Proc. 4th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD 2000), Lyon, France.

Fast Hierarchical Clustering Based on Compressed Data and OPTICS

Markus M. Breunig, Hans-Peter Kriegel, Jörg Sander
Institute for Computer Science, University of Munich
Oettingenstr. 67, D-80538 Munich, Germany
{breunig | kriegel | sander}@dbs.informatik.uni-muenchen.de
phone: +49-89-2178-2225
fax: +49-89-2178-2192

Abstract: One way to scale up clustering algorithms is to squash the data by some intelligent compression technique and cluster only the compressed data records. Such compressed data records can e.g. be produced by the BIRCH algorithm. Typically they consist of the sufficient statistics of the form (N, X, X^2) where N is the number of points, X is the (vector-)sum, and X^2 is the square sum of the points. They can be used directly to speed up k-means type of clustering algorithms, but it is not obvious how to use them in a hierarchical clustering algorithm. Applying a hierarchical clustering algorithm e.g. to the centers of compressed subclusters produces a very weak result. The reason is that hierarchical clustering algorithms are based on the distances between data points and that the interpretaion of the result relies heavily on a correct graphical representation of these distances. In this paper, we introduce a method by which the sufficient statistics (N, X, X^2) of subclusters can be utilized in the hierarchical clustering method OPTICS. We show how to generate appropriate distance information about compressed data points, and how to adapt the graphical representation of the clustering result. A performance evaluation using OPTICS in combination with BIRCH demonstrates that our approach is extremely efficient (speed-up factors up to 1700) and produces high quality results.

1 Introduction

Knowledge discovery in databases (KDD) is known as 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 which should be applicable in this process is cluster analysis. Therefore, improving clustering algorithms with respect to efficiency and the quality of the results has received a lot of attention in the last few years in the research community..

The goal of a clustering algorithm is to group objects into meaningful subclasses. Applications of clustering are, e.g., the creation of thematic maps in geographic information systems, the detection of clusters of objects in geographic information systems and to explain them by other objects in their neighborhood, or the clustering of a Weblog database to discover groups of similar access patterns corresponding to different user profiles.

There are different types of clustering algorithms suitable for different types of applications. The most commonly known distinction is between *partitioning* and *hierar-*

chical clustering algorithms (see e.g. [8]). Partitioning algorithms construct a partition of a database *D* of *n* objects into a set of *k* clusters. Typical examples are the *k*-means [9] and the *k*-medoids [8] algorithms. Most hierarchical clustering algorithms such as the single link method [10] and OPTICS [1] do not construct a clustering of the database explicitly. Instead, these methods compute a representation of the data set (single link: dendrogram, OPTICS: reachability plot) which reflects its clustering structure. It depends on the application context whether or not the data set is decomposed into definite clusters. Heuristics to find a decomposition of a hierarchical representation of the clustering structure exist, but typically the results are analyzed interactively by a user.

Clustering algorithms, in general, do not scale well with the size and/or dimension of the data set. One way to overcome this problem is to use *random sampling* in combination with a clustering algorithm (see e.g. [6]). The idea is to apply a clustering algorithm only to a randomly chosen subset of the whole database. The clustering for the whole database is then "inferred" from the clustering of the subset. This usually leads to a significant inaccuracy of the clustering, introduced by sampling variance.

Another approach is to use more intelligent *data compression* techniques to squash the data into a manageable amount, and cluster only the compressed data records. This approach has been pioneered in the BIRCH algorithm [11]. More general compression techniques to support clustering algorithms are e.g. presented in [2], [4]. Compressed data items can be produced by any of the above compression methods in linear time. They are, however, tailored to *k*-means type of clustering algorithms. On the other hand, it is not obvious how to use compressed data items in a hierarchical clustering algorithm without unacceptably deteriorating the quality of the result.

In this paper, we show how compressed data items of the above form can be used in the hierarchical clustering method OPTICS. For this purpose, we adapt the OPTICS algorithm so that it can profit maximally from the information generated in the compression step. A detailed qualitative and performance analysis for real and synthetic data sets is conducted using the BIRCH method for the compression step.

The rest of the paper is organized as follows. In section 2, we give the problem statement in more detail. Section 3 elaborates on how OPTICS is extended to make optimal use of the information generated by a common data compression step. In section 4, we present basic experimental results and discuss the tradeoff between quality of the results and improvement of the runtime for OPTICS when using compressed data. Finally, section 7 concludes the paper.

2 Problem Statement

Specialized data compression methods have been developed to scale up clustering algorithms. The sufficient statistics intended to support clustering algorithms are basically the same for all these compression methods. As an example, we give a short description of the method BIRCH and only discuss the major differences and the common features for the other methods in this section. Then, we will show why a hierarchical clustering algorithm cannot directly benefit from the compressed data while *k*-means type of clustering algorithms are well supported.

The clustering method *BIRCH* [11] uses a highly specialized tree-structure for the purpose of clustering very large sets of *d*-dimensional vectors. It incrementally computes compact descriptions of subclusters, called Clustering Features.

Definition 1: (Clustering Feature, *CF*)

Given a set of n d-dimensional data points $\{X_i\}$, $1 \le i \le n$. The *Clustering Feature* (*CF*) for the set $\{X_i\}$ is defined as the triple $CF = (n, LS, ss)$, where

$LS = \sum_{i=1\ldots n} X_i$ is the linear sum and $ss = \sum_{i=1\ldots n} X_i^2$ the square sum of the points.

The *CF*-values are sufficient to compute information about subclusters like centroid, radius and diameter. They satisfy an important additivity condition, i.e. if $CF_1 = (n_1, LS_1, ss_1)$ and $CF_2 = (n_2, LS_2, ss_2)$ are the clustering features for sets of points S_1 and S_2 respectively, then $CF_1 + CF_2 = (n_1 + n_2, LS_1 + LS_2, ss_1 + ss_2)$ is the clustering feature for the set $S_1 \cup S_2$.

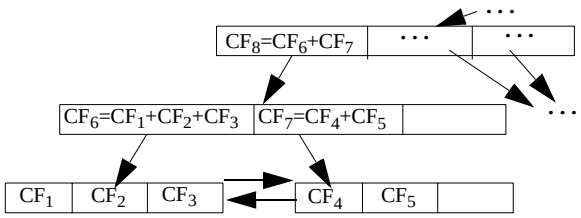


Fig. 1. CF-tree structure

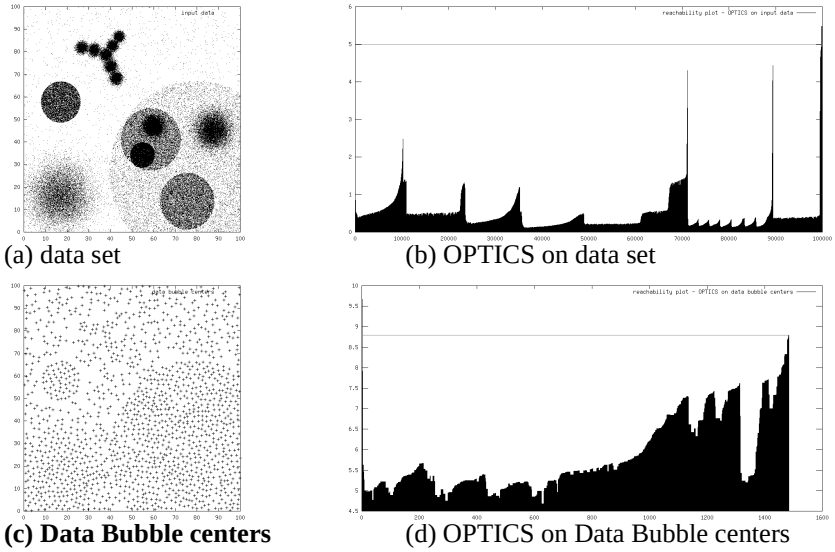
The CFs are organized in a balanced tree with branching factor B and a threshold T (see figure 1). A non-leaf node represents a subcluster consisting of all the subclusters represented by its entries. A leaf node has to contain at most L entries and the diameter of each entry in a leaf node has to be less than T.

BIRCH performs a sequential scan over all data points and builds a *CF*-tree similar to the construction of B^+-trees. A point is inserted by inserting the corresponding *CF*-value into the closest leaf. If an entry in the leaf can absorb the new point without violating the threshold condition, its *CF* is updated. Otherwise, a new entry is created in the leaf node, and, if the leaf node then contains more than L entries, it and maybe its ancestors are split. A clustering algorithm can then be applied to the entries in the leaf nodes of the *CF*-tree.

Bradley et al. [2] propose another compression technique for scaling up clustering algorithms. Their method produces basically the same type of compressed data items as BIRCH, i.e. triples of the form (n, LS, ss) as defined above. The method is, however, more specialized to k-means type of clustering algorithms than BIRCH in the sense that the authors distinguish different sets of data items. A very general framework for compressing data has been introduced recently by DuMouchel et al. [4]. Their technique is intended to scale up a large collection of data mining methods.

The application of k-means type clustering algorithms to compressed data items is rather straightforward. The k-means clustering algorithm represents clusters by the mean of the points contained in that cluster. It starts with an assignment of data points to k initial cluster centers, resulting in k clusters. It then iteratively performs the following steps while the cluster centers change: 1) Compute the mean for each cluster. 2) Re-assign each data point to the closest of the new cluster centers. When using CFs, the algorithm just has to be extended so that it treats the triplets (n, LS, ss) as data points LS/n with a weight of n when computing cluster means.

When we want to apply a hierarchical clustering algorithm to compressed data items, however, it is not clear whether we can follow a similar approach, i.e. treat clustering features as data points LS/n. Since hierarchical clustering algorithms do not compute any cluster centers but use only distances between points and between clusters, the



(a) data set (b) OPTICS on data set

(c) Data Bubble centers (d) OPTICS on Data Bubble centers

Fig. 2. 2-*d* example data set and its reachability plot

main problem with this approach is that we need a new concept of how to utilize the weight *n* of a clustering feature in a hierarchical clustering algorithm. Applying hierarchical clustering only to the set of centers *LS*/*n* of compressed data items (*n*, *LS*, *ss*) will in general produce only very inferior results.

Figure 2 (a) shows the 2-dimensional example used throughout the paper. The data set consists of 100,000 points grouped into several nested clusters of different densities and distributions (uniform and gaussian). Figure 2 (b) is the "reachability plot" for this data set produced by OPTICS. The "dents" in the plot represent the clusters, clearly showing the hierarchical structure. For a detailed explanation of reachability plots and their interpretation see [1]. Figure 2 (c) shows the distribution of the 1,484 clustering feature centers produced by BIRCH and (d) the corresponding reachability plot.

The failure observed in the example results from the following two problems when applying a hierarchical clustering algorithm to compressed data items:

- Hierarchical clustering algorithms need a notion of distance between data points which is obviously not represented well by only the distance between cluster feature centers. If we cannot infer some information about the distances between the points compressed into a single clustering feature, a hierarchical clustering algorithm cannot determine the correct clustering structure of the data set.
- Utilization of the output of a hierarchical clustering algorithm relies heavily on the graphical representation of the result. If we cannot integrate an appropriate representation of the number of points compressed into a single clustering feature, we loose much of the information about the clustering structure of a data set.

3 Extending OPTICS to Process Compressed Input Data

In this section we review the hierarchical clustering method OPTICS [1], and define the concept of *Data Bubbles* which contain information about compressed input data. We then extend OPTICS to work on Data Bubbles instead of data points.

3.1 Review of OPTICS

First, we define the basic concepts of neighborhood and nearest neighbors.

Definition 2: (ε-neighborhood and *k*-distance of an object *P*)
Let *P* be an object from a database *D*, let ε be a distance value, let *k* be a natural number and let *d* be a distance metric on *D*. Then:

- the ε-neighborhood $N_\varepsilon(P)$ is a set of objects *X* in *D* with $d(P,X) \le \varepsilon$: $N_\varepsilon(P) = \{ X \in D \mid d(P,X) \le \varepsilon \}$,
- the *k*-distance of *P*, *k-dist*(*P*), is the distance $d(P, O)$ between *P* and an object $O \in D$ such that at least for *k* objects $O' \in D$ it holds that $d(P, O') \le d(P, O)$, and for at most *k*-1 objects $O' \in D$ it holds that $d(P, O') < d(P, O)$. Note that *k*-dist(*P*) is unique, although the object *O* which is called 'the' *k*-nearest neighbor of *P* may not be unique. When clear from the context, we write $N_k(P)$ as a shorthand for $N_{k\text{-}dist(P)}(P)$, i.e. $N_k(P) = \{ X \in D \mid d(P, X) \le k\text{-}dist(P)\}$.

The objects in the set $N_k(P)$ are called the "*k*-nearest-neighbors of *P*" (although there may be more than *k* objects contained in the set if the *k*-nearest neighbor of *P* is not unique).

In [5] a density-based notion of clusters is introduced. The basic idea is that for each object of a cluster, the ε-neighborhood has to contain at least a minimum number of objects. Such an object is a *core object*. Clusters are defined as maximal sets of density-connected objects. An object *P* is density-connected to *Q* if there exists an object *O* such that both *P* and *Q* are density-reachable from *O* (directly or transitively). *P* is directly density-reachable from *O* if $P \in N_\varepsilon(O)$ and *O* is a core object. Thus, a flat partitioning of a data set into a set of clusters is defined, using *global* density parameters. Very different local densities may be needed to reveal and describe clusters in different regions of the data space. In [1] the density-based clustering approach is extended to create an augmented *ordering* of the database representing its density-based clustering structure. This cluster-ordering contains information which is equivalent to the density-based clusterings corresponding to a broad range of parameter settings. This cluster-ordering of a data set is based on the following notions of "core-distance" and "(density-)reachability-distance".

Definition 3: (core-distance of an object *P*)
Let *P* be an object from a database *D*, let ε be a distance value and let *MinPts* be a natural number. Then, the *core-distance* of *P* is defined as

$$core\text{-}dist_{\varepsilon,MinPts}(P) = \begin{cases} \text{UNDEFINED, if } |N_\varepsilon(P)| < MinPts \\ MinPts\text{-}dist(P), \text{ otherwise} \end{cases}.$$

The core-distance of an object *P* is the smallest distance $\varepsilon' \le \varepsilon$ such that *P* is a core object with respect to ε' and *MinPts* - if such a distance exists, i.e. if there are at least *MinPts* objects within the ε-neighborhood of *P*. Otherwise, the core-distance is UNDEFINED.

Definition 4: (reachability-distance of an object *P* w.r.t. object *O*)

Let *P* and *O* be objects, $P \in N_\varepsilon(O)$, let ε be a distance value and let *MinPts* be a natural number. Then, the *reachability-distance* of *P* with respect to *O* is defined as

$$reach\text{-}dist_{\varepsilon,MinPts}(P, O) = \begin{cases} \text{UNDEFINED, if } |N_\varepsilon(O)| < MinPts \\ max(core\text{-}dist_{\varepsilon, MinPts}(O), d(O, P)), \text{otherwise} \end{cases}.$$

Intuitively, *reach-dist*(*P*,*O*) is the smallest distance such that *P* is directly density-reachable from *O* if *O* is a core object. Therefore *reach-dist*(*P*,*O*) cannot be smaller than *core-dist*(*O*) because for smaller distances no object is directly density-reachable from *O*. Otherwise, if *O* is not a core object, *reach-dist*(*P*,*O*) is UNDEFINED. Figure 3 illustrates these notions.

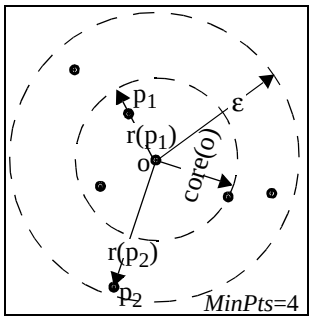


Fig. 3. core-dist(*O*), reach-dists r(P_1,*O*), r(P_2,*O*)

Using the core- and reachability-distances, OPTICS computes a "walk" through the data set, and assigns to each object *O* its core-distance and the smallest reachability-distance with respect to an object considered *before O* in the walk (see [1] for details). This walk satisfies the following condition: Whenever a set of objects *C* is a density-based cluster with respect to *MinPts* and a value ε' smaller than the value ε used in the OPTICS algorithm, then a permutation of *C* (possibly without a few border objects) is a subsequence in the walk. Therefore, the *reachability-plot*, which consists of the reachability values of all objects, plotted in the OPTICS ordering, yields an easy to understand visualization of the clustering structure of the data set. Roughly speaking, a low reachability-distance indicates an object within a cluster, and a high reachability-distance indicates a noise object or a jump from one cluster to another cluster.

3.2 From Sufficient Statistics to Data Bubbles

In section 2, we have seen different methods to compute subsets of the input data and sufficient statistics for these sets containing the number of points, the linear sum and the square sum. Based on these statistical information we define Data Bubbles as a convenient abstraction on which density-based hierarchical clustering can be done.

Definition 5: (Data Bubble)

Let $X=\{X_i\}$, $1 \le i \le n$ be a set of *n d*-dimensional data points.

Then, a *Data Bubble B* is defined as a triple $B = (n, M, e)$, where

$$M = \left(\sum_{i = 1 \ldots n} X_i \right) / n \text{ is the } \textit{center} \text{ of } X, \text{ and } e = \sqrt{\frac{\sum_{i = 1..n} \sum_{j = 1..n} (X_i - X_j)^2}{n \cdot (n-1)}}$$

is called the *extent* of *X* (i.e. the average pairwise distance between the points in *X*).

Data Bubbles are a compressed representation of the sets of points they describe. If the points are approximately uniformly distributed around the *center*, a sphere of radius *extent* around the *center* will contain most of the points described by the Data Bubble. The following lemma shows how to compute Data Bubbles from sufficient statistics.

Corollary 1:

Let $X=\{X_i\}$, $1 \le i \le n$ be a set of n d-dimensional data points. Let LS be the linear sum and ss the square sum of the points in X as defined in definition 1. Then, the Data Bubble B describing X is equal to $B = \left(n, \frac{LS}{n}, \sqrt{\frac{2 \cdot n \cdot ss - 2 \cdot LS^2}{n \cdot (n-1)}} \right)$.

3.3 Basing OPTICS on Data Bubbles by Modifying the core-dist and reach-dist

In any state, the OPTICS algorithm needs to know which data object is closest to the ones considered so far. To extend OPTICS to work on Data Bubbles, we therefore need a suitable measure for the distance between Data Bubbles. Given a distance between Data Bubbles, it is possible to extend the OPTICS algorithm by defining a suitable core- and reachability-distance for Data Bubbles.

If we assume that a Data Bubble B is a good description of its points, i.e. that all (or almost all) points it describes are inside a sphere of the extent of B around the center of B, we can compute the expected k-nearest neighbor distance of the points in B in the following way:

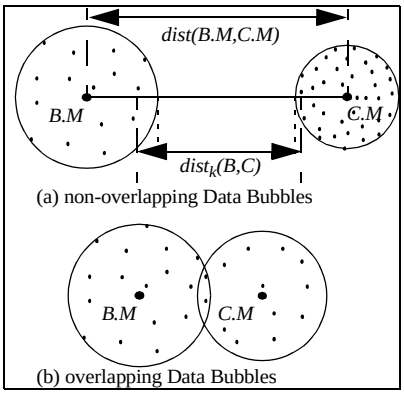


Fig. 4. distance of Data Bubbles

Lemma 1: (expected k-nearest neighbor distance inside a Data Bubble B)

Let $B = (n, M, e)$ be a Data Bubble of dimension d. If the n points described by B are uniformly distributed inside a sphere with center M and radius e, then the *expected k-nearest neighbor distance of B* is then equal to $nndist(k, B) = \left(\frac{k}{n}\right)^{1/d} \cdot e$.

Proof: The volume of a d-dimensional sphere of radius e is $V_S(e) = \frac{\sqrt{\pi}^d}{\Gamma\left(\frac{d}{2}+1\right)} \cdot e^d$

(where Γ is the Gamma-Function). If the n points are uniformly distributed in such a sphere, we expect exactly one point in the volume $V_S(e) / n$ and k points in the volume $k\ V_S(e) / n$, which is exactly the volume of a sphere of radius *nndist(k, B)*.■

Using the radius and the expected k-nearest neighbor distance, we can now define a distance measure between two Data Bubbles that is suitable for OPTICS.

Definition 6: (distance between two Data Bubbles)

Let $B=(n_1, M_1, e_1)$ and $C=(n_2, M_2, e_2)$ be two Data Bubbles and k a natural number. Then, the *k-distance* between B and C is defined as

$$dist_k(B, C) = \begin{cases} dist(M_1, M_2) - (e_1 + e_2) + nndist(k, B) + nndist(k, C) \\ \qquad \text{if } dist(M_1, M_2) - (e_1 + e_2) \ge 0 \\ max(nndist(k, B), nndist(k, C)) \qquad \text{otherwise} \end{cases}.$$

We have to distinguish two cases (c.f. figure 4). The distance between two non overlapping Data Bubbles is the distance of their centers, minus their radii plus their expected *k*-nearest neighbor distances. If the Data Bubbles overlap, their distance is the maximum of their expected *k*-nearest neighbor distances. Intuitively, this distance definition is intended to approximate the distance of the two closest points in the Data Bubbles. Using this distance, we can define the notion of a core-distance and a reachability-distance.

Definition 7: (core-distance of a Data Bubble *B*)
Let *B* be a Data Bubble, let ε be a distance value and let *MinPts* be a natural number. Then, the *core-distance* of *B* is defined as

$$core\text{-}dist_{\varepsilon,MinPts}(B) = \begin{cases} \text{UNDEFINED if } \left(\sum_{X=(n,M,e) \in N_\varepsilon(B)} n \right) < MinPts \\ dist(B,C) \quad \text{otherwise} \end{cases},$$

where C is the Data Bubble in $N_\varepsilon(B)$ with minimal *dist(B, C)* such that

$$\left(\sum_{X=(n,M,e) \in N_\varepsilon(B) \wedge dist(B,X) < dist(B,C)} n \right) \geq MinPts \text{ holds.}$$

Note that $core\text{-}dist_{\varepsilon,MinPts}(B=(n,M,e))=0$ if $n \geq MinPts$.

This definition is based on a similar notion as the core-distance for data points. For points, the core-distance is undefined if the number of points in the ε-neighborhood is smaller than *MinPts*. Analogously, the core-distance for Data Bubbles is undefined if the sum of the numbers of points represented by the Data Bubbles in the ε-neighborhood is smaller than *MinPts*. For points, the core-distance (if defined) is the distance to the *MinPts*-neighbor. For Data Bubbles, it is the distance to the Data Bubble containing the *MinPts*-neighbor.

Given the core-distance, the reachability-distance for Data Bubbles is defined in the same way as the reachability-distances on data points.

Definition 8: (reachability-distance of a Data Bubble *B* w.r.t. Data Bubble *C*)
Let *B* and *C* be Data Bubbles, $B \in N_\varepsilon(C)$, let ε be a distance value and let *MinPts* be a natural number. Then, the *reachability-distance* of *B* with respect to *C* is defined as

$$reach\text{-}dist_{\varepsilon,MinPts}(B,C) = \begin{cases} \text{UNDEFINED, if } \left(\sum_{X=(n,M,e) \in N_\varepsilon(B)} n \right) < MinPts \\ max(core\text{-}dist_{\varepsilon,MinPts}(C), dist(C,B)) \text{ otherwise} \end{cases}.$$

3.4 Modifying the Output

In order to handle the output correctly, we need to define a *virtual* reachability for the data points described by a Data Bubble $B=(n, M, e)$. If we assume that the points described by B are uniformly distributed in a sphere of radius e around the center M, and B describes at least *MinPts* points, we expect the reachability of most of these points to be close to the *MinPts*-nearest neighbor distance. If, on the other hand, B contains less than *MinPts* points, we expect the core-distance of any of these points to be close to the core-distance of B itself. We use this heuristics to define the virtual reachability as an approximation of the true, but unknown, reachability of the points described by B.

Definition 9: (virtual reachability of a Data Bubble B w.r.t Data Bubble C)
Let $B=(n, M, e)$ and C be Data Bubbles and *MinPts* a natural number. The *virtual reachability* of the N points described by B w.r.t. C is then defined as

$$\textit{virtual-reachability}(B, C) = \begin{cases} \textit{nndist}(MinPts, B) & \text{if } n \geq MinPts \\ \textit{reach-dist}(B, C) & \text{otherwise} \end{cases}.$$

As Data Bubbles can contain strongly varying numbers of data points, we need to weigh each Data Bubble by the number its data points, if we want to maintain the proportional sizes of the clusters in the reachability plot. The output of the original OPTICS algorithm is generated by appending the reachability of the next chosen data point to the output file. When outputting a Data Bubble B, we first append the reachability of B to the output file (marking the jump to B) followed by n-times the virtual reachability of B (marking the n points that B describes).

4 Experimental Evaluation

4.1 Efficiency

In figure 5 the speed-up factors of OPTICS on Data Bubbles over "pure" OPTICS for different dimensional data are shown for a constant compression to approx. 1000 Data Bubbles. The size of the input data sets ranges from 50,000 points to 400,000 points. All data sets contain 30% noise and 10 clusters of random locations and sizes. The speed-up ranges from a factor of about 50 for small data sets to more than 1,700 for large, high-dimensional data. This is the wall-clock speed-up, i.e. it includes both CPU and I/O-time.

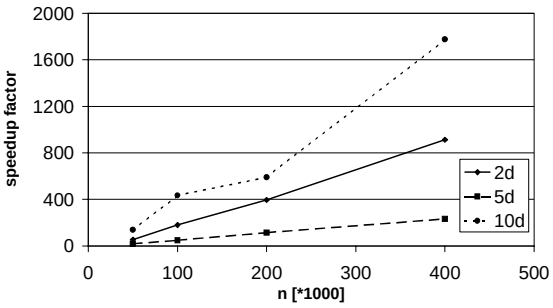


Fig. 5. Speed-up Factors

For "pure" OPTICS, an index structure [3] supported the range-queries (the index build time is not included; if it were, the speed-up would be even higher!), while OPTICS on the Data Bubbles used the sequential scan. We expect the scale-up to be approx. constant.

4.2 Understanding the Trade-off between Quality and Efficiency

We have seen so far that we gain large performance improvements by trading in relatively small amounts of quality. To get a better understanding of the trade-off involved see figure 6, which shows different compressed representations. For each of these, the runtime is depicted in the graph on the left side, followed by the number of Data Bubbles (DB) and the percentage of Data Bubbles relative to the number of original data points. Furthermore, each compression is visualized as a 2-dimensional plot, where the Data Bubbles B=(*n*, *M*, *e*) are shown as circles with center *M* and radius *e*. Finally, on the far right side we see the reachability plots generated from these Data Bubbles. Note that sometimes the order of the clusters in the reachability switches; this is a positive effect of the non-deterministic choices in the walk that OPTICS allows for.

The compression rate increases from top to bottom, i.e. the Data Bubbles get larger and the reachability plots become coarser. The top-most experiment, in which the data is compressed by 86.4%, shows no visible deterioration of quality at all. Compression rates of 94.6% and 98.5% show almost none. With a compression rate of 99.5%, the general clustering structure is still visible, with the one exception that the star-formed cluster consisting of 7 small, gaussian subclusters, starts to loose its internal structure. And even for the most extreme case on the bottom, in which the 100,000 data points are compressed into just 228 Data Bubbles, the general clustering structure is still preserved rather well.

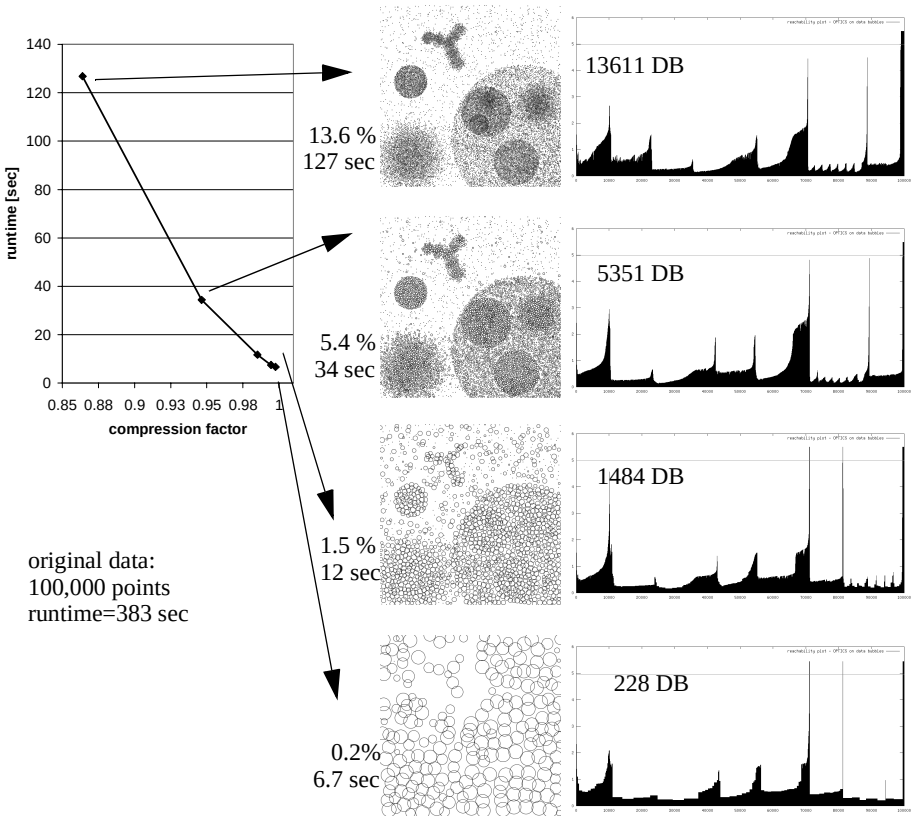


Fig. 6. Trade-off Quality vs. Performance

5 Conclusions

In this paper, we adapt the hierarchical clustering algorithm OPTICS to work on compressed data. First, we generate Data Bubbles consisting of essential information about compressed data points. Second, we adapt OPTICS to take Data Bubbles as input and make maximal use of the contained information. Third, we modify the graphical representation of the OPTICS result accordingly. Combining OPTICS with the compression technique BIRCH yields performance speed-up-factors between 50 and more than 1,700 for large, high-dimensional data sets, while at the same time maintaining the high quality of the clustering results.

Currently, we are evaluating the quality of our approach on high-dimensional data. In the future, we plan to compare data compression with sampling and to utilize other data compression methods to further improve the quality of the results, e.g. by using higher-order moments. We are also investigating how to extend the method to categorical data sets.

Acknowledgments

We thank the authors of BIRCH for making their code available to us and Ekkehard Krämer for his help with the implementation.

References

1. Ankerst M., Breunig M. M., Kriegel H.-P., Sander J.: "*OPTICS: Ordering Points To Identify the Clustering Structure*", Proc. ACM SIGMOD'99 Int. Conf. on Management of Data, Philadelphia, 1999, pp. 49-60.
2. Bradley P. S., Fayyad U., Reina C.: "*Scaling Clustering Algorithms to Large Databases*", Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining, New York, NY, AAAI Press, 1998, pp. 9-15.
3. Berchthold S., Keim D., Kriegel H.-P.: "*The X-Tree: An Index Structure for High-Dimensional Data*", 22nd Conf. on Very Large Data Bases, Bombay,India,1996, pp. 28-39.
4. DuMouchel W., Volinsky C., Johnson T., Cortez C., Pregibon D.: "*Sqashing Flat Files Flatter*", Proc. 5th Int. Conf. on Knowledge Discovery and Data Mining, San Diego, CA, AAAI Press, 1999, pp. 6-15.
5. Ester M., Kriegel H.-P., Sander J., Xu X.: *"A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise"*, Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press, 1996, pp. 226-231.
6. Ester M., Kriegel H.-P., Xu X.: "*Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification*", Proc. 4th Int. Symp. on Large Spatial Databases (SSD'95), Portland, ME, 1995, LNCS 591, Springer, 1995, pp. 67-82.
7. Fayyad U., Piatetsky-Shapiro G., Smyth P.*: "Knowledge Discovery and Data Mining: Towards a Unifying Framework*". Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, Portland, OR, 1996, pp. 82-88.
8. Kaufman L., Rousseeuw P. J.: *"Finding Groups in Data: An Introduction to Cluster Analysis"*, John Wiley & Sons, 1990.
9. MacQueen, J.: *"Some Methods for Classification and Analysis of Multivariate Observations"*, Proc. 5th Berkeley Symp. on Math. Statist. and Prob., Vol. 1, 1965, pp. 281-297.
10. Sibson R.: *"SLINK: An Optimally Efficient Algorithm for the Single-Link Cluster Method"*, The Computer Journal, Vol. 16, No. 1, 1973, pp. 30-34.
11. Zhang T., Ramakrishnan R., Linvy M.: *"BIRCH: An Efficient Data Clustering Method for Very Large Databases"*, Proc. ACM SIGMOD Int. Conf. on Management of Data, ACM Press, New York, 1996, pp. 103-114.