Visual Mining of Cluster Hierarchies

Hans-Peter Kriegel, Stefan Brecheisen, Eshref Januzaj, Peer Kröger, Martin Pfeifle

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

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. [10], [1], [8], [9], [2], [6], [11].

Particularly, the task of finding similar shapes in 2-D and 3-D becomes more and more important. Examples for new applications that require the retrieval of similar 3D objects include databases for molecular biology, medical imaging and computer aided design.

Hierarchical clustering was shown to be effective for evaluating similarity models [12],[13]. 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* (*Browsing OPTICS-Plots for Similarity Search*). BOSS is based on techniques related to *visual data mining*. It helps to visually analyze cluster hierarchies by providing meaningful cluster representatives.

The remainder of the paper is organized as follows: After briefly introducing reachability plots, we present in Section 2 the application areas of hierarchical 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 the notions of cluster recognition and cluster representatives respectively, which form the theoretical foundations of BOSS. In Section 5, we describe the actual industrial prototype we developed and evaluate its usefulness in Section 6. The paper concludes in Section 7 with a short summary and a few remarks on future work.

2 Hierarchical 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 reachability plots computed by OPTICS, which served as a starting point for BOSS. The technical aspects related to BOSS are described later in Section 5.

2.1 Reachability Plots

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 [15]).
- There exist a very efficient variant of the OPTICS algorithm which is based on data bubbles [7], where we have to trade only very little quality of the clustering result for a great increase in performance.
- There exist an efficient incremental version of the OPTICS algorithm [13].

The reachability plots computed by OPTICS help the user to get a meaningful and quick overview over a large data set. The output of OPTICS is a linear ordering of the database objects minimizing a binary relation called *reachability* which is in most cases equal to the minimum distance of each database object to one of its predecessors in the ordering. Instead of a dendrogram, which is the common representation of hierarchical clusterings, the resulting reachability plot is much easier to analyse. The reachability values can be plotted for each object of the cluster-ordering computed by OPTICS. 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.



Fig.1: Reachability plots computed by optics (right) for a 2D dataset (left)

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 1: 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.

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.

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 2a).

- 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 2b).
- 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 2c).

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



Fig.2: Different approaches to visual data mining

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.

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 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 reachability 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 3).
- The user is not only interested in the shape and the number of the clusters, but also in the specific parts 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 4). 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



Fig.3: Browsing through reachability plots with different density thresholds ε_{cut}

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.

Evaluation of Similarity Models. In general, similarity models can be evaluated by computing *k*-nearest neighbor queries (*k*-nn queries). As shown in [14], 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.



Fig.4: Hierarchically ordered representatives



Fig.5: Sample narrowing clustersa) data space, b) reachability plot and c) cluster hierarchy

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 which are both 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. [16]. 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 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 maximum *m* and the average reachabilities of the regions to the left and to the right of *m*. The authors in [16] 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 consisting of exactly one smaller sub-cluster (cf. Figure 5). 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 5 illustrates this fact. The narrowing cluster A consists of one cluster B which is itself narrowing consisting of one cluster C. The algorithm cluster_tree



a) detection of root clusters, b) detection of subclusters

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 dependent on the ξ -parameter but also fails to detect the cluster hierarchy.

3.2 Drop-Down Clustering

This new cluster recognition algorithm is based on the novel concept of *inflexion points* which allow the detection of narrowing subclusters. A point *o* is an inflexion point iff the gradient of the reachability values changes considerably (cf. Figure 6b).

Since our method works in a top-down fashion, we call it *Drop-Down Clustering*. The idea behind is the successive use of the visual interpretation of the cluster ordering -as described in Figure 1- which is based on the fact that the reachability plot can be cut by any level ε_{cut} to the abscissa to extract a clustering. Starting from an initial clustering we simply drop the ε_{cut} -value in order to find substructures. Since it is not practical to test each possible ε_{cut} -value, we have to extract interesting values for a cut from the reachability values of the objects.

The Drop-Down Clustering algorithm starts by generating an initial root clustering (cf. Figure 6a). This does not contain all elements of the plot, as clusters separated by noise are assumed to be not related. Basically, a set of clusters forms the basis for a set of hierarchical clusters. This initial clustering is generated as follows: The objects in the reachability plot are sorted by descending reachability distance while retaining relative order among equal elements. The sorted list is now scanned until two objects are found whose indices are more than *MinPts* apart, indicating that every element in-between these

two is smaller than either, thus constituting a dip in the graph. A top level cluster has been found, and all elements included in this cluster are removed from the sorted list. The scan can now continue until all elements have been removed or viewed.

The second part of the algorithm now separately analyzes each cluster found during the initial clustering (cf. Figure 6b). The extraction of further (sub-)clusters is a recursive procedure. The procedure starts with a set of elements from the reachability graph which is sorted by descending reachability values where elements having the same reachability value are arranged according to the cluster ordering. This list is sequentially tested for an object which is an inflexion point. An inflexion point can either indicate the start or end of a narrowing subcluster, or be responsible for two new subclusters.

Should a subcluster be found, it may be added to the resulting cluster hierarchy, after which it is then recursively processed to discover potential substructures. All discovered subclusters must conform to the following constraints:

- The minimum cluster size constraint of *MinPts* objects must be satisfied, i.e. there are at least *MinPts* objects located between the start point and the end point of the cluster (e.g. Cluster *B* in Figure 5 must contain at least *MinPts* objects).
- The current cluster has at least *MinPts* objects less than the cluster of its parent node in the hierarchy (e.g. Cluster *B* in Figure 5 must have at least *MinPts* objects less than cluster *A*).

Let us note, that we could also claim a minimum ratio of reachabilities at the boundary of a cluster and inside a cluster as postulated in [16] or increment/decrement the required minimum cluster size.

Obviously, the Drop-Down algorithm is able to extract narrowing clusters. Experimental comparisons with the methods in [16] and [4] are presented in Section 6.

4 Cluster Representatives

In this section, we present different approaches to determine representatives 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 4. 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, we assume that *DB* is a database of multimedia objects, *dist*: $DB \times DB \rightarrow IR$ is a metric distance function on objects in *DB* and $N_{\varepsilon}(o) := \{q \in DB \mid dist(o,q) \le \varepsilon\}$ where $o \in DB$ and $\varepsilon \in IR$. A cluster $C \subseteq DB$ is represented by a set of *k* objects of the cluster, denoted as *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*.

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 The Minimum Core-Distance Approach

The second approach to choose representative objects of hierarchical clusters uses the density-based clustering notion of OPTICS. To compute the reachability, OPTICS determines for each object the so called *core-distance*:

Definition 1 (Core-distance).

Let *o* be an object from a database *DB*, let ε be a distance value, let $N_{\varepsilon}(o)$ be the ε -neighborhood of *o*, let *MinPts* be a natural number and let *MinPts-distance(o)* be the distance from *o* to its *MinPts*-nearest neighbor. Then, the *core-distance* of *o* is defined as:

 $core-distance(o) = \begin{cases} \infty, & \text{if } |N_{\varepsilon}(o)| < MinPts \\ MinPts-distance(o), & \text{otherwise} \end{cases}$

The core-distance of an object indicates the density of the surrounding region. The smaller the core-distance of an object 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) can be computed as:

 $REP(C) := \{ o \in C \mid \forall x \in C : core-distance(o) \le core-distance(x) \}.$

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 The Maximum Successor Approach

Based on the core-distance, the reachability-distance (or short: reachability) is defined as:

Definition 2 (*Reachability-Distance*). Let $o \in DB$, let $N_{\varepsilon}(o)$ be the ε -neighborhood of o, let *MinPts* be a natural number. Then, the *reachability-distance* of $p \in DB$ with respect to o is defined as: reachability-distance (p,o) = max (core-distance(o), distance(p,o))



Fig.7: Sample successor graph for a cluster of seven objects

The result of OPTICS is an ordering of the database minimizing the reachability relation. At each step of the ordering, the object *p* having the minimum reachability wrt. the already processed objects occurring before *p* in the ordering is chosen. Thus, if the reachability of object *p* is not ∞ , it is determined by *reachability-distance(p,o)* where *o* is an object located before *p* in the cluster ordering. We call *o* the *predecessor* of *p*.

Definition 3 (*Successors*). Let $o \in DB$. Then, the set of successors is defined as $S(o) := \{s \in DB \mid o \text{ is predecessor of } s\}$.

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.

We can model this successor-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 *reachability-distance* (x,y). A sample successor graph is illustrated in Figure 7.

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.

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 Sum of the Inverse Reachability distances of the successors of o within C, denoted by $SIR_C(o)$:

$$SIR_{C}(o) = \begin{cases} 0, & \text{if } S(o) = \emptyset\\ \sum_{\substack{s \in S(o)\\s \in C}} \frac{1}{1 + reachability - distance(s, o)}, & \text{otherwise} \end{cases}$$

We add 1 to *reachability-distance* (s,o) in the denominator to weight the impact of the number of successors over the significance of the reachability values. Based on SIR(o), the representatives can be computed as follows:

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



Fig.8: BOSS distributed architecture

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 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 8).

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 9) 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



Fig.9: BOSS screenshot



Fig.10: Sample Clustering of CAR Parts **a**) *Drop-Down-Clustering*, **b**) *ξ-Clustering* and **c**) *Tree-Clustering*

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 9). 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 realworld test data sets. The first one contains approximately 200 CAD objects from a German car manufacturer, the second one 5000 CAD objects from an American aircraft producer. We tested on a workstation with a 1.7 GHz CPU and 2 GB RAM.

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

Cluster Recognition. Automatic cluster recognition is clearly very desirable when analyzing large sets of data. In this case, we will be looking at a subset of a database of CAD objects representing car parts. The results are depicted in Figure 10.

This data exhibits the commonly seen quality of unpronounced but nevertheless to the observer clearly visible clusters. The *Tree-Clustering* algorithm does not find any clusters at all, whereas the ξ -clustering approach successfully recognizes some clusters while missing out on significant subclusters. On the other hand, our new Drop-Down-Algorithm detects many clusters. Furthermore, it detects a lot of meaningful cluster hierar-

	CAR (200 parts)	PLANE (5000 parts)
x-clustering	0.348 s	9.714 s
cluster_tree	0.130 s	3.019 s
Drop-Down clustering	0.104 s	0.707 s

Fig.11: CPU time for cluster recognition

chies, consisting of narrowing subclusters. To sum up, in all our tests the Drop-Downalgorithm detected much more clusters than the other two approaches, without producing any redundant and unnecessary cluster information.

The overall runtime for the three different cluster recognition algorithms is depicted in Figure 11. Our new Drop-Down clustering algorithm does not only produce the most meaningful results, but also in the shortest time. It seems to be the only algorithm which is suitable for interactive use, if the data sets contain several thousand elements.

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 a sample cluster from the plot depicted in Figure 10a in order to evaluate the different approaches for cluster representatives. In our first tests, we set the number of representatives k to 1.

The objects of one cluster are displayed in Figure 12. The three 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 cluster, both representatives sum up the majority of elements within this cluster. This cannot be said of the representative computed by the commonly used medoid method, which selects an object from the trailing end of the cluster.

Summary. The results of our experiments show, that our new approaches for the automatic cluster extraction and for the determination of representative objects outperform



Fig.12: Representatives displayed by the BOSS object viewer

existing methods. It theoretically and empirically turned out, that our Drop-Down-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 most likely 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 obviously unsuitable in the presence of narrowing clusters. To overcome these shortcomings, we proposed a new method, called *Drop-Down-Clustering*. The experimental evaluation showed that this algorithm is able to extract narrowing clusters. The cluster hierarchies produced by the Drop-Down-Algorithm 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 unsuitable 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 concentrate on efficient algorithms for incremental hierarchical clustering. These new cluster algorithms should allow an easy determination of the cluster hierarchy together with an easy extraction of meaningful representatives.

References

- Agrawal R., Faloutsos C., Swami A. *Efficient Similarity Search in Sequence Databases*. 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. Agrawal R., Lin K.-I., Sawhney H., Shim K.: Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases. Proc. 21th Int. Conf. on Very Large Databases (VLDB 95), pages 490-501, 1995.
- 3. Ankerst M.: Visual Data Mining. PhD thesis, Institute for Computer Science, University of Munich, 2000.
- Ankerst M., Breunig M. M., Kriegel H.-P., Sander J.: "OPTICS: Ordering Points To Identify the Clustering Structure", Proc. Int. Conf. on Management of Data (SIG-MOD), Philadelphia, PA, 1999, pp. 49-60.
- 5. Berchtold S., Keim D. A., Kriegel H.-P.: Using Extended Feature Objects for Partial Similarity Retrieval. VLDB Journal, 6(4):333-348, 1997.

- Berchtold S., Kriegel H.-P.: S3: Similarity Search in CAD Database Systems. Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 97), Tucson, AZ, pages 564-567, 1997.
- Breuning M., Kriegel H.-P., Kröger P., Sander J.: Data Bubbles: Quality Preserving Performance Boosting. Proc. ACM SIGMOD 2001 Int. Conf. on Management of Data, Santa Barbara, CA, 2001
- Faloutsos C., Equitz M., Flickner M., Niblack W., Petkovic D., Barber R.: *Efficient* and Effective Querying by Image Content. Journal of Intelligent Information Systems, 3:231-262, 1994.
- Faloutsos C., Ranganathan M., Manolopoulos Y.: Fast Subsequence Matching in Time-Series Databases. Proc. ACM SIGMOD Int. Conf. on Management of Data, Minneapolis, MN, pages 419-429, 1994.
- 10. Jagadish H. V.: A *Retrieval Technique for Similar Shapes*. Proc. ACM SIGMOD Int. Conf. on Management of Data, 208-217,1991.
- 11. Keim D.: Efficient Geometry-based Similarity Search of 3D Spatial Databases.Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 99), Philadelphia, PA, pages 419-430, 1999.
- Kriegel H.-P., Brecheisen S., Kröger P., Pfeifle M., Schubert M.: Using Sets of Feature Vectors for Similarity Search on Voxelized CAD Objects. Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'03), San Diego, CA, 2003.
- Kriegel H.-P., Kröger P., Gotlibovich I.: Incremental OPTICS: Efficient Computation of Updates in a Hierarchical Cluster Ordering. Proc. 5th Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK'03), Prague, Czech Republic, 2003.
- Kriegel H.-P., Kröger P., Mashael Z., Pfeifle M., Pötke M., Seidl T.: *Effective Similarity Search on Voxelized CAD Objects*. Proc. 8th Int. Conf. on Database Systems for Advanced Applications (DASFAA'03), Kyoto, Japan, 2003.
- 15. McQueen J.: Some Methods for Classification and Analysis of Multivariate Observations. 5th Berkeley Symp. Math. Statist. Prob., volume 1, pages 281–297, 1967.
- Sander J., Qin X., Lu Z., Niu N., Kovarsky A.: Automatic Extraction of Clusters from Hierarchical Clustering Representations. Proc. 7th Pacic-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2003), Seoul, Korea, 2003.