

Automatic Extraction of Clusters from Hierarchical Clustering Representations

Jörg Sander, Xuejie Qin, Zhiyong Lu, Nan Niu, Alex Kovarsky
Department of Computing Science, University of Alberta
Edmonton, AB, Canada T6G 2E8
{joerg, xuq, zhiyong, nan, kovarsky}@cs.ualberta.ca

Abstract

Hierarchical clustering algorithms are typically more effective in detecting the true clustering structure of a data set than partitioning algorithms. However, hierarchical clustering algorithms do not actually create clusters, but compute only a hierarchical representation of the data set. This makes them unsuitable as an automatic pre-processing step for other algorithms that operate on detected clusters. This is true for both dendrograms and reachability plots, which have been proposed as hierarchical clustering representations, and which have different advantages and disadvantages. In this paper we first investigate the relation between dendrograms and reachability plots and introduce methods to convert them into each other showing that they essentially contain the same information. Based on reachability plots, we then introduce a technique that automatically determines the significant clusters in a hierarchical cluster representation. This makes it for the first time possible to use hierarchical clustering as an automatic pre-processing step that requires no user interaction to select clusters from a hierarchical cluster representation.

Keywords: Hierarchical clustering, OPTICS, Single-Link method, dendrogram, reachability-plot

1. Introduction and Related Work

Knowledge discovery in databases (KDD) is the non-trivial process of identifying valid, novel, potentially useful, and understandable patterns in large amounts of data. One of the primary data analysis tasks in this process is cluster analysis. It is often a first and important step in analyzing a data set, understanding its properties, and preparing it for further analysis.

There are many different types of clustering algorithms for different types of applications. The most common distinction is between partitioning and hierarchical clustering algorithms (see e.g. [KR 90]).

Partitioning algorithms create a “flat” decomposition of a data set into a set of clusters. Examples of partitioning algorithms are the k -means [Mac 67] and the k -medoids algorithms PAM and CLARA [KR 90] and CLARANS [NH 94], or density-based approaches such as [JD 88], [EK SX 96], [SCZ 98], or [HK 98]. They need, however, in general some input parameters that specify either the number of clusters that a user wants to find or a threshold for point density in clusters. Correct parameters for partitioning algorithms, which allow the algorithm to reveal the true clustering structure of a data set, are in general difficult to determine, and they may not even exist.

Hierarchical clustering algorithms, on the other hand, do not actually partition a data set into clusters, but compute only a hierarchical representation of the data set, which reflects its possibly hierarchical clustering structure. The Single-Link method [Sib 73] is a well-known example of this type of method. Other algorithms such as Average-Link or Complete-Link, which produce similar hierarchical structures have also been suggested (see e.g. [JD 88], [HT 93]) and are widely used. The result of these algorithms is a dendrogram, i.e., a tree that iteratively splits a data set into smaller subsets until each subset consists of only one object. A different hierarchical clustering algorithm, which generalizes density-based clustering, is OPTICS [ABKS 99]. This algorithm produces another type of output representation, a so-called reachability plot, which is a bar plot of distance values showing clusters as “dents” in the plot.

Hierarchical clustering algorithms have several advantages over partitioning algorithms: they are more robust with respect to their input parameters, they are less influenced by cluster shapes, they are less sensitive to largely differing point densities of clusters, and they can represent nested clusters.

An important property of many real-data sets is that clusters of very different point densities may exist in different regions of the data space and that clusters may be nested, which makes it very hard to detect all these clusters using a partitioning algorithm. The reason is that in these cases global parameters, which are explicitly or implicitly required by most partitioning algorithms and which would characterize all clusters in the data space, do not exist. The 2-dimensional data depicted in Figure 1(a) – nested clusters, and in Figure 2(a) – largely differing densities illustrate the problems (the circles and ellipses in the figures should be ignored at this point, they illustrate an aspect that is explained later). In cases like these, hierarchical clustering algorithms are a better choice for detecting the true clustering structure.

There are, however, also problems with hierarchical clustering algorithms, and alleviating those problems is the focus of this paper. The first problem is that hierarchical clustering representations are sometimes not easy to understand, and depending on the application and the user’s preferences, one of the representations, dendrogram or reachability-plot, may be preferable, independent of the algorithm used.

For very small data sets, a dendrogram may give a clearer view of the cluster membership of individual points than a reachability plot, and in some application areas such as biology, domain experts may prefer tree representations because they are more used to it. However, dendrograms are much harder to read than reachability plots for more than a few hundred data points, since the diagram grows rapidly when the size of the data set increases. When the user has to scroll through several screens, identifying the clusters becomes very difficult, and the horizontal lines connecting the nodes may in fact visually obscure the clustering structure instead of helping the user to identify the clusters. When the data set is large, identifying the overall clustering structure is much easier in reachability plots. Figure 1 and Figure 2 show a reachability plot in part (b) of the figures, and a Single-Link dendrogram in part (c) of the figures, for the depicted data sets. To indicate which regions in the hierarchical clustering representations correspond to which clusters in the data sets, we used corresponding labels (“A” through “D”). More details about how to interpret these hierarchical clustering representations will be given in section 2. Roughly speaking, in a reachability plot, clusters are indicated by a “dent” in the plot; in a dendrogram, the nodes of the tree represent potential clusters.

The second problem with hierarchical clustering algorithms is that clusters are not explicit in the output of the algorithms and have to be determined somehow from the representation. There are two common approaches. In the first approach, a user selects and extracts manually each single cluster from a dendrogram or reachability plot, guided by a visual inspection of the graphical representation. The second

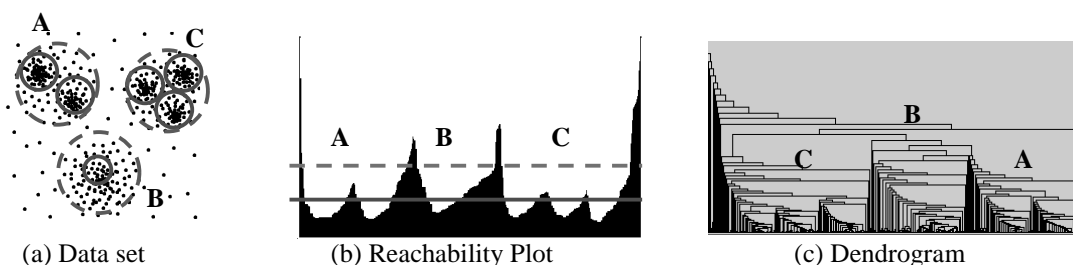


Figure 1: Example dataset 1: hierarchical clustering structure, i.e., nested clusters.

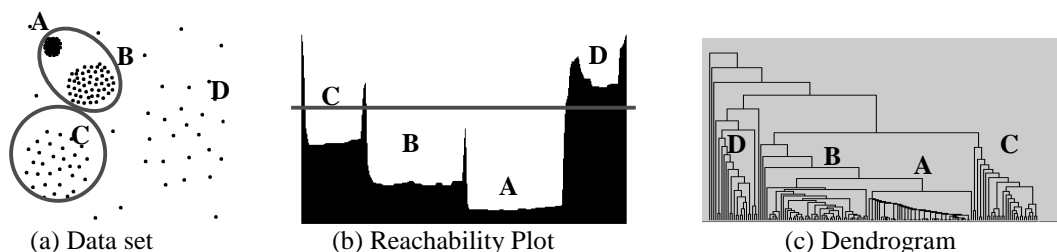


Figure 2: Example dataset 2: largely differing densities and sizes of clusters.

approach is semi-automatic: a user first defines a horizontal cut through the clustering representation, and then the resulting connected components (usually those having a minimum size) are automatically extracted as clusters. The latter approach has two major drawbacks: when clusters have largely differing densities, a single cut cannot determine all of the clusters, and secondly, it is often difficult to determine where to cut through the representation so that the extracted clusters are significant. The resulting clusters for some cut-lines through the given reachability plots are illustrated in Figure 1 and Figure 2. The circles and ellipse roughly describe the extracted clusters that correspond to the cut-line having the same style: solid or dashed. The example in Figure 2 also shows a case where there is clearly no cut-line through the hierarchical representation, which would determine all the clusters present in this data set.

Both approaches have the disadvantage that they are unsuitable for an automated KDD process in which the output of hierarchical clustering algorithms are the input of subsequent data mining algorithms, which analyze the data set on the level of clusters. A typical application of such a data mining process is, for instance, the description and characterization of detected clusters in a geographic database by higher-level features such as the diameter, the number neighbors of a certain type etc (see e.g., [KN96]). The ability to automatically extract clusters from hierarchical clustering representations as a preprocessing step to generate summary information for the clusters will also make it possible to track and monitor the changes of a hierarchical clustering structure over time in a dynamic data set.

To our knowledge, the only proposal for a method to extract automatically clusters from a hierarchical clustering representation can be found in [ABKS 99]. The authors propose a method for reachability plots that is based on the steepness of the “dents” in a reachability plot. Unfortunately, this method requires an input parameter, which is difficult to understand and hard to determine. There is no known and reliable heuristics to set the value for this parameter. The result of the method is in fact rather sensitive to the value of the input parameter. Small variations in the parameter can result in drastic changes to the final hierarchical clustering structure extracted. For large and complex hierarchical structures, this method certainly can speed-up the process of manually extracting significant clusters from a reachability plot, if a user can find a suitable parameter setting after trying several different values. But it is definitely not suitable in an automatic preprocessing step in a larger KDD process.

Our paper contributes to a solution of these problems in the following ways. First, we analyze the relation between hierarchical clustering algorithms that have different outputs, i.e. between the Single-Link method, which produces a dendrogram, and OPTICS, which produces a reachability plot. Although the algorithms and the resulting representations may seem significantly different, we will show that the clustering representations have essentially the same properties, and that they can even be considered equivalent in a special case. Based on this analysis we develop methods to convert dendrograms and reachability plots into each other, making it possible to choose the most advantageous representation, independent of the algorithm used. Converting a dendrogram into a reachability plot is also an important step in our new method for automatic cluster extraction from hierarchical clustering representations. We introduce a new technique to create a tree that contains only the significant clusters from a hierarchical representation as nodes. Our new tree has the additional advantage that it can be directly used as input to other algorithms that operate on detected clusters. The result of this method can be either a cluster tree (if the subsequent algorithms for characterizing and analyzing clusters can handle hierarchical structures), or only the leaves of this tree (if the algorithm requires a flat partitioning of the data set as an input). Selecting only the leaves from our cluster tree corresponds to selecting the most significant clusters simultaneously from different levels of a dendrogram or reachability plot (rather than applying a simple cut at a certain level of a hierarchical clustering representation, which may miss important clusters).

The rest of the paper is organized as follows. Section 2 analyzes the relation between dendrograms and reachability plots in detail, and introduces methods to convert between these two representations. In section 3, we propose and evaluate a method to automatically extract the significant clusters from a hierarchical clustering representation. This method is based on reachability plots. However, if a hierarchical clustering result is given as a dendrogram the method can still be applied since the method proposed in section 2 can be used to convert the dendrogram into a reachability plot first. Section 4 concludes the paper with a summary and some directions for future research.

2. Converting Between Hierarchical Clustering Representations

2.1 The relation between the Single-Link Method and OPTICS

The Single-Link method and its variants like Average-Link or Complete-Link create a recursive hierarchical decomposition of a given data set. Starting with the clustering obtained by placing every object in a unique cluster; in every step the two closest clusters in the current clustering are merged until all points are in one cluster. Methods like Average-Link and Complete-Link differ only in the distance function that is used to compute the distance between two clusters.

The result of such a hierarchical decomposition is then represented by a dendrogram, i.e., a tree that represents the merges of the data set according to the algorithm. Figure 3 (left) shows a very simple data set and a corresponding Single-Link dendrogram, which is interpreted as follows: The root represents the whole data set, a leaf represents a single object, an internal node represents the union of all the objects in its sub-tree, and the height of an internal node represents the distance between its two child nodes.

OPTICS is a hierarchical clustering algorithm that generalizes the density-based notion of clusters introduced in [EK SX 96]. It is based on the notions of core-distance and reachability-distance for objects with respect to parameters Eps and $MinPts$. The parameter $MinPts$ allows the core-distance and reachability-distance of a point p to capture the point density around that point. Using the core- and reachability-distances, OPTICS computes a “walk” through the data set, and assigns to each object p its core-distance and the smallest reachability-distance $reachDist$ with respect to an object considered before p in the walk (see [ABKS 99] for details).

For the special parameter values $MinPts = 2$ and $Eps = \infty$, the core-distance of an object p is always the distance of p to its nearest neighbor, and the reachability-distance of an object p relative to an object q is always the distance between p and q . The algorithm starts with an arbitrary object assigning it a reachability-distance equal to ∞ . The next object in the output is then always the object that has the shortest distance d to any of the objects that were “visited” previously by the algorithm. The reachability-value assigned to this object is d . We will see in the next two subsections that a similar property holds for the sequence of object and the height of certain ancestor nodes between the objects in a dendrogram.

Setting the parameter $MinPts$ in OPTICS to larger values will weaken the so-called single-link effect (which, as a side effect, also smoothes the reachability plot), and if the dimension of the data set is not too high, OPTICS is more efficient than traditional hierarchical clustering algorithms since it can be supported by spatial index structures. The output is a *reachability plot*, which is a bar plot of the reachability values assigned to the point in the order they were visited. Figure 3 (right) illustrates the effect of the $MinPts$ parameter on a reachability plot. Such a plot is interpreted as following: “Valleys” in the plot represent clusters, and the deeper the “valley”, the denser the cluster. The tallest bar between two “valleys” is a lower bound on the distance between the two clusters. Large bars in the plot, not at the border of a cluster represent noise, and “nested valleys” represent hierarchically nested clusters.

Although there are differences between dendrograms and reachability plots, both convey essentially the same information. In fact, as we will investigate in the next subsections, there is a close relationship between a dendrogram produced by the Single-Link method and a reachability plot of OPTICS for $Eps = \infty$ and $MinPts = 2$, which allows us to convert between these results without loss of information.

For other algorithms such as Average-Link, Complete-Link, and OPTICS using higher values for $MinPts$, there is no strict equivalence of the results (although in practice the result will be similar since the detected clustering structures are a property of the data set and not a property of the algorithm used). Nevertheless, the proposed methods for converting between dendrograms and reachability plots are also

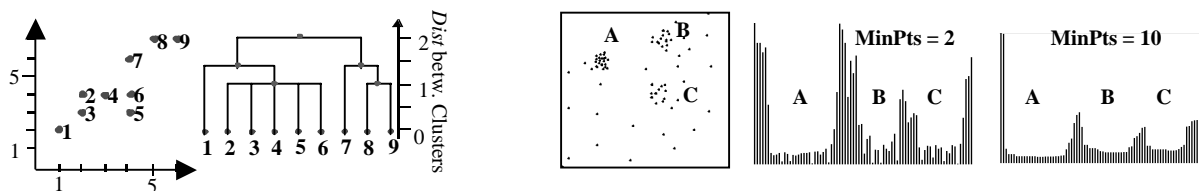


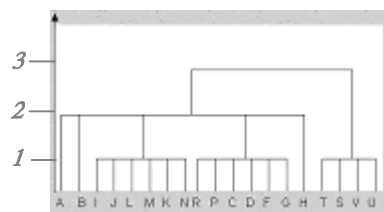
Figure 3: Left: Illustration of a Dendrogram – Right: Illustration of Reachability Plots

applicable in those cases. Converting, e.g., a dendrogram produced by the Average-Link method into a reachability plot will generate a reachability plot that contains the same cluster information as the dendrogram (even though there may be no *MinPts* parameter so that OPTICS would generate a strictly equivalent result). Vice versa: converting a reachability plot of OPTICS for a higher *MinPts* value into a dendrogram will result in a tree representation of the same clustering information as the reachability plot (even though no other hierarchical method may produce a strictly equivalent result). Thus, these conversion methods allow a user to choose the best representation, given the properties of the data set, the user’s preferences, or some application needs, e.g., algorithms that process a clustering result further and expect a certain input format – one example of such a method is our technique for automatic cluster extraction from a hierarchical clustering representations, proposed in section 3.

Please note that neither a dendrogram nor a reachability plot for a given data set is unique. There are many ways to draw a tree of merges as produced by, e.g., the Single-Link method. Intuitively: given one particular dendrogram, rotating the two children of any node in this representation and re-drawing the tree, we get another dendrogram, which is also a valid representation of the same clustering structure. For instance, in the dendrogram from Figure 3 (left), the right sub-tree of the root can as well be drawn first to the left, followed by the left sub-tree, drawn to the right. Using different starting points in the OPTICS algorithm will also produce different reachability plots, which however represent the same clustering structure. For instance, in Figure 3 (right), if we would start with a point in cluster C, the “dent” for that region would then come before the “dents” for clusters A and B.

2.2 Converting dendrograms into reachability plots

It is easy to see that the order in which the leaf nodes are drawn in a dendrogram satisfies the following



condition: for any two points in leaf nodes it holds that the height of their smallest common ancestor node in the tree is a lower bound for the distance between the two points. For instance, using the example dendrogram depicted to the left, the minimum distance between T (or any of the four points T, S, V, U) to I (or any other point among A, through R), is at least 3 since the smallest common ancestor node which separates between I and T is the root at height 3.

We get an even stronger condition if we assume that the dendrogram is organized in the following way: for any internal node (representing the merging of its two children), the right child is drawn in a way that the point in this sub-tree which is closest to the set of points in the left child (the point that determined the distance between the two children in the Single-Link method) is always the leftmost leaf in the sub-tree of the right child. This property can always be achieved via suitable rotations of child nodes in a given dendrogram. In such a dendrogram the height of the smallest common ancestor between a point p and p ’s left neighbor q is the minimum distance between p and the whole set of points to the left of p .

Recall that in a reachability plot of OPTICS (using $MinPts = 2$ and $Eps = \infty$) the height of a bar (the reachability value) for a point p is also the smallest distance between p and the set of points to the left of p in the reachability plot. This relationship allows us to easily convert a dendrogram into a reachability plot. We iterate through the leaves of a given dendrogram from left to right and assign each point its reachability value: the reachability value of the first point is (always) infinity in any reachability plot. For each other point p in the dendrogram the reachability value is simply the height of the smallest common ancestor of p and the left neighbor of p . In pseudo code:

```
leafs[0].r_dist = ∞;
for (i = 1; i < n; i++)
    leafs[i].r_dist = height_of_smallest_common_ancestor(leafs[i], leafs[i-1]);
```

The transformation of our example dendrogram can be illustrated as following (see Figure 4): For the first point in the dendrogram A: $r_dist = \infty$. For the next point B, the height of the smallest common ancestor of A and B is 2. The height of the smallest common ancestor of the next point I, and the previous point B is also 2. This is the smallest distance between I and the group of points $\{A, B\}$. Point I is the first point

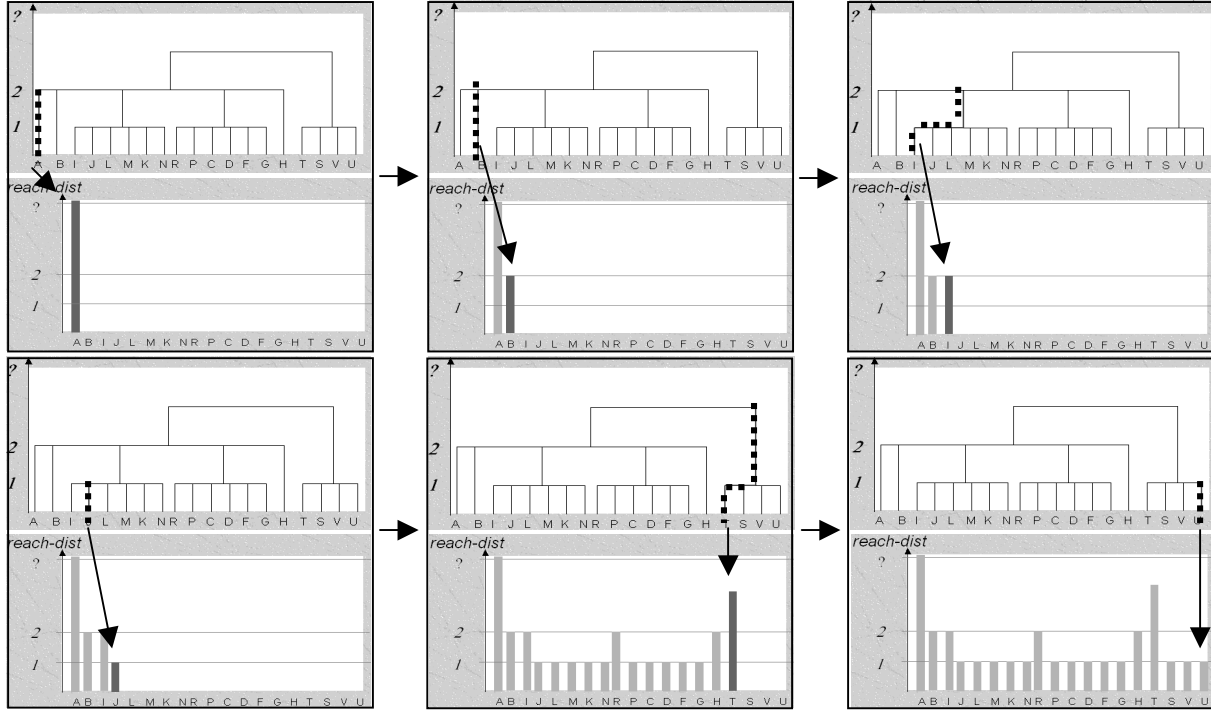


Figure 4: Illustration of the conversion of a dendrogram into a reachability plot

in a cluster, but its reachability value is still high. This is correct since in reachability plots the value for “border” objects of a cluster indicates the distance of that cluster to the previous points. For the next point J, the reachability value will be set equal to 1, since this is the height of the smallest common ancestor with previous point I. This is again the smallest distance between the current point J and the set of point to the left of J. In the same way we can derive the reachability values for the following points in the dendrogram, from point L to point H. When we consider the point T in the dendrogram we see that it is the first point of a cluster (containing T, S, V, and U) with an internal “merging distance” of 1. Again, its reachability distance is correctly set to 3, since the smallest common ancestor between T and H is the root of the dendrogram at height 3 (– also the smallest distance between T and the points to the left of T).

A reachability plot created by OPTICS (using $MinPts = 2$ and $Eps = \infty$) contains more information than an arbitrary Single-Link dendrogram. Our transformation will generate a reachability plot in a strict sense only when applied to the dendrogram that satisfies the additional condition mentioned above. However, when converting an arbitrary dendrogram we will still get a bar plot that reflects the hierarchical clustering structure well enough for cluster analysis. Clusters will still form “dents” in the plot, only the order of the points may be different and the border points may not be correct, since this information was not represented in the dendrogram. For instance, if in our previous example the order of the points {T, S, V, U} would be different in the dendrogram the shape of the resulting reachability plot would not change. Instead of T, one of the other points S, U, V in the cluster would get a high reachability value, separating this group as a cluster from the rest of the points.

2.3 Converting reachability plots into dendrograms

To understand how to transform a reachability plot into a dendrogram consider the properties that hold for a reachability plot generated by OPTICS (using $MinPts = 2$ and $Eps = \infty$). Assume that the order generated by OPTICS is $(p_1, p_2, \dots, p_{i-1}, p_i, p_{i+1}, \dots, p_n)$. The reachability value d_i of point p_i is the shortest distance between p_i and the set of points $SI = \{p_1, p_2, \dots, p_{i-1}\}$ to the left of p . Furthermore, p_i is the point that is closest to the set SI among all points in the set of points $S2 = \{p_i, p_{i+1}, \dots, p_n\}$ to the right of p (this is guaranteed by the way OPTICS algorithm select the next point after processing p_{i-1}).

Now consider applying the Single-Link method to this data set. Obviously, at some point (possibly

after some iterations) a node $N2$ containing p_i (and possibly other points from $S2$) will be merged with a node $N1$ containing only points from $S1$. The distance used for the merging (the height of the resulting node in the dendrogram) will be the reachability value d_i of p_i . The node $N1$ (containing points to the left of p_i) will always include p_{i-1} (the left neighbor of p_i). The reason is that in fact a stronger condition on the distance between p_i and the points to its left holds in a dendrogram: there is a point p_{i-m} to the left of p_i with the largest index such that its reachability value is larger than d_i , and d_i is the smallest distance to the set of points $SI^* = \{p_{i-m}, \dots, p_{i-1}\}$. When the node $N2$ containing p_i will be merged in the Single-Link method at the distance d_i , it will be merged with the node $N1$ containing its nearest neighbor in SI^* . But, since the reachability values of the points in SI^* are smaller than d_i , – except for d_{i-m} , the points in SI^* will have been merged already in previous steps of the Single-Link method. Hence they will be contained in $N1$, including p_{i-1} . Based on this observation, we can transform a reachability plot into a dendrogram using the following method:

```

// RPlot=[ $p_1, \dots, p_n$ ] is the input reachability plot; each element RPlot[i]= $p_i$  has the following attributes:
// RPlot[i].r_dist = reachability distance of  $p_i$ 
// RPlot[i].current_node = the current node in the dendrogram where  $p_i$  belongs to
// PointList = [ $q_1, \dots, q_n$ ] is a sequence of references to the points in ascending order of
// the corresponding reachability values – excluding  $p_1$  since  $p_1.r\_dist = \infty$  in any reachability plot.

for (i = 1; i < n; i++)
    create a leaf node N_i for RPlot[i];
    RPlot[i].current_node = N_i; //assign the new leaf node to be the points initial current_node

for (i = 1; i < n; i++)
    p = PointList[i]; // pick the point with the next smallest r_dist
    q = left neighbor of p in Rplot;
    create a new node N in the dendrogram;
    N.height = p.r_dist;
    N.left_child = q.current_node;
    N.right_child = p.current_node;
    p.current_node = q.current_node = N;

```

In Figure 5 we use the simple reachability plot from the previous section to illustrate the algorithm. In the first step, the algorithm will connect a point with the smallest reachability value, here J, with its left neighbor, here I. It will create a new node $N1$ with its height equal to the reachability value of J. The current_node for both I and J will become the new node $N1$. In the second step, it selects the next point from the sorted list of points (i.e., with the next smallest reachability value), say L. Again, it will create a new node $N2$ and connect the current_node of L (which is the leaf containing L) with the current_node of L's left neighbor. The point is J, and J's current_node is the node N created in the previous step. After processing the points in our example with a reachability value of 1, the intermediate dendrogram looks like the one depicted. The points I, J, L, M, K, and N at this stage will have the same current node, e.g. N_r , and similar for the other points. The next smallest reachability value in the next step will be 2. Assume the next point found in PointList (with a reachability value of 2) is point B. B is merged with B's left neighbor A. The current nodes for A and B are still the leaves that contain them. A new node N_u is created with the reachability value of B as its height. The current node for both A and B is set to N_u . In the next step, if I is selected, the current_node of I ($=N_r$) will be merged with current_node of I's left neighbor B ($=N_u$). After that, node N_s will be merged with this new node in the next step and so on until all points in our example with a reachability value of 2 are processed; the intermediate result looks like depicted above. The only point remaining in PointList at this point is T with a reachability value of 3. For point T the current_node is N_r , the current node of T's left neighbor H is N_v . Hence N_r and N_v are merged into a new node N_w at height 3 (the reachability value of T).

Figure 6 demonstrates our method using the data set shown in Figure 3, where we have a larger variation of distances between points than in the example we used for the explanation of the method (but this also make the dendrograms less readable). Obviously, if we have a reachability plot for parameters

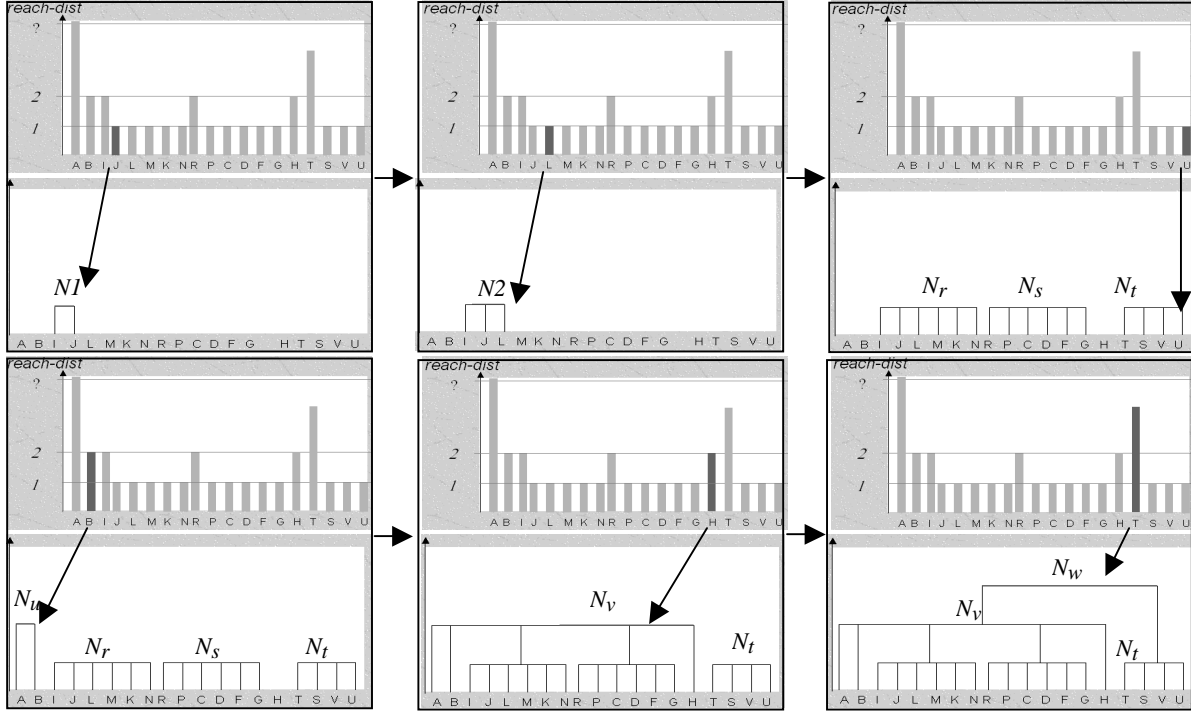


Figure 5: Illustration of the conversion of a reachability plot into a dendrogram

other than $MinPts = 2$ and $Eps = \infty$, our method generates a dendrogram where the merging distances are based on the minimum *reachability* distances between groups of points instead of point-wise distances.

3. Extracting Clusters from Dendrograms and Reachability Plots

As argued in section 1, it is desirable to have a simplified representation of a hierarchical clustering structure that contains only the significant clusters, so that hierarchical clustering can be used as an automatic pre-processing step in a larger KDD process.

We base our method on a reachability plot representation of a hierarchical clustering structure. To apply this method to dendrograms we have to transform the dendrogram into a reachability plot first using the method introduced in the previous section. Figure 7(a) illustrates our goal of simplifying a hierarchical clustering structure. Given a hierarchical representation such as the depicted reachability plot, the goal of our algorithm is to produce a simple tree structure, whereby the root represents all of the points in the dataset, and its children represent the biggest clusters in the dataset, possibly containing sub-clusters. As we proceed deeper into the tree the nodes represent smaller and denser clusters. The tree should only contain nodes for significant clusters as seen by a user who would identify clusters as “dents” or “valleys” in a reachability plot. The dents are roughly speaking regions in the reachability plot, having relatively low reachability values, and being separated by higher reachability values. Our automatic cluster extraction method first tries to find the reachability values that separate clusters. Those high reachability

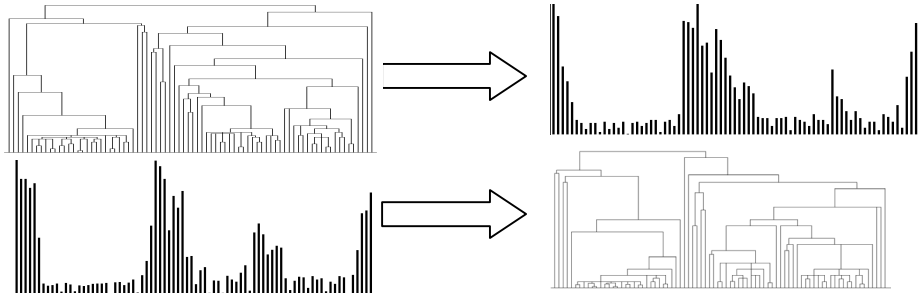


Figure 6: Transformation between Dendrogram and Reachability plot for the data set in Figure 3

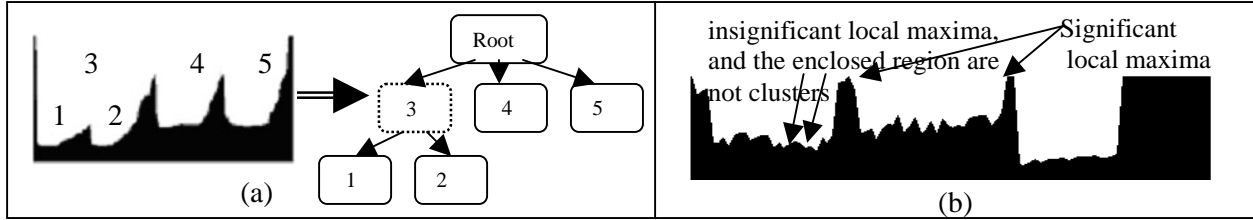


Figure 7: (a) Desired transformation; (b) illustration of insignificant local maxima and clusters

values are local maxima in the reachability plot. The reachability values of the immediate neighbors to the left and right side of such a point p are smaller than the reachability value of p , or do not exist (if p is the leftmost or rightmost point in the reachability plot). However, not all local maxima points are separating clusters, and not all regions enclosed by local maxima are prominent clusters. A local maximum may not differ significantly in height from the surrounding values, and local maxima that are located very close to each other create regions that are not large enough to be considered important clusters. This is especially true when using a small value for MinPts in OPTICS or a reachability plot converted from a Single-Link dendrogram (see Figure 7(b) for an illustration).

To ignore regions that are too small in the reachability plot we assume (as is typically done in other methods as well) a minimum cluster size. In our examples, we set the minimum cluster size to 0.5% of the whole data set, which still allows us to find even very small clusters. The main purpose is the elimination of “noisy” regions that consists of many insignificant local maxima, which are close to each other.

The significance of a separation between regions is determined by the ratio between the height of a local maximum p and the height of the region to the left and to the right of p . The ratio for a significant separation is set to 0.75, i. e. for a local maximum p to be considered a cluster separation, the average reachability value to the left and to the right has to be at least 0.75 times lower than the reachability value of p . This value represents approximately the ratio that a user perceives as the minimum required difference in height to indicate a relevant cluster separation. We experimented with different ratios and in fact, any value in the range 0.7-0.8 always gives good results, i.e., results, which automatically determine the clusters that a user would select manually (evidence is given at the end of this section in the experimental evaluation of our method).

In the first step of our algorithm, we collect all points p whose reachability value is a local maximum in a neighborhood that encloses minimum-cluster-size many point to the left of p and to the right of p (if this is possible – points close to the border of the reachability plot may have less points on one side). The local maxima points are sorted in descending order of their reachability value, and then processed one by one to construct the final cluster tree. The pseudo-code for a recursive construction of a cluster tree is shown in Figure 8. Intuitively, the procedure always removes the next largest local maximum s from the maxima list (until the list is empty), determines whether this split point justifies new nodes in the cluster tree – and if yes, where those nodes have to be added. This is done in several steps: First, two new nodes are created by distributing the points in the current node into two new nodes according to the selected split point s : All points that are in the reachability plot to the left of s are inserted into the first node; the second node contains all the points that are right of s in the reachability plot (for the recursion, also the list of all local maxima points for the current node are distributed accordingly into two new lists). The two new nodes are possible children of the current node or the parent node, depending on the subsequent tests.

We first check that the difference in reachability between the newly created nodes and the current local maximum indicates a relevant separation (as discussed above). If the test is not successful, the current local maximum is ignored and the algorithm continues with the next largest local maximum. If the difference is large enough, we have to check whether the newly created nodes contain enough points to be considered a cluster. If one of these nodes does not have the minimum cluster size it is deleted. If the nodes are large enough, they will be added to the tree. There are two possibilities to add the new nodes to the tree: as children of the current node, or as children of the parent node, replacing the current node. If we would always add the nodes as children of the current node, we would create a binary tree. This may not result in the desired representation as illustrated in Figure 9. To determine whether two reachability

```

cluster_tree(N, parent_of_N, L)
// N is a node; the root of the tree in the first call
// parent_of_N is the parent node of N; nil if N is the root of the tree
// L is the list of local maxima points, sorted in descending order of reachability
if (L is empty) return; // parent_of_N is a leaf

// take the next largest local maximum point as a possible separation between clusters
N.split_point = s = L.first_element();
L = L - {s};

// create two new nodes and add them to a list of nodes
N1.points = {p ∈ N.points | p is left of s in the reachability plot};
N2.points = {p ∈ N.points | p is right of s in the reachability plot};
L1 = {p ∈ L | p lies to the left of s in the reachability plot};
L2 = {p ∈ L | p lies to the right of s in the reachability plot};
Nodelist NL = [(N1, L1), (N2, L2)];

// test whether the resulting split would be significant:
if ((average reachability value in any node in NL) / s.r_dist > 0.75)
  // if split point s is not significant, ignore s and continue
  cluster_tree(N, parent_of_N, L); //
else //remove clusters that are too small
  if (N1.NoOfpoints) < min_cluster_size) NL.remove(N1, L1);
  if (N2.NoOfpoints) < min_cluster_size) NL.remove(N2, L2);
  if (NL is empty) return; // we are done and parent_of_N will be a leaf

  // check if the nodes can be moved up one level
  if (s.r_dist and parent_of_N.split_point.r_dist are approximately the same)
    let parent_of_N point to all nodes in NL instead of N;
  else
    let N point to all nodes in NL;
  // call recursively for each new node
  for each element (N_i, L_i) in NL
    cluster_tree(N_i, parent_of_N, L_i)

```

Figure 8: Algorithm for constructing a cluster tree from a reachability plot

values are approximately the same, we use a simple but effective test: we assume that the reachability value of the parent node (after scaling) is the true mean of a normed normal distribution ($\sigma=1$). If the (scaled) reachability value of the current node is within one standard deviation of the parent's value, we consider the two values as similar enough to attach the newly created nodes to the parent node of the current node instead of the current node itself. This heuristic gives consistently good results for all classes of reachability plots that we tested.

We performed a large number experiments to evaluate our new method, but due to space limitations, we can only show a limited number of representative results in Figure 10. For comparison, we also show the results of the ξ -cluster method that was proposed in the original OPTICS publication [ABKS 99]. We also show the data sets that were clustered using OPTICS. This illustration should only help to understand the extracted clusters; the compared methods are, strictly speaking, not dependent the actual data set, but only on the properties of the reachability plot from which (potential) clusters are extracted.

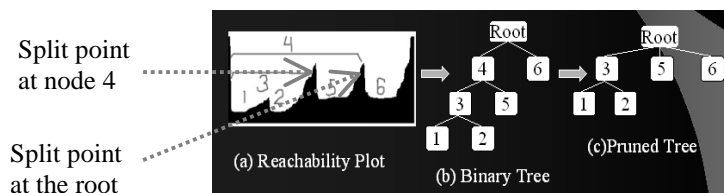


Figure 9: Illustration of the “tree pruning” step

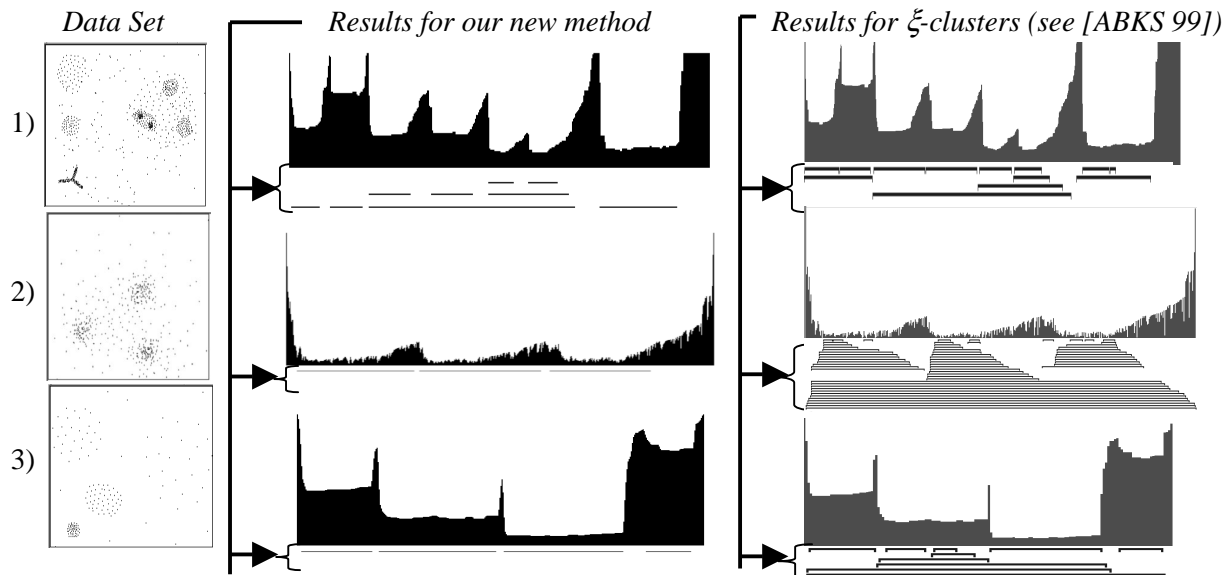


Figure 10: Performance comparison of our new automatic cluster extraction method with the extraction of ξ -clusters presented in [ABKS 99], using for reachability plots with different characteristics

The results for our new method are shown in the middle column, and the results for ξ -clusters are given in the left column. For both methods, the extracted cluster trees are depicted underneath each reachability plot. The nodes are represented as a horizontal bars that extend along the points, which are included in this node, and which would be used to generate cluster descriptions such center, diameter, etc for each node. This tree itself is drawn in reverse order, i.e., the leaves are closest to the reachability plot. The root of the tree is always omitted since it always contains all point and therefore would be represented by a bar at the bottom of each plot, extending over the whole width of the plot.

The experimental results we show are for three different reachability plots, each representing an important class of reachability plots having similar properties. The essential characteristics: of the three reachability plots are the following: (a) relatively smooth plot with nested clusters, (b) noisy plot and fuzzy cluster separation, (c) smooth plot with clusters having largely differing densities (c). Examples (a) and (c) are also examples where the method of a simple cut through the dendrogram will not reveal all existing clusters at the same time, because such a cut does not exist.

The results clearly show the superiority of our method: in all three different cases, our new method generates a cluster tree, which is very similar to what a user would extract as clusters from the given reachability plots. The ξ -cluster method, on the other hand, always constructs rather convoluted and much less useful cluster trees – using its default parameter value $\xi=0.03$ (for each of the examples there are some ξ -values, which give slightly better results than those shown above, but those best ξ -values are very

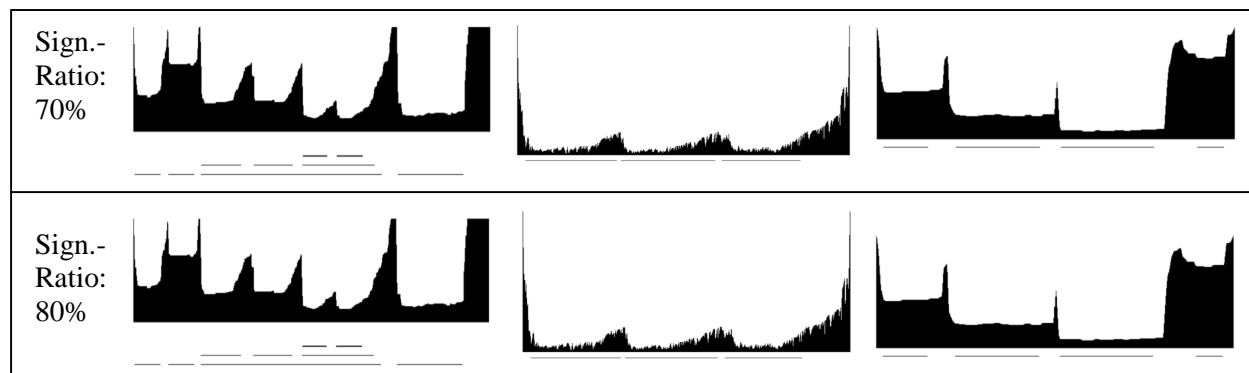


Figure 11: Results for different reachability ratios to determine significant cluster separation in our method

different for each example, and even then, none of them matched the quality of the results for our proposed new method). Note that we did not have to change any parameter values to extract the clusters from largely differing reachability plot (whereas finding the best ξ - values for the each example would involve an extensive trial and error effort). The internal parameter value of a 75% ratio for a significant cluster separation works well in all our experiments, and as noted above. Our method is in fact very robust with respect to this ratio, as shown in Figure 11 where we present the results for the same data as above, but for different ratios (70%, and 80%). In fact, we only suggest in extreme cases that a user might consider changing this value to a higher or lower value. The rule to set the value for this ratio is: the lower the value, the less deep a “valley” can be (relative to its surrounding) to be considered as a significant cluster region in a reachability plot.

4. Conclusions

In this paper we first showed that dendrograms and reachability plots contain essentially the same information and introduced methods to convert them into each other. We then proposed and evaluated a new technique that automatically determines the significant clusters found by hierarchical clustering methods. This method is based on a reachability plot representation, but also applicable to dendrograms via the introduced transformation. The major purpose of these methods is to allow hierarchical clustering as an automatic pre-processing step in a data mining process that operates on detected clusters in later steps. This was not possible before, since hierarchical clustering algorithms do not actually determine clusters, but compute only a hierarchical representation of a data set. Without our methods, a user typically has to manually select clusters from the representation based on visual inspection. Our algorithm generates a cluster tree that contains only the significant clusters in a given representation as nodes.

There are some directions for future work: currently, a cluster from a hierarchical clustering result is represented simply as the set of points it contains. For an automatic comparison of different clustering structures, we will investigate the generation of more compact cluster descriptions. Using those descriptions, we will also investigate methods to compare different cluster trees generated by our method. This is an important prerequisite for data mining applications like monitoring the clustering structure and its changes over time in a dynamic data set.

References

- [ABKS 99] Ankerst M., Breunig M. M., Kriegel H.-P., Sander J.: “OPTICS: Ordering Points To Identify the Clustering Structure”, Proc. ACM SIGMOD, Philadelphia, PA, 1999, pp 49-60.
- [EKSX 96] Ester M., Kriegel H.-P., Sander J., Xu X.: “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”, Proc. KDD’96, Portland, OR, 1996, pp. 226-231.
- [HK 98] Hinneburg A., Keim D.: “An Efficient Approach to Clustering in Large Multimedia Databases with Noise”, KDD’98, New York City, NY, 1998.
- [JD 88] Jain A. K., Dubes R. C.: “Algorithms for Clustering Data,” Prentice-Hall, Inc., 1988.
- [KN 96] Knorr E. M., Ng R.T.: “Finding Aggregate Proximity Relationships and Commonalities in Spatial Data Mining,” IEEE Trans. on Knowledge and Data Engineering, Vol. 8, No. 6, December 1996, pp. 884-897.
- [KR 90] Kaufman L., Rousseeuw P. J.: “Finding Groups in Data: An Introduction to Cluster Analysis”, John Wiley & Sons, 1990.
- [Mac 67] MacQueen J.: “Some Methods for Classification and Analysis of Multivariate Observations”, Proc. 5th Berkeley Symp. Math. Statist. Prob., 1967, Vol. 1, pp. 281-297.
- [NH 94] Ng R. T., Han J.: “Efficient and Effective Clustering Methods for Spatial Data Mining”, Proc. VLDB’94, Santiago, Chile, Morgan Kaufmann Publishers, San Francisco, CA, 1994, pp. 144-155.
- [SCZ 98] Sheikholeslami G., Chatterjee S., Zhang A.: “WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases”, Proc. VLDB’98, New York, NY, 1998, pp. 428 - 439.
- [Sib 73] Sibson R.: “SLINK: an optimally efficient algorithm for the single-link cluster method”, the Computer Journal Vol. 16, No. 1, 1973, pp. 30-34.