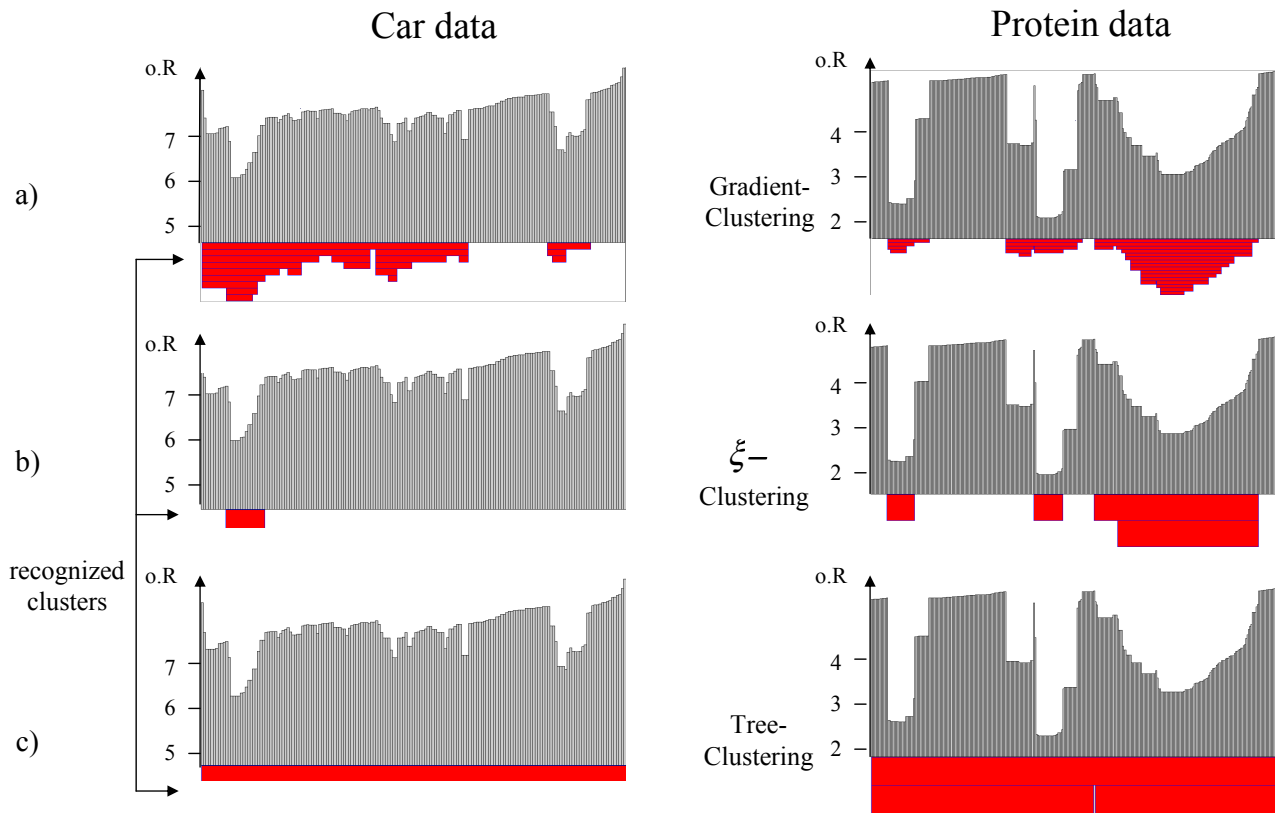

Figure 13: Sample cluster of car parts. a) Gradient-Clustering, b) ξ-Clustering, c) Tree-Clustering

not detect the narrowing cluster B inside of cluster A because it tries to create steep down-areas containing as many points as possible. Thus, it will merge the two steep edges if their steepness exceeds the threshold ξ. On the other, it is able to detect cluster C within A.

Finally, we look at our new *gradient-clustering* algorithm. Figure 13a shows that the recognized cluster structure is close to the intuitive one, which an experienced user would manually derive. Clusters which are clearly distinguishable and contain more than $MinPts$ elements are detected by this algorithm. Not only does it detect a lot of clusters, but it also detects a lot of meaningful cluster hierarchies, consisting of narrowing subclusters.

To sum up, in all our tests the *gradient-clustering* algorithm detected much more clusters than the other two approaches, without producing any redundant and unnecessary cluster information.

6.1.2 Efficiency

In all tests, we first created the reachability plots and then applied the algorithms for cluster recognition and representation. Let us note that we could also have integrated the *gradient-clustering* into the OPTICS run without causing any noteworthy overhead.

The overall runtimes for the three different cluster recognition algorithms are depicted in Table 1. Our new *gradient-clustering* algorithm does not only produce the most meaningful results, but also in sufficiently short time. This is due to its runtime complexity of $O(n)$.

6.2 Cluster Representation

After a cluster recognition algorithm has analyzed the data, algorithms for cluster representation can help to get a quick visual overview of the data. With the help of representatives, large sets of objects may be characterized through a single object of the data set. We extract sample clusters from both data sets in order to evaluate the different approaches for cluster representatives. In our first tests, we set the number of representatives to $k = 1$.

The objects of one cluster from the car data set are displayed in Figure 14 and the objects of one cluster from the protein data set are displayed in Figure 15. The annotated objects are the representatives computed by the respective algorithms. Both the *Maximum Successor* and the *Minimum Core Distance* approaches give good results. Despite the slight inhomogeneity of the clusters, both representatives sum up the majority of the elements within both clusters. This cannot be said of the representatives computed by the commonly used medoid method, which selects objects from the trailing end of the cluster. These two clusters and their corresponding representatives are no isolated cases, but reflect our general observations. Nevertheless, there have

	Car data (200 parts)	Protein data (5,000 molecules)
ξ-clustering	0.221 s	5.057 s
`cluster_tree`	0.060 s	1.932 s
Gradient Clustering	0.310 s	3.565 s

Table 1: CPU time for cluster recognition.

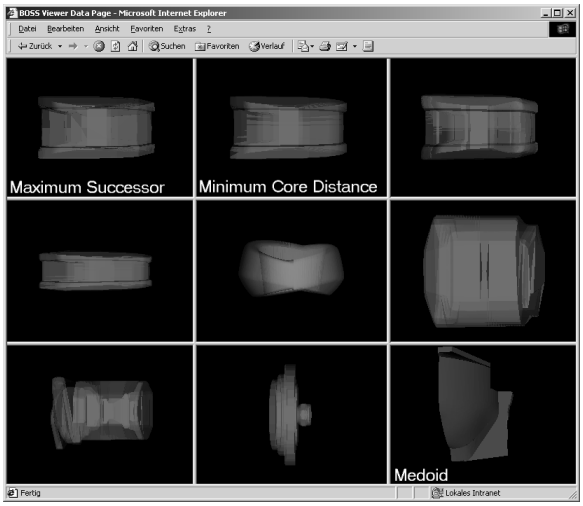

Figure 14: A cluster of CAD objects with corresponding representative objects.

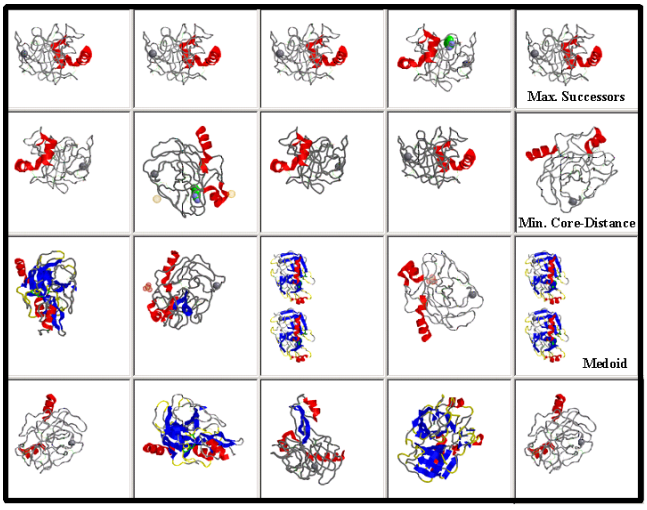

Figure 15: A cluster of proteins with corresponding representative objects.

been some rare cases where the medoid approach yielded the more intuitive representative than the other two approaches.

If we allow a higher number of representatives, for instance $k = 3$, it might be better to display the representatives of all three approaches to reflect the content of the cluster, instead of displaying the three best representatives of one single approach. If we want to confine ourselves to only one representative per cluster, the best possible choice is to use the representative of the *Maximum Successor*-approach.

6.3 Summary

The results of our experiments show, that our new approaches for the automatic cluster extraction and for the determination of representative objects outperform existing methods. It theoretically and empirically turned out, that our *gradient-clustering* algorithm seems to be more practical than recent work for automatic cluster extraction from hierarchical cluster representations. We also empirically showed that our approaches for the determination of cluster representatives is in general more suitable than the simple (extended) medoid approach.

7 Conclusions

In this paper, we proposed hierarchical clustering combined with automatic cluster recognition and selection of representatives as a promising visualization technique. Its areas of application include visual data mining, similarity search and evaluation of similarity models. We surveyed three approaches for automatic extraction of clusters. The first method, ξ-clustering, fails to detect some clusters present in the clustering structure and suffers from the sensitivity concerning the choice of its input parameter. The algorithm `cluster_tree` is by design unsuitable in the presence of narrowing clusters. To overcome these shortcomings, we proposed a new method, called *gradient-clustering*. The experimental evaluation showed that this algorithm is able to extract narrowing clusters. Furthermore, it can easily be integrated into the hierarchical clustering algorithm. Thus, it produces no noteworthy overhead. The cluster hierarchies produced by the *gradient-clustering* are similar to the clustering structures which an experienced user would manually extract.

Furthermore, we presented three different approaches to determine representative objects for clusters. The commonly known medoid approach is shown to be questionable for real-world data, while the approaches minimizing the core-distance and maximizing the successors both deliver good results.

Finally, we described our industrial prototype, called BOSS, that implements the algorithms presented in this paper.

In our future work we will apply our new prototype to various application ranges.

References

[1] R. Agrawal, C. Faloutsos, and A. Swami. "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)*, pages 69–84. Springer, 1993.

[2] R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim. "Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases". In *Proc. 21th Int. Conf. on Very Large Databases (VLDB'95)*, pages 490–501, 1995.

[3] M. Ankerst. *Visual Data Mining*. PhD thesis, Institute for Computer Science, University of Munich, 2000.

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

[5] S. Berchtold, D. A. Keim, and H.-P. Kriegel. "Using Extended Feature Objects for Partial Similarity Retrieval". *VLDB Journal*, 6(4):333–348, 1997.

[6] S. Berchtold and H.-P. Kriegel. "S3: Similarity Search in CAD Database Systems". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'97), Tucson, AZ*, pages 564–567, 1997.

[7] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. "The Protein Data Bank". *Nucleic Acids Research*, 28:235–242, 2000.

[8] M. M. Breunig, H.-P. Kriegel, P. Kröger, and J. Sander. "Data Bubbles: Quality Preserving Performance Boosting for Hierarchical Clustering". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'01), Santa Barbara, CA*, pages 79–90, 2001.

[9] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "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*, pages 291–316. AAAI Press, 1996.

[10] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, et al. "Efficient and Effective Querying by Image Content". *Journal of Intelligent Information Systems*, 3:231–262, 1994.

[11] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. "Fast Subsequence Matching in Time-Series Databases". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'94), Minneapolis, MN*, pages 419–429, 1994.

[12] H. V. Jagadish. "A Retrieval Technique for Similar Shapes". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'91)*, pages 208–217, 1991.

[13] D. A. Keim. "Efficient Geometry-based Similarity Search of 3D Spatial Databases". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), Philadelphia, PA*, pages 419–430, 1999.

[14] H.-P. Kriegel, S. Brecheisen, P. Kröger, M. Pfeifle, and M. Schubert. "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.

[15] H.-P. Kriegel, P. Kröger, and I. Gotlibovich. "Incremental OPTICS: Efficient Computation of Updates in a Hierarchical Cluster Ordering". In *Proc. Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK'03), Prague, Czech Republic*, volume 2737 of *Lecture Notes in Computer Science (LNCS)*, pages 224–233. Springer, 2003.

[16] H.-P. Kriegel, P. Kröger, Z. Mashael, M. Pfeifle, M. Pötke, and T. Seidl. "Effective Similarity Search on Voxelized CAD Objects". In *Proc. 8th Int. Conf. on Database Systems for Advanced Applications (DASFAA'03), Kyoto, Japan*, 2003.

[17] J. McQueen. "Some Methods for Classification and Analysis of Multivariate Observations". In *5th Berkeley Symp. Math. Statist. Prob.*, volume 1, pages 281–297, 1967.

[18] J. Sander, X. Qin, Z. Lu, N. Niu, and A. Kovarsky. "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.