First publ. in: IEEE computer graphics and applications 19 (1999), 5, pp. 22-31

HD-Eye: Visual Mining of High-Dimensional Data

Alexander Hinneburg, Daniel A. Keim, and Markus Wawryniuk
University of Halle, Germany

An advanced clustering algorithm combined with new visualization methods interactively clusters data more effectively. Experiments show these techniques improve the data mining process.

Because of the fast rate of technological progress, the amount of data stored in databases increases rapidly. This proves true for traditional relational databases and complex 2D and 3D multimedia databases that store images, CAD drawings, geographic information, and molecular biology data. We can view relational databases as high-dimensional databases, since the attributes correspond to the dimensions of the data set. The same also holds true for multimedia data. For efficient retrieval, such data must usually be transformed into high-dimensional feature vectors such as color histograms,[1] shape descriptors,[2] Fourier vectors,[3] and text descriptors.[4] Many of the mentioned applications rely on very large databases consisting of millions of data objects with several tens to a few hundreds of dimensions.

Clustering in high-dimensional databases poses an important problem. However, we can apply a number of different clustering algorithms to high-dimensional data. The most prominent representatives include partitioning algorithms such as Clarans,[5] hierarchical clustering algorithms, and locality-based clustering algorithms such as (general) density-based spatial clustering of applications with noise, called (G)DBScan,[6] and distribution-based clustering of large spatial databases (DBCLASD).[7]

The basic idea of *partitioning algorithms* is to construct a partition of the database into k clusters represented by the gravity of the cluster (k-means) or by one representative object of the cluster (k-medoid). *Hierarchical clustering algorithms* decompose the database into several levels of partitionings usually represented by a dendrogram—a tree that splits the database recursively into smaller subsets. The third class of algorithms, *locality-based clustering algorithms*, usually group neighboring data elements into clusters based on local conditions.

Unfortunately, most approaches aren't designed to cluster high-dimensional data, thus the performance of existing algorithms degenerates rapidly with increasing dimensions. To improve efficiency, researchers have proposed optimized clustering techniques. Examples include grid-based clustering;[8] balanced iterative reducing and clustering using hierarchies (Birch),[9] which builds on the cluster-feature tree; statistical information grid (Sting), which uses a quadtree-like structure containing additional statistical information;[10] and density-based clustering (DenClue), which uses a regular grid to improve efficiency.[11] Unfortunately, the curse of dimensionality also severely affects the resulting clustering's effectiveness. A detailed comparison of the existing methods shows problems in effectiveness, especially in the presence of noise. Elsewhere,[12] we showed that the existing clustering methods either suffer from a marked breakdown in efficiency (which proves true for all index-based methods) or have notable effectiveness problems (which is basically true for all other methods).

Our idea, presented in this article, is to combine an advanced clustering algorithm with new visualization methods for a more effective interactive clustering of the data. The starting point is a novel and efficient clustering algorithm called OptiGrid,[12] which builds on a generalized multidimensional grid and uses contracting projections as well as complex hyperpolygonal objects as separators in the multidimensional space (see the section "Visual finding of projections and separators"). Choosing the contracting projections and specifying the separators for building the multidimensional grid, however, are two difficult problems that can't be done fully automatically.

Visualization technology can help in performing these tasks. A large number of potential visualization techniques can be used in data mining (for more information see the "Visual Data Mining Techniques" sidebar). Examples include geometric projection techniques such as prosection matrices and parallel coordinates, icon-based techniques, hierarchical techniques, graph-based techniques, pixel-oriented techniques, and combinations thereof. In general, the visualization techniques

Visual Data Mining Techniques

Geometric projection techniques aim to find interesting projections of multidimensional data sets. The class of geometric projection techniques includes techniques of exploratory statistics such as principal component analysis, factor analysis, and multidimensional scaling, many of which are subsumed under the term ***projection pursuit***.[1]

Geometric projection techniques also include the parallel coordinate visualization technique.[2] The basic idea is to map the ***k***-dimensional space onto the two display dimensions by using ***k*** equidistant axes parallel to one of the display axes. The axes correspond to the dimensions and are linearly scaled from the minimum to the maximum value of the corresponding dimension. Each data item is presented as a polygonal line, intersecting each of the axes at that point that corresponds to the value of the considered dimension.

Another class of techniques for visual data mining are the icon-based techniques (or iconic display techniques). The idea is to map each multidimensional data item to an icon. An example is the stick figure technique.[3] It maps two dimensions to the display dimensions and the remaining dimensions are mapped to the angles and/or limb lengths of the stick figure icon. This technique limits the number of dimensions that can be visualized. The shape-coding approach,[4] an icon-based technique, visualizes an arbitrary number of dimensions. The icon used in this approach maps each dimension to a small array of pixels and arranges the pixel arrays of each data item into a square or rectangle. The pixels corresponding to each of the dimensions are mapped to gray scale or color according to the dimension's data value. The small squares or rectangles corresponding to the data items are then arranged successively in a line-by-line fashion.

Pixel-oriented techniques aim to map each data value to a colored pixel and present the data values belonging to one attribute in separate windows. Since the pixel-oriented techniques use only one pixel per data value, the techniques allow a visualization of the largest amount of data, which is possible on current displays (up to about 1,000,000 data values). If one pixel represents each data value, the main question is how to arrange the pixels on the screen. The pixel-oriented techniques use different arrangements for different purposes. An overview of pixel-oriented techniques can be found elsewhere.[5,6]

The hierarchical techniques subdivide the ***k***-dimensional space and present the subspaces in a hierarchical fashion. The dimensional stacking technique,[7] for example, subdivides the ***k***-dimensional space into 2D-subspaces. Another example is the Cone Tree[8] technique. The basic idea of the graph-based techniques lies in effectively presenting a large graph using specific layout algorithms, query languages, and abstraction techniques. Examples appear elsewhere.[9,10]

In addition to the visualization technique, effective data exploration requires using some interaction and distortion techniques. The interaction techniques let the user directly interact with the visualization. Examples of interaction techniques include interactive mapping, projection,[11] filtering,[12] zooming,[13] and interactive linking and brushing.[14] Interaction techniques allow dynamic changes of the visualizations according to the exploration objectives, but they also make it possible to relate and

continued on p. 4

work in conjunction with some interaction techniques and also some distortion techniques.

Unfortunately, the existing techniques, don't effectively support the projection- and separator-finding process needed for an efficient clustering in high-dimensional space. Therefore, we developed a number of new visualization techniques that represent the important features of a large number of projections. These techniques help identify the most interesting projections and select the best separators. For 1D projections we developed a pixel-oriented representation of the point-density projections. Given a large number of interesting projections, the user may also employ a different visualization that represents the most important maxima of the point-density projections and their separation potential by small polygonal icons. The iconic representation reduces the information in the pixel-oriented visualization and allows a quick overview of the data.

For 2D projections, we use a similar iconic representation to help users find interesting projections and 2D pixel representations that also lets users directly specify complex hyperpolygonal separators within the visualization. Due to the large number of projections in the higher-dimensional case, only the iconic visualization can be used.

We integrated all visualization techniques by using a tree-like visualization of the projection and separator hierarchy. To show the effectiveness of our new visualization techniques, we used the system for clustering real data from molecular biology. The experiments show the effectiveness of our approach.

Overview of the HD-Eye approach

The HD-Eye approach builds on an advanced clustering algorithm called OptiGrid.[12]

Basic considerations

We start with a well-known and widely accepted definition of clustering. To do this, we need a point-density function, which can be determined based on kernel density estimation.[13,14]

> *Definition 1: Density function-kernel density estimation*. Let D denote a set of n d-dimensional points and h the smoothness level. Then, we can define the density function based on the kernel density estimator K as

$$\hat{f}^D(x) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

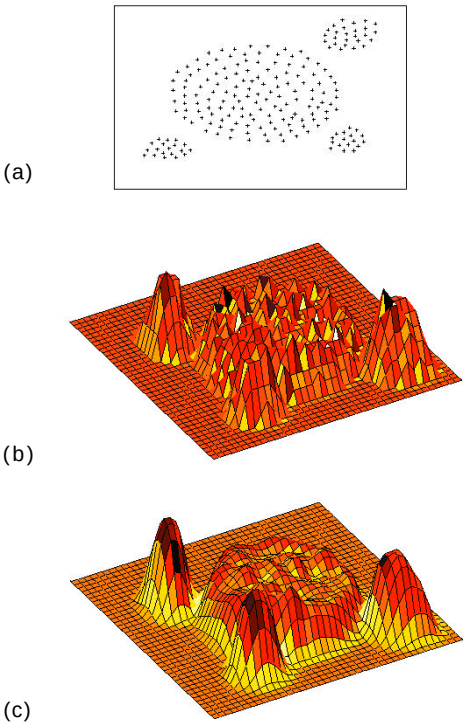
continued from p. 3

combine multiple independent visualizations. Note that connecting multiple visualizations by linking and brushing, for example, provides more information than considering the component visualizations independently.

The distortion techniques finally help in the interactive exploration process by providing means for focusing while preserving an overview of the data. Distortion techniques show portions of the data with a high level of detail while others are shown with a much lower level of detail. A number of simple and complex distortion techniques may be used for this purpose. Examples include fisheye views[15] and the hyperbolic tree.[16]

References

1. P.J. Huber, "Projection Pursuit," ***The Annals of Statistics***, Vol. 13, No. 2, 1985, pp. 435-474.
2. A. Inselberg and B. Dimsdale, "Parallel Coordinates: A Tool for Visualizing Multidimensional Geometry," ***Proc. Visualization 1990***, IEEE CS Press, Los Alamitos, Calif., 1990, pp. 361-370.
3. R.M. Pickett and G.G. Grinstein, "Iconographic Displays for Visualizing Multidimensional Data," ***Proc. IEEE Conf. on Systems, Man, and Cybernetics***, IEEE Press, Piscataway, N.J., 1988, pp. 514-519.
4. J. Beddow, "Shape Coding of Multidimensional Data on a Microcomputer Display," ***Proc. Visualization 1990***, IEEE CS Press, Los Alamitos, Calif., 1990, pp. 238-246.
5. D.A. Keim, ***Visual Support for Query Specification and Data Mining***, PhD thesis, University of Munich, July 1994 and Shaker-Publishing Company, Aachen, Germany, 1995.
6. D.A. Keim and H.-P. Kriegel, "VisDB: Database Exploration Using Multidimensional Visualization," ***IEEE Computer Graphics and Applications***, Vol. 14, No. 5, Sept. 1994, pp. 40-49.
7. J. LeBlanc, M.O. Ward, and N. Wittels, "Exploring ***N***-Dimensional Databases," ***Proc. Visualization 1990***, IEEE CS Press, Los Alamitos, Calif., 1990, pp. 230-237.
8. G. Robertson, S. Card, and J. Mackinlay, "Cone Trees: Animated 3D Visualizations of Hierarchical Information," ***Proc. ACM CHI Int'l Conf. on Human Factors in Computing***, ACM Press, New York, 1991, pp. 189-194.
9. S. Eick and G.J. Wills, "Navigating Large Networks with Hierarchies," ***Proc. Visualization 1993***, IEEE CS Press, Los Alamitos, Calif., 1993, pp. 204-210.
10. R.A. Becker, S.G. Eick, and G.J. Wills, "Visualizing Network Data," ***IEEE Trans. on Visualization and*** Computer ***Graphics***, Vol. 1, No. 1, 1995, pp. 16-28.
11. D. Asimov, "The Grand Tour: A Tool For Viewing Multidimensional Data," ***Society for Industrial and Applied Mathematics (SIAM) J. of Science and Statistical Computing,*** Vol. 6, No. 1, 1985, pp. 128-143.
12. C. Ahlberg and B. Shneiderman, "Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays," ***Proc. ACM CHI Int'l Conf. on Human Factors in Computing***, ACM Press, New York, 1994, pp. 313-317.
13. C. Ahlberg, C. Williamson, and B. Shneiderman, "Dynamic Queries for Information Exploration: An Implementation and Evaluation," ***Proc. ACM CHI Int'l Conf. on Human Factors in Computing***, ACM Press, New York, 1992, pp. 619-626.
14. A. Buja et al., "Interactive Data Visualization Using Focusing and Linking," ***Proc. Visualization 1991***, IEEE CS Press, Los Alamitos, Calif., 1991, pp. 156-163.
15. M. Sarkar and M. Brown, "Graphical Fisheye Views," ***Comm. of the ACM***, Vol. 37, No. 12, 1994, pp. 73-84.
16. J. Lamping, R. Rao, P. Pirolli, "A Focus + Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies," ***Proc. ACM CHI Conf. on Human Factors in Computing (CHI 95)***, ACM Press, New York, 1995, pp. 401-408.



1 Example for density functions: (a) data set, (b) square wave, and (c) Gaussian.

Kernel density estimation provides a powerful framework for finding clusters in large data sets. Researchers have proposed various kernels K in the statistics literature. Examples include square wave functions or Gaussian functions. Figure 1 shows an example for the density function of a 2D data set using a square wave and Gaussian kernel.

A detailed introduction to kernel density estimation lies beyond the scope of this article, but you can find more information elsewhere.[13,14] Previously, we showed that clusters can be defined as the maxima of the density function, which lie above a certain noise level ξ.[11]

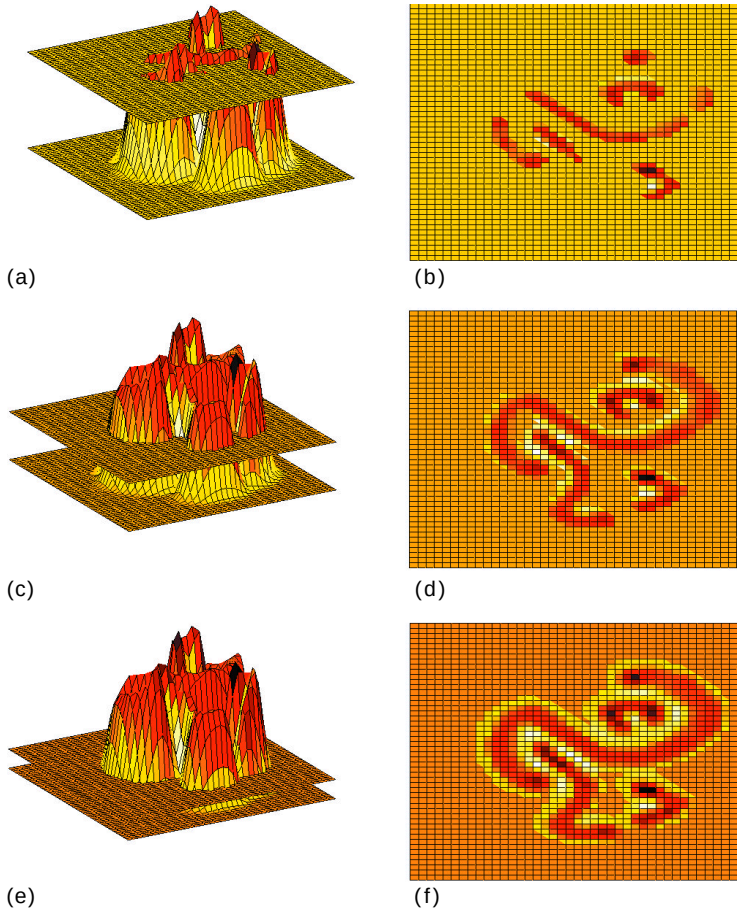
Definition 2: Center-defined cluster. A *center-defined cluster* for a maximum x^* of the density function $\hat{f}^D$ is the subset $C \subseteq D$, with $x \in C$ being density-attracted by x^* and $\hat{f}^D(x^*) \geq \xi$. We call points $x \in D$ *outliers* if they're density-attracted by a local maximum x^* with $\hat{f}^D(x^*) < \xi$.

According to this definition, each local maximum of the density function that lies above the noise level ξ becomes a cluster of its own and consists of all points density-attracted by the maximum. In the example presented in Figure 1, we obtained four clusters for most

possible noise levels. The notion of *density-attraction* is defined by the gradient of the density function. The definition can be extended to clusters defined by multiple maxima and can approximate arbitrarily shaped clusters.

Definition 3: Multicenter-defined cluster. A *multicenter-defined cluster* for a set of maxima X is the subset $C \subseteq D$, where $\forall x \in C$ exists a $x^* \in X$ with $\hat{f}_B^D(x^*) \geq \xi$, x is density-attracted to x^* and $\forall x_1^*, x_2^* \in X$ exists with a path $P \subset F^d$ from x_1^* to x_2^* above noise level ξ.

Figure 2 shows the multicenter-defined clusters for different ξ. With an increasing ξ, more and more clusters get separated. Note that the resulting clusters may have an arbitrary shape.



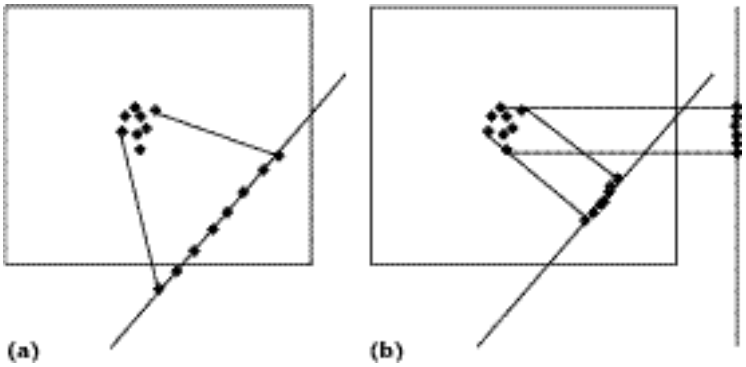
2 Example of multicenter-defined clusters for different ξ. (a), (b) $\xi = 4$; (c), (d) $\xi = 2$; and (e), (f) $\xi = 1$.

Projections and separators

Determining clusters essentially involves data points condensing into groups and separating from other groups of data points. Using the point density for clustering, clusters become separated by the valleys between the maxima. In the OptiGrid approach, we use lower dimensional projections of the high-dimensional data to effectively determine the separations. Useful projections must be contracting, since only contracting projections provide the upper bound property (see Lemma 5) necessary for detecting the separating valleys correctly.

Definition 4: Contracting projection. A contracting projection for a given d-dimensional data space S and an appropriate metric $\|\cdot\|$ is a linear transformation P defined on all points $x \in S$

$$P(x) = Ax \text{ with } \|A\| \max_{y \in s} \left(\frac{\|Ay\|}{y} \right) \leq 1$$



3 General (a) and contracting (b) projections.

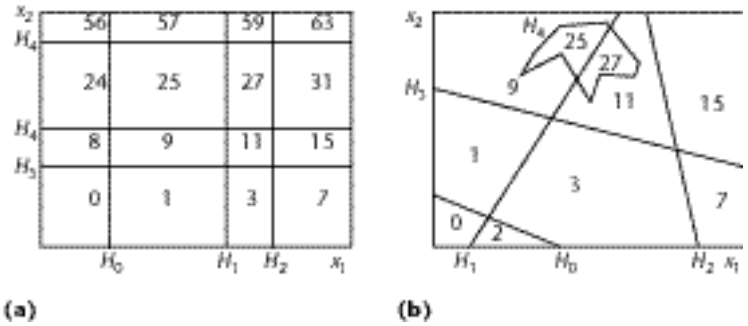
Figure 3 shows an example of the difference between general and contracting projections.

Lemma 5 states that the density at a point x' in a projected space of the data is an upper bound for the density on the plane orthogonal to the projection plane in the original feature space. The lemma shows a way of determining separators that partition the data without dividing the clusters. Informally, Lemma 5 states that a set of data points that can be separated in a contracting projection (with a small error) are also separated in the original data space (without a larger error).

Lemma 5: Upper bound property of contracting density projections. Let $P(x) = Ax$ be a contracting projection, $P(D)$ the projection of the data set D, and $\hat{f}^{P(D)}(x')$ the density for a point $x' \in P(S)$. Then, using the same kernel for $\hat{f}^D$ and $\hat{f}^{P(D)}$,

$$\forall x \in S \text{ with } P(x) = x' : \hat{f}^{P(D)}(x') \geq \hat{f}^D(x)$$

Definition 6: Separator. A *separator* is a



4 Examples of 2D grids: (a) regular and (b) irregular.

geometric object that partitions R^d into two half spaces h_0, h_1. The decision function $H(x)$ determines the half space, where a point $x \in R^d$ is located:

$$H(x) = \begin{cases} 1 & x \in h_1 \\ 0 & x \text{ else} \end{cases}$$

To determine the decision function efficiently, we use contracting projections of the form $P : R^d \to R^{d'}$, $d' < d$. Examples for separators are $(d-1)$-dimensional hyperplanes defined by a split value x_s or multidimensional cylinders defined by a closed polygon p. The formulas for the decision functions follow:

1. Hyperplane P:

$$R^d \to R^1,\ H(x) = \begin{cases} 1 & P(x) \geq x_s \\ 0 & \text{else} \end{cases}$$

2. Cylinder P:

$$R^d \to R^2,\ H(x) = \begin{cases} 1 & P(x) \text{ inside of polygon } p \\ 0 & \text{else} \end{cases}$$

Multidimensional grid

The combination of several separators results in a multidimensional grid. Since we can't store the grid explicitly in high-dimensional space, we need a coding function c that assigns a label to all points belonging to the same grid cell. We define a general notion of arbitrary (nonequidistant, irregular) grids as follows.

Definition 7: Multidimensional grid. A *multidimensional grid G* for the data space S is defined by a set $H = \{H_1, \ldots, H_k\}$ of separators. We define the coding function $c^G: S \to N$ as follows (where N stands for the space of natural numbers):

$$x \in S,\ c(x) = \sum_{i=1}^{k} 2^i \cdot H_i(x)$$

The grid notation lets us determine the relevant subsets efficiently. Examples of a regular and an irregular grid appear in Figure 4.

The HD-Eye algorithm

The HD-Eye algorithm builds on the OptiGrid algorithm for clustering high-dimensional data sets. The OptiGrid algorithm works recursively. In each step, it partitions the actual data set into a number of subsets if possible. The algorithm treats the subsets containing at least one cluster recursively. The partitioning uses a multidimensional grid defined by a number of separators. OptiGrid chooses the separators in regions with minimal point density. The recursion stops for a subset if no good separators can be found. Another similar approach, Classification and Regressions Trees (CART),[15] also automatically partitions the data in a recursive way, but uses additional data to guide the partitioning.

The OptGrid algorithm, as described so far, mainly detects center-defined clusters. However, it easily extends to also detect multicenter-defined clusters according to Definition 3. The algorithm just has to evaluate the density between the center-defined clusters determined by OptiGrid and link the clusters if the density is high enough.

Determining good projections and finding complex separators necessary for building the multidimensional grid proves difficult. In addition, it can't be done automatically because of the diverse cluster characteristics of different data sets. Visualization techniques can improve the effectiveness of the clustering process considerably and find clusters otherwise missed.

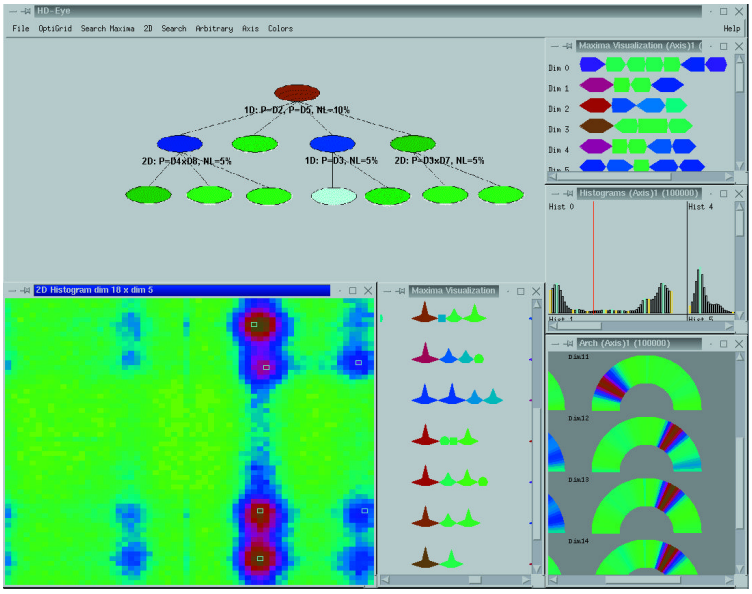
Now we'll give a brief overview of our visual clustering tool, HD-Eye. Two main tasks need visual support: first, finding the appropriate contracting projections for partitioning the data and second, finding and specifying good separators based on these projections. Both tasks require the user's intuition and creativity and can't be done automatically. The main difference between the OptiGrid and the HD-Eye approach lies in the visual determination of the projections and separators. Through visual feedback, users gain a deeper understanding of the data set. Therefore, they can identify additional projections and separators that the original OptiGrid algorithm can't find.

The HD-Eye algorithm works as follows for a data set D:

1. Initialize the cluster hierarchy with $root \leftarrow D$
2. While a node v with the data set D_v can be split

- Visually find projections usable for separating data $\to$ $P = \{P_1, \ldots, P_k\}$
- Visually find separators $\to \{H = H_1, \ldots, H_r\}$
- Construct multidimensional grid G defined by the separator set H and insert all data point $x \in D_v$ into G
- Determine clusters, that is, determine highly populated grid cells in G
- Refine the clusters
- Add the clusters as child nodes of v to the cluster hierarchy

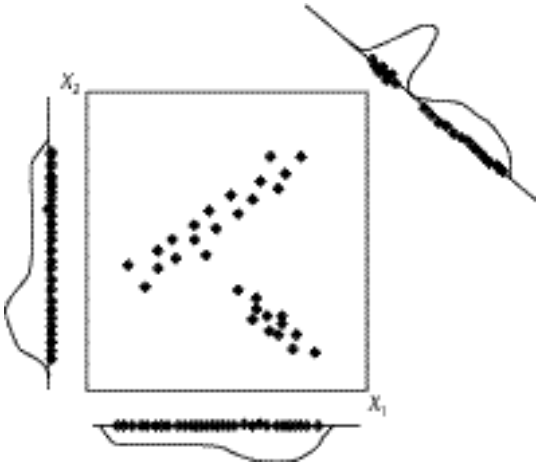
The HD-Eye algorithm splits the data set recursively. The recursion tree forms a cluster hierarchy that represents the cluster structure intuitively. The projection and separator tree (or simply cluster tree) appears in the main overview window of the HD-Eye system (see Figure 5). The algorithm lets the user partition the data set first using a set of simple 1D and 2D orthogonal projections, then using a set of complex hyperpolygonal separators. Multiple levels of the cluster hierarchy use complex separators that recursively refine the cluster structure using different noise levels (NL). Note that the separators allow the user to find clusters of arbitrary shape and complex (linear, quadratic, and higher order) dependencies.



5 Examples of a projection and separator tree. Clockwise from the top: a projection and separator tree, iconic representation for 1D projections, curve-based density plots, color-based density plots, iconic representation for 2D projections, and a color-based 2D density plot.

Visual finding of projections and separators

One of the essential parts of the HD-Eye algorithm is to find a set of projections $P = \{P_1, \ldots, P_k\}$, useful for separating the data into clusters. In other work[12] we showed that axes-parallel projections suffice for detecting center-defined clusters with no linear dependencies between the attributes. Clusters with linear or higher order dependencies between the dimensions (arbitrary-shaped clusters) require general projections and separators. Figure 6 shows an example. You can see that the axes parallel projections don't preserve the information well, which is necessary for clustering.



6 Example of the need for general projections.

Visual techniques that preserve some characteristics of the data set can be invaluable for obtaining good separators. In contrast to dimension reduction approaches such as principal component analyses (FastMap[16]) or projection pursuit,[17] our approach doesn't require that a single projection preserve all clusters. In the projections some clusters may overlap and therefore not be distinguishable. For our approach, we only need projections that separate the data set into at least two subsets without dividing any clusters. The subsets may then be refined using other projections and possibly partitioned further based on separators in other projections. Since we only use contracting projections, partitioning the data set in the minima of the projections doesn't cause large errors in the clustering (see Lemma 5). Based on the visual representation of the projections, it's possible to find clusters with unexpected characteristics (shapes, dependencies) very difficult or impossible to find by tuning the parameter settings of automatic clustering algorithms.

The most important information about a projection is whether it contains well-separated clusters. Note that well-separated clusters in the projection could result from more than one cluster in the original space. If it's possible to partition the data into subsets, the more complex structure can be detected recursively using other projections. In our approach we code the important information (whether the projection contains well-separated clusters) in three different ways:

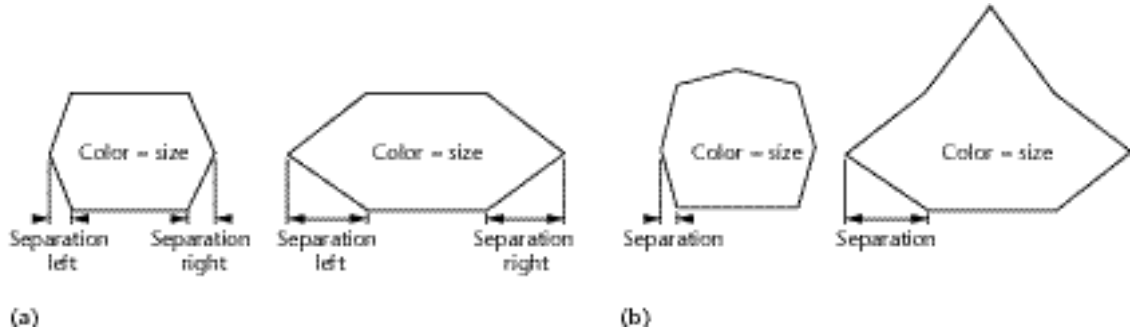
1. abstract iconic display
2. color-based point density
3. curve-based point density

Method 1 has two variants. The first applies to projections $P : \mathrm{R}^d \rightarrow \mathrm{R}^1$, while the second applies to general contracting projections $P : \mathrm{R}^d \rightarrow \mathrm{R}^{d'}$, $d' \leq d$. Methods 2 and 3 are limited to $d' \leq 2$.

Next we'll describe the three methods and discuss how they can aid in finding interesting projections of the form $P : \mathrm{R}^d \rightarrow \mathrm{R}^{d'}$, $d' \leq 2$. Then we'll briefly describe how the separators may be specified directly within the visualizations.

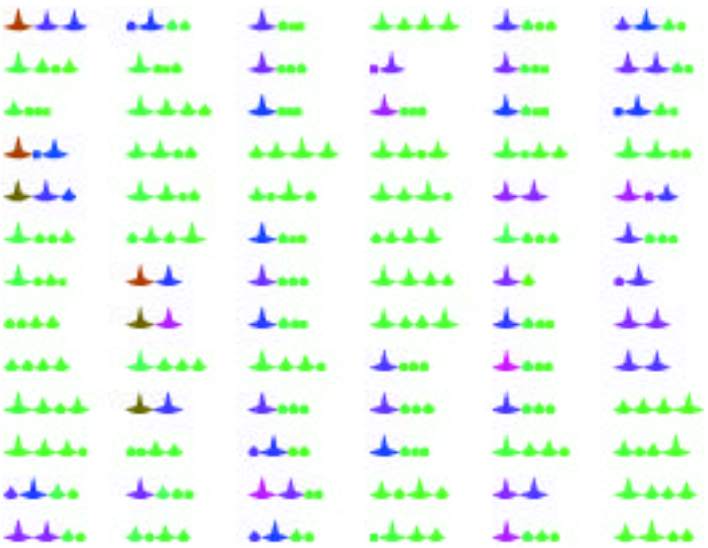
Abstract iconic visualizations of projections

The large number of possible contracting projections demand a visualization technique for displaying the main properties of the data distribution of as many projections as possible. Since we're interested in finding



7 Structure of the icons: (a) one dimensional and (b) multidimensional.

8 Examples of the icons: (a) one dimensional and (b) multidimensional.



9 Iconic representation of a large number of projections.

projections that partition the data set well, the main properties include

- the number of maxima of the projected data's density function,
- the number of data items belonging to the maxima, and
- how well the maxima are separated from each other.

In our approach we determine these properties based on histogram information of the point density in the projected space. The abstract iconic display method uses an icon-based technique to display the properties. The number of data points belonging to the maxima corresponds to the icon's color. The color follows a given color table, ranging from dark colors for large maxima to bright colors for small maxima.

The measure of how well a maxima is separated from the others corresponds to the icon's shape. The degree of separation varies from sharp pikes for well-separated maxima to blunt pikes for weak-separated maxima (see Figure 7). The icons for the maxima of a projection are arranged in a vertical line. In the 1D case (variant 1), a natural order of the maxima exists and the degree of separation can be measured in two directions (left and right). The degree of separation to the left corresponds to the height of the smallest bin between the current maxima and the next maxima on the left, and analogously for the separation to the right. In the multidimensional case, the direction of the separation with respect to other clusters doesn't make sense. Therefore, in the visualization we order the maxima to obtain a good contrast between the important maxima and the less important ones. The used heuristic determines the degree of separation of a cluster in a projection by the highest peak at the border of the cluster, which means the smaller the maximum density at the border, the better the separation of the cluster. For the cluster detection in the projected space we employ a variant of the DenClue algorithm.[11] Figure 7 shows the structure of the icons, and Figure 8 shows examples of visualizations resulting from a 1D and a 2D projection.

The iconic representation of the density function's maxima serves for identifying the interesting projections and for finding good separators. Figure 9 shows an example with a larger number of 2D projections. The interesting projections have at least two well-separated maxima (dark, sharp icons).

Color- and curve-density visualizations of projections

The abstract iconic visualizations of the projection help in finding interesting projections among a large number of projections. For further exploration such as testing whether the projections allow useful separators,

we can use color-density and curve-density plots. Figure 10 shows 1D examples. The darkness of the color increases with higher point density. The examples correspond to the example of the abstract 1D iconic visualization method (Figure 8a). We used color-density plots in the form of an arc to prevent confusion among different plots in cases of overview displays.

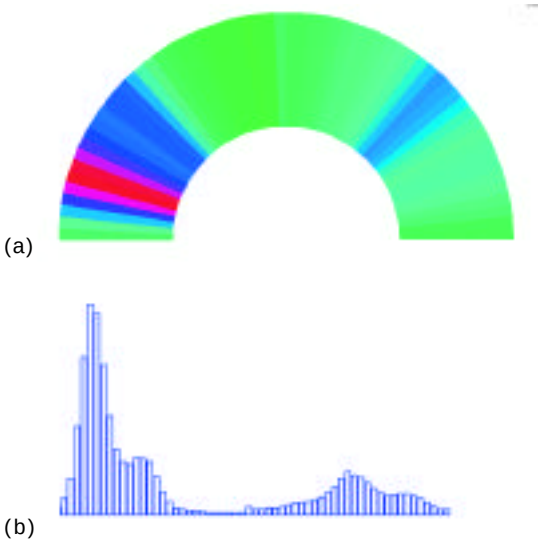
In addition to the 1D plots, we also support 2D color-density plots. The basic idea is to code the point density of the projection by color. Figure 11 shows an example of a 2D color-density plot, which corresponds to the abstract iconic representation in Figure 8b. The user can also use color-density plots to specify complex hyperpolygonal separators intuitively (see the section "Visual finding of separators").

Methodology for preselecting interesting projections

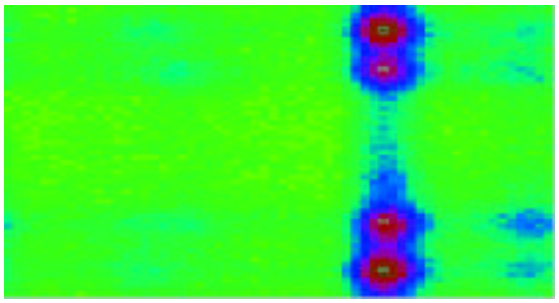
Since the number of potentially interesting projections is very large (infinite for arbitrary multidimensional projections), we must have some procedure for preselecting projections. The HD-Eye system uses an interactive optimization procedure, which starts by proposing some projections to the user. The initial set of projections consists of all axes parallel projections, some diagonal projections, and some random combinations of previous projections. From this set, the user can select the interesting projections. In case the presented projections don't suffice, the user can generate other combinations from the selected ones using crossover and mutation operators applied to the projection matrices. During a transition from an iteration to the next, the algorithm preserves the selected projection and generates new ones by applying the crossover and mutation opertators to the selected projections. The procedure acts as a sort of genetic algorithm where the user's selections define the fitness function. The iteration for good projections stops when the user finds satisfying projections or realizes that the data set can't be partitioned further.

Visual finding of separators

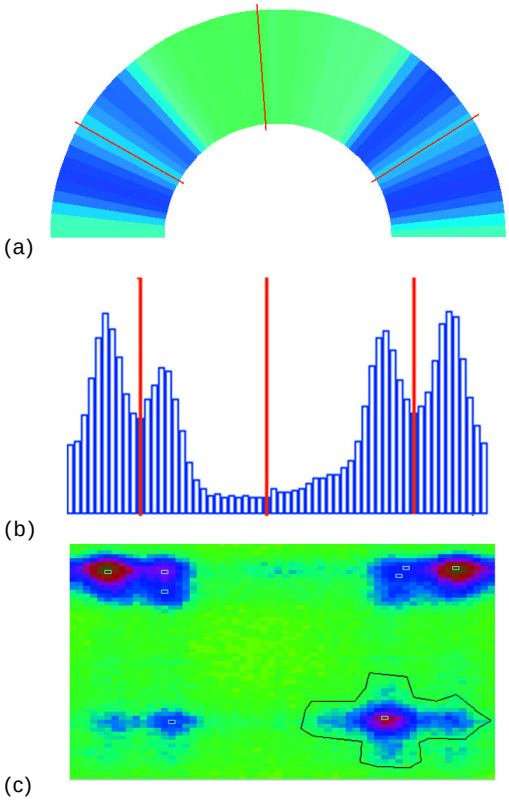
When users find a good projection that separates data points into clusters, they can use the visualization to directly specify one or multiple separators. Due to unknown characteristics of the clusters, finding separators can't be done effectively in an automatic way. In the HD-Eye system, users visually specify the separators with interactive tools. A separator can be defined by simple geometric objects such as split lines or closed split polygons in the projected space. The borders of the split objects are best placed in regions of relatively low point density. Using the upper bound property (Lemma 5), the point density in a projection represents an approximation of the point density in the data space. For finding separators in 1D projections, we use the color and curve density plots introduced previously with a high granularity (that is, with a large number of grid cells). Users place split lines for the separators directly in the visualizations using the mouse. In the 2D case, the split polygons can be drawn directly into the 2D color density plots. Figure 12 shows examples of separator lines in 1D color- and curve-density plots and a 2D color-density plot.



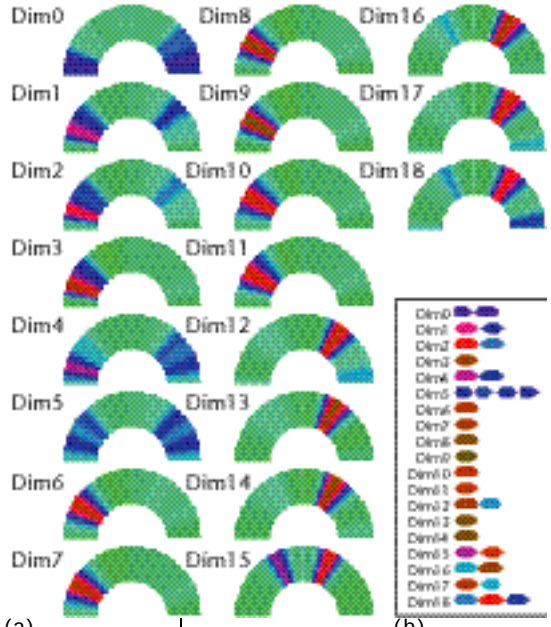
10 Examples of the 1D density plots: (a) color and (b) cur ve.



11 Example of the 2D color density plots.

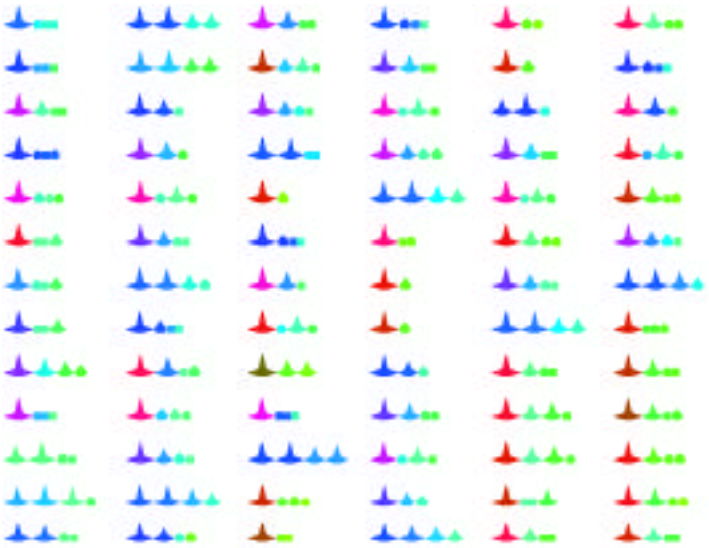


12 Examples of separators: (a) 1D color density plot, (b) 1D cur ve density plot, and (c) 2D color density plot.



(a) (b)

13 Separators for the molecular biology data set (19 dimensional): (a) icon groups and (b) color-density plots.

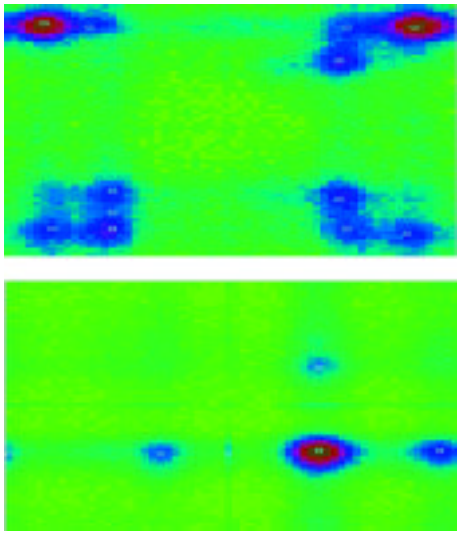


14 Separators for the molecular biology data (19 dimensional).

Experiments

Here we present the results of applying the HD-Eye system to real high-dimensional data from molecular biology. The experiments demonstrate the effectiveness of our visual techniques.

The data used for the experiment comes from a complex simulation of a very small but flexible peptide.[18] The data generated by the simulation describe the conformation of the peptide as a 19-dimensional point. The simulation covers a period of 50 nanoseconds with two snapshots taken every picosecond, resulting in about 100,000 data points. (The simulation took several weeks of CPU time.) The simulation's purpose was to determine the molecule's behavior in the conformation space. The large amount of high-dimensional data makes it difficult to find, for example, the molecule's preferred conformations. This proves important for applications in the pharmaceutical industry, since small, flexible peptides form the basis for many medications. The flexibility of the peptides, however, results in peptides with many intermediate, unstable conformations. The preferred conformations correspond to large clusters and intermediate conformations show up as noise.



15 2D density plots of the separators for the molecular biology data (19 dimensional).

At first, we used 1D projections to the coordinate axes of the data. The icon-based overview plot (Figure 13a) shows that only the dimensions 0, 1, 2, 4, 5, 12, 15, 16, 17, and 18 separate clusters effectively. The color-density plots (Figure 13b) allow users to specify the separators.

In the second step, we used 2D projections. The icon-based overview plot (Figure 14) shows many 2D projections with well-separated clusters. The 2D density plots (Figure 15) offer good possibilities for polygonal separators. The plots also allow deeper insight into the correlations between the dimensions.

Conclusions

In this article we propose a new approach to visual clustering in high-dimensional data sets. Implemented in the HD-Eye system, our approach combines the strengths of an advanced automatic clustering algorithm with new visualization techniques that effectively support the clustering process by representing the important information visually. The visualization techniques use a combination of pixel-oriented density plots and iconic representations of the data and allow users to directly specify cluster separators in the visualizations. Experimental evaluation shows that the combination of automatic and visual techniques significantly improves the effectiveness of the data mining process and provides a better understanding of the results.

We believe our new approach of combining the strengths of automatic and visual methods will noticeably affect ways of clustering high-dimensional data in the future. Our plans for future work include fine-tuning our method for specific applications and extending it to include other visual representations. ■

References

1. H. Shawney and J. Hafner, "Efficient Color Histogram Indexing," *Proc. Int'l Conf. on Image Processing*, IEEE Press, Piscataway, N.J., 1994, pp. 66-70.
2. R. Methrotra and J.E. Gray, "Feature-Index-Based Similar Shape Retrieval," *Proc. 3rd Working Conf. on Visual Data - base Systems*, Chapman and Hall, London, 1995, pp. 46-65.
3. T. Wallace and P. Wintz, "An Efficient Three-Dimensional Aircraft Recognition Algorithm Using Normalized Fourier Descriptors," *Computer Graphics and Image Processing*, Academic Press, Vol. 13, 1980, pp. 99-126.
4. K. Kukich, "Techniques for Automatically Correcting Words in Text," *ACM Computing Surveys*, Vol. 24, No. 4, 1992, pp. 377-440.
5. R.T. Ng and J. Han, "Efficient and Effective Clustering Methods for Spatial Data Mining," *Proc. 20th Int'l Conf. on Very Large Databases*, Morgan Kaufmann, San Francisco, 1994, pp. 144-155.
6. M. Ester et al., "Density-Connected Sets and their Application for Trend Detection in Spatial Databases," *Proc. 3rd Int'l Conf. on Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, Calif., 1997.
7. X. Xu et al., "A Distribution-Based Clustering Algorithm for Knowledge Discovery in Large Spatial Databases," *Proc. IEEE 14th Int'l Conf. on Data Engineering (ICDE 98)*, IEEE Computer Society Press, Los Alamitos, Calif., 1998, pp. 324-331.
8. E. Schikuta, "Grid Clustering: An Efficient Hierarchical Clustering Method for Very Large Data Sets," *Proc. 13th Conf. on Pattern Recognition*, Vol. 2, IEEE CS Press, Los Alamitos, Calif., pp. 101-105.
9. T. Zhang, R. Ramakrishnan, and M. Linvy, "Birch: An Efficient Data Clustering Method for Very Large Databases," *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, ACM Press, New York, 1996, pp. 103-114.
10. W. Wang, J. Yang, and R. Muntz, "Sting: A Statistical Information Grid Approach to Spatial Data Mining," *Proc. 23rd Int'l Conf. on Very Large Databases*, Morgan Kaufmann, San Francisco, 1997, pp. 186-195.
11. A. Hinneburg and D.A. Keim, "An Efficient Approach to Clustering in Large Multimedia Databases with Noise," *Proc. 4rd Int. Conf. on Knowledge Discovery and Data Min - ing*, AAAI Press, Menlo Park, Calif., 1998, pp. 58-65.
12. A. Hinneburg and D.A. Keim, "Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering," to be published in *Proc. 25th Int'l Conf. on Very Large Databases*, Morgan Kaufmann, San Francisco, 1999.
13. B.W. Silverman, *Density Estimation*, Chapman and Hall, London, 1986.
14. D.W. Scott, *Multivariate Density Estimation*, Wiley and Sons, New York, 1992.
15. L. Breiman et al., *Classification and Regression Trees*, CRC Press, Monterey, Calif., 1984.
16. C. Faloutsos and K.D. Lin, "FastMap: A Fast Algorithm for Indexing, Data Mining, and Visualization of Traditional and Multimedia Datasets, *Proc. ACM SIGMOD Int'l Conf. on Managment of Data*, ACM Press, New York, 1995, pp. 163-174.
17. P.J. Huber, "Projection Pursuit," *The Annals of Statistics*, Vol. 13, No.2, 1985, pp. 435-474.
18. X. Daura et al., "Reversible Peptide Folding in Solution by Molecular Dynamics Simulation," *J. Molecular Biology*, Vol. 280, 1998, pp. 925-932.

Alexander Hinneburg *is an assistant in the database and visualization research group at the University of Halle and is working towards his PhD in computer science. His research interests are in mining large databases and bioinformatics. He developed and implemented two algorithms for clustering high-dimensional data, called DenClue and OptiGrid. He received an MS in computer science from the University of Halle in 1997.*

Daniel A. Keim *is a professor at the Institute for Computer Science at the University of Halle. He works in the areas of visualization and data mining. In the field of visualization, he developed several new techniques that use visualization technology for exploring large databases, and he was the chief engineer in designing the VisDB and VisualPoints systems—two visual database exploration systems. In the area of automated data mining, he participated in developing two clustering algorithms, called DenClue and OptiGrid. He received an MS in computer science from the University of Dortmund in 1990 and his PhD and habilitation in computer science from the University of Munich in 1994 and 1997, respectively.*

Markus Wawryniuk *is currently working on his MS degree in computer science at the Martin-Luther University of Halle in Germany. He is especially interested in databases and their applications. He implemented the main portion of HD-Eye system. He is also currently developing a Web interface of an image database for preserving old photographies of palestina.*

Readers may contact the authors at the Institute of Computer Science, University of Halle, Kurt-Mothes-Str. 1, 06120 Halle (Saale), Germany, e-mail {hinneburg, keim, wawryniu}@informatik.uni-halle.de.