Automated Hierarchical Density Shaving: A Robust Automated Clustering and Visualization Framework for Large Biological Data Sets

Gunjan Gupta, Alexander Liu, and Joydeep Ghosh

Abstract—A key application of clustering data obtained from sources such as microarrays, protein mass spectroscopy, and phylogenetic profiles is the detection of functionally related genes. Typically, only a small number of functionally related genes cluster into one or more groups, and the rest need to be ignored. For such situations, we present Automated Hierarchical Density Shaving (Auto-HDS), a framework that consists of a fast hierarchical density-based clustering algorithm and an unsupervised model selection strategy. Auto-HDS can automatically select clusters of different densities, present them in a compact hierarchy, and rank individual clusters using an innovative stability criteria. Our framework also provides a simple yet powerful 2D visualization of the hierarchy of clusters that is useful for further interactive exploration. We present results on Gasch and Lee microarray data sets to show the effectiveness of our methods. Additional results on other biological data are included in the supplemental material.

Index Terms—Mining methods and algorithms, data and knowledge visualization, clustering, bioinformatics.

1 Introduction

In many real-world clustering problems, only a subset of the data needs to be clustered, while the rest should be ignored. For example, consider gene-expression data sets that measure expression levels of genes compared to a control across a few thousand genes over several microarray experiments. The experiments typically focus on a specific "theme" such as stress response. Only the genes involved in the corresponding active biological processes show correlated behavior. Correlated genes are better identified via clustering or biclustering when the data from nonpertinent genes are either manually removed before cluster analysis or are pruned during the clustering process itself.

Other types of biological data with similar properties include protein mass spectroscopy (with gel electrophoresis) [19], protein arrays [40], and phylogenetic profile data [35]. Though protein mass spectroscopy technology differs greatly from protein arrays, the data sets produced can have the same format; each data point represents a protein, while the features represent the expression level of a protein across multiple experiments or samples. The goal of clustering is to find protein-protein interactions. In phylogenetic profile data, each data point represents a gene; the feature space consists of evolutionary conservation levels of genes shared across multiple species. In this paper, we will address mainly gene-expression data sets; the accompanying supplemental material contains additional experiments on additional types of data, including phylogenetic profile data.

A wide variety of parametric approaches [5] have been applied to exhaustively cluster a data set based on the assumption that each cluster is a member of some parametric family (e.g., Gaussians). However, in problems where the subset of data that clusters well is small compared to the overall data set, the "don't care" points can overwhelm a method that optimizes over all the data points. In contrast, certain nonparametric clustering algorithms (e.g., [15] and [1]) that use *kernel density estimation* [29] at each data point to find dense clusters are capable of clustering only a subset of the data.

In 1968, Wishart [44] proposed an algorithm called *Hierarchical Mode Analysis* (HMA) that used kernel density estimation to yield a compact hierarchy of dense clusters while disregarding data in sparse regions. HMA seems to have gotten lost in time and is not known to most current researchers. One reason could be that HMA is slow $(O(n^3))$ and memory intensive $(O(n^2))$ and predated the era of downloadable public-domain code. Interestingly enough, a specific setting of HMA as described in [44] yields results identical to DBSCAN [15], a widely used algorithm that was independently proposed almost three decades later.

In this paper, we present a framework called Automated Hierarchical Density Shaving (Auto-HDS) that builds upon the HMA algorithm and greatly broadens its scope and effectiveness. Key improvements include

1. creating a faster $(O(n^2))$ and much more memory efficient $(O(n \log n))$ algorithm appropriate for larger data sets,
2. the ability to use a variety of distance metrics including Pearson Distance, a biologically relevant distance measure,

- G. Gupta is with the Amazon.com, PO Box 14403, Seattle, WA 98114. E-mail: gunjan@ideal.ece.utexas.edu.
- A. Liu and J. Ghosh are with the Department of Electrical and Computer Engineering, University of Texas, 1 University Station C0803, Austin, TX 78712. E-mail: {aliu, ghosh}@ece.utexas.edu.

Manuscript received 15 Jan. 2006; revised 3 July 2007; accepted 15 Feb. 2008; published online 7 Mar. 2008.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-0214-1206.
Digital Object Identifier no. 10.1109/TCBB.2008.32.

3. a robust *unsupervised* model selection that discovers small clusters of varying densities while simultaneously pruning large amounts of irrelevant data,
4. a novel effective visualization of the resulting cluster hierarchy,
5. an extension of runt pruning [42] that enables interactive near-real-time clustering, and
6. a Java-based implementation that is available for free download and incorporates several additional features to support interactive clustering and exploration.

Certain high-throughput biological data sets have several characteristics/requirements that match the abilities of Auto-HDS, including

1. pertinent clusters within the data sets often vary in density; for example, a large number of somewhat weakly correlated genes could form a group that is as important as a small number of highly correlated genes,
2. biological subprocesses may result in subclusters within clusters, best revealed by a multilevel technique,
3. a large number of irrelevant genes or other biological entities may be present, and
4. a fully unsupervised setting is desirable as there is little or no labeled data for selecting model parameters such as number of clusters.

This also increases the usefulness of interactively visualizing and examining the cluster hierarchy for cluster comprehension and selection. Note that this complete set of desiderata that Auto-HDS addresses cannot be fulfilled satisfactorily by other existing clustering methods.

To date, applications of Auto-HDS on gene-expression data have shown good results. Our Java-based implementation, Gene Density Interactive Visual Explorer (Gene DIVER), is especially useful for practitioners. It exploits an efficient heap data structure and a serialization API to provide a memory-efficient, scalable, and platform-independent implementation of our framework. Gene DIVER also includes a sophisticated SWING-based visualization and interactive user interface, with special features for exploring clustered genes using the latest information from two online biological databases.

A subset of the work presented in this paper appeared in [22] and [23]. More details on some aspects of this paper, especially algorithms, can be found in a related technical report [21] and dissertation [19].

Notation. Boldfaced variables, e.g., $\mathbf{x}$, represent vectors, whose $i\text{th}$ element is denoted by either x_i or $x(i)$. Sets of vectors are represented by calligraphic uppercase alphabets such as $\mathcal{X}$ and are enumerated as either $\{\mathbf{x}_i\}_{i=1}^n$ or $\{\mathbf{x}(i)\}_{i=1}^n$, where $\mathbf{x}_i$ or $\mathbf{x}(i)$ are the individual elements. $|\mathcal{X}|$ represents the size of set $\mathcal{X}$. Boldfaced capital letters such as $\mathbb{M}$ represent 2D matrices. $\mathbb{R}$ and $\mathbb{R}^d$ represent the domain of real numbers and a $d\text{-dimensional}$ vector space, respectively.

2 Related Work

Clustering has a very long and storied history [16], [28] and has been extensively studied and applied across a wide range of disciplines. This has naturally resulted in a very wide variety of clustering approaches, from information theoretic [11] to graph theoretic [41] to those inspired by network flows [14]. Specific applications have inspired particular approaches, e.g., biclustering methods for microarray data analysis [33], [3] and directional generative models for text [4]. This rich heritage is largely due to not only the wide applicability of clustering but also the "ill-posed" nature of the problem and the fact that no single method can be best for all types of data/requirements. To keep this section short, we will concentrate only on work most pertinent to this paper: density-based approaches and certain techniques tailored for biological data analysis.

A variety of density-based methods have been developed that use different notions of density to identify clusters. One of the most widely cited density-based algorithms is DBSCAN [15]. In DBSCAN, if a point has at least *MinPts* points within a distance of ϵ, then all these points are assigned to the same cluster. DBSCAN is particularly well suited for low-dimensional data for which fast database indexing methods can be used. Different choices of ϵ and *MinPts* can give dramatically different clusterings; choosing these parameters are a potential pitfall for the user. OPTICS [1] proposed a visualization to make it easier to select these parameters and also supports the discovery of a hierarchy on which additional interactive exploration can be done.

As mentioned earlier, HMA [44] was a pioneering approach for finding a few dense regions in the data. It searched for "modes" or local peaks of the underlying multivariate distribution without restricting this distribution to any particular parametric shape. It could also overcome "chaining," a problem that occurs when two valid clusters are connected by a chain of spurious points, which adversely affected some popular methods of its time, most notably single-link hierarchical clustering.

A remarkable feature of HMA is its ability to provide a highly compact hierarchy that depicts the (hierarchical) relationship among the different modes/dense regions in the data. The compactness fundamentally stems from its ability to ignore the less dense or "don't care" points while building the hierarchy. In many ways, HMA was ahead of its time; one of the cluster labeling and selection methods suggested in [44] results in an algorithm whose output is identical to that of DBSCAN. Furthermore, the method in [44] also contains a solution for selecting DBSCAN's ϵ parameter, which is otherwise difficult to choose particularly for high-dimensional data sets. We describe HMA in more detail in Section 3.2.

DHC [30] proposes a hierarchical grouping of biological time-series data that can be used to visually browse similar genes. Although the general idea and motivation of [30] seem related to what we propose in this paper, the algorithms and the key issues that our method resolves are significantly different. The cluster hierarchy built by DHC uses a heuristic of *attraction* that assumes that the data is uniformly distributed in the original $d\text{-dimensional}$ space. However, high-dimensional data often resides in much lower dimensional manifolds [43].

TABLE 1
Salient Characteristics of HDS and Other Related Methods

Method	DBSCAN	OPTICS	HMA	Gene Shaving	DHC	Auto-HDS
TC high-d	n^2 [a] to $n^2\log(n)$	$>$ DBSCAN	n^3	n^3	Unavail.	n^2 [b] to $n^2\log(n)$
TC low-d	n^2	>DBSCAN	n^3	n^3	Unavail.	n^2
TC Indexed low-d	$n\log n$	>DBSCAN	n^3	NA	NA	$n\log n$
Main memory use	$O(n)$	>DBSCAN	$O(n^2)$	$O(n^2)$	Unavail.	$O(n\log n)$ [c]
Cluster Hierarchy	No	Yes	Yes	No	Yes	Yes
Compact Hierarchy	No	No	Yes	No	Yes	Yes
Overlapping Clusters	No	No	No	Yes	No	No
Data type	low-d spatial [d]	low-d spatial	Unavail. [e]	gene-exp.	gene-exp.	high-d bio. data [f]
Dist. Func.	Euclidean [g]	Euclidean [h]	high-d, Euclidean	Sq. Euclidean [i]	time series	Pearson Distance [j]
Visualization	No	Yes	No	No	Yes	Yes
Model Selection	No	No	No	No	Yes	Yes
Auto. Cluster Selection	No	No	No	No	No	Yes
Select mixed dens. clust. [k]	No	No	No	Yes	No	Yes

For large high-dimensional biological data sets. "TC" stands for time complexity.
[a] *For $n_{avg\epsilon} < n/\log(n)$.*
[b] *For $n_{avg\epsilon} < n/\log(n)$.*
[c] *Gene DIVER implementation.*
[d] *As implemented by [15] using indexing; the current popular usage. A modification that works well for high-dimensional results in the DS algorithm (Section 4).*
[e] *We have not found any large-scale applications.*
[f] *Pearson distance enables application to a variety of biological data sets.*
[g] *As tested and applied popularly.*
[h] *Same as DBSCAN.*
[i] *As a consequence of using PCA.*
[j] *Also applicable with cosine similarity and euclidean distance.*
[k] *All the selected clusters need not have the same density.*

Note that since most density-based algorithms, including both DBSCAN and DHC, have difficulties with high-dimensional inputs, typically, a feature selection/extraction preprocessing step needs to be employed first to reduce the dimensionality when clustering high-dimensional data with a density-based algorithm. In contrast, in our experiments, we were able to apply Hierarchical Density Shaving (HDS) directly to the original input space and still obtain good results.

Specialized algorithms have been proposed that address other issues with clustering biological data [31], [38], [12], [10]. For example, discovering overlapping gene clusters is important since many genes participate in multiple biological processes [2], [33]. Gene Shaving [26] repeatedly applies Principal Component Analysis in a greedy fashion to find small subsets of correlated genes and allows them to belong to multiple clusters. For each discovered subset, Gene Shaving *shaves* a fraction of the least relevant genes at each iteration. We reuse the word "shaving" in Auto-HDS in the same context. However, Gene Shaving is different from our method in many ways. Gene Shaving uses the Squared euclidean distance to measure loss. This measure is not appropriate for characterizing additive and multiplicative coexpression patterns [10]. While Gene Shaving requires the number of clusters k as an input, Auto-HDS finds k automatically. Gene Shaving greedily finds overlapping clusters one at a time; the next cluster is found by using orthogonalized residue obtained after removing the previous cluster. In contrast, HDS finds all the clusters from a hierarchy built by one shaving sequence. HDS performs shaving on points ordered by density, whereas Gene Shaving orders and shaves genes that have the least correlation with the principal component.

Table 1 compares some existing approaches with our framework using key features that are relevant for clustering large high-dimensional biological data sets.

3 Preliminaries

3.1 Distance Measure

Consider a set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subseteq \mathbb{R}^d$ of data points for which a relevant symmetric distance measure $d_S(\mathbf{x}_i, \mathbf{x}_j)$ is available or readily computable.[1] Let $\mathbf{M}_S$ denote the corresponding $n \times n$ symmetric distance matrix, i.e., $\mathbf{M}_S(i,j) = d_S(\mathbf{x}_i, \mathbf{x}_j)$. The algorithms described in this paper only require $\mathbf{M}_S$ as input; an explicit embedding of the data points in $\mathbb{R}^d$ is not required. For a given d_S and a positive number r_ϵ, the (local) *density* $\rho_{r_\epsilon}(\mathbf{x})$ at any given point $\mathbf{x}$ is proportional to the number of points in $\mathcal{X}$ that are within r_ϵ of $\mathbf{x}$:[2]

$$\rho_{r_\epsilon}(\mathbf{x}) \propto |\{\mathbf{y} \in \mathcal{X} : d_S(\mathbf{y}, \mathbf{x}) \le r_\epsilon\}|. \quad (1)$$

Note that a "rectangular" kernel has been chosen, leading to certain computational advantages later on.

3.2 Hierarchical Mode Analysis

The HMA algorithm uses the notion of density defined by (1) to discover the distinct modes corresponding to the dense regions in $\mathcal{X}$. Given an integer $n_\epsilon < n$, HMA examines the radii of spheres centered at each point and containing n_ϵ points and sequentially marks the points as being "dense" in ascending order of the corresponding sphere sizes. Each successive "dense" point is also given a cluster identity, which either indicates membership to a previously formed cluster or the creation of a new cluster.

1. Popular choices for d_S include the euclidean distance and "1 minus Pearson correlation."
2. Including $\mathbf{x}$ itself.

Note that n_ϵ serves as a scale parameter [9] and governs the smoothness of the solution. The key steps are

1. Using the distance matrix $\mathbf{M}_S$, determine the distances $d^{n_\epsilon}(\mathbf{x})$ from each point $\mathbf{x}$ to its n_ϵth nearest point.
2. Sort the distances d^{n_ϵ} in ascending order. The smallest distance in d^{n_ϵ} is the distance between the "densest" point and its n_ϵ nearest neighbors. This densest point forms the first cluster mode.
3. The next dense point is the point with the next smallest value in d^{n_ϵ}; r_ϵ is set to this value. The algorithm takes one of three actions:
   - i The new point does not lie within r_ϵ of another dense point, in which case it initializes a new cluster mode.
   - ii The point lies within r_ϵ of dense points from one cluster only, and the point is added to this cluster.
   - iii The point is within r_ϵ of dense points from multiple clusters. In this case, the clusters concerned are fused into one, and the point joins the fused cluster.
4. A note is kept of the nearest neighbor distance between clusters. Whenever r_ϵ exceeds the distance between two clusters, the two clusters merge into a single cluster.
5. Steps 3 and 4 are iterated until all points are clustered. Note that at the end of the ith iteration, the i densest points have been assigned cluster labels.

For the set of labeled points $\mathcal{G}$, at the end of the ith iteration of HMA, it can be shown that two dense points $\mathbf{x}, \mathbf{y} \in \mathcal{G}$ belong to the same cluster if $d_S(\mathbf{x}, \mathbf{y}) < r_\epsilon$. Consequently, if there exists a chain of points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{m-1}, \mathbf{x}_m$ in $\mathcal{G}$ such that $\{d_S(\mathbf{x}_l, \mathbf{x}_{l-1}) < r_\epsilon\}_{l=2}^m$, then $\mathbf{x}_1$ and $\mathbf{x}_m$ also belong to the same cluster.

HMA produces a cluster hierarchy that describes the relationship between clusters in different iterations. When a point forms a new cluster mode, this corresponds to the birth or emergence of a cluster; when two clusters merge, this corresponds to two parent clusters joining to form a child cluster. If a point simply joins one existing cluster, then that cluster expands. HMA is able to find a very compact hierarchy of clusters that corresponds to the actual "modes" or generating distributions, based on its ability to treat less dense points as "don't care" points or outliers at any given stage.

4 Density Shaving (DS)

The cluster labels in the ith iteration of HMA cannot be found without the cluster labels from the previous iteration. We now introduce an algorithm called DS, which can directly find the same clustering as any arbitrary iteration of HMA. Along with the distance matrix $\mathbf{M}_S$ and n_ϵ, DS takes an additional input parameter f_{shave}, the fraction of least dense points to *shave* or exclude from consideration. The parameter r_ϵ is computed using $r_\epsilon = d^{n_\epsilon}(n_c)$, where d^{n_ϵ} is defined as in step 2 of HMA, and $n_c = \lceil n(1 - f_{shave}) \rceil$ is the total number of points assigned cluster labels. The output of the algorithm consists of k clusters labeled 1 to k formed by the set $\mathcal{G}$ of n_c densest points and a "don't care" set $\mathcal{O}$ containing the remaining points that are labeled zero.

TABLE 2
A Summary of the Data Sets Used

Dataset	Source	n	d	D	true k	k_A
Gasch	Mic.	173	$6,151$	d_p	12	11
Lee	Mic.	5,612	591	d_p	NA	9
Sim-2	Sim.	1,298	2	Euc.	5	5

Mic. stands for gene-expression data from microarray experiments, Euc. stands for euclidean distance, d_p stands for Pearson distance (see (5) in the Appendix), and D is the distance function used for clustering.

DS works by applying a graph traversal process to discover the clusters formed by the n_c densest points. It simply uses the fact that two points $\mathbf{x}_1$ and $\mathbf{x}_m \in \mathcal{G}$ are in the same cluster if and only if there exists a chain of dense points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{m-1}, \mathbf{x}_m$ such that $\{d_S(\mathbf{x}_l, \mathbf{x}_{l-1}) < r_\epsilon\}_{l=2}^m$. This is achieved by first determining the set $\mathcal{G}$ and then assigning cluster labels to the points in $\mathcal{G}$ such that the chaining rule described above is obeyed. Details of the algorithm, including its pseudocode, are contained in [21]. By using heap sort to find the neighbors within r_ϵ of dense points, an implementation of DS with a time complexity of $O(max(n^2, nn_{avg\epsilon}\log(n)))$ is possible,[3] where $n_{avg\epsilon}$ is the average neighborhood size within distance r_ϵ of the dense points.

The ability of DS to estimate an r_ϵ corresponding to f_{shave} automatically is very useful in practice, especially when using Pearson Distance (which scales between zero and two, see the Appendix), where there is no intuitively obvious value of r_ϵ to select. For example, in our experiments on very high-dimensional data, we found that the difference in the value of r_ϵ when clustering 10 percent less points is usually small (e.g., ~0.01 for clustering 500 versus 550 points for Gasch genes (see Table 2)) and does not follow volumetric intuitions of a uniform distribution.

While DS can be applied as a stand-alone clustering algorithm, we mainly use DS to construct the HDS algorithm described in Section 5.

Connection between DBSCAN, DS, and HMA. The ith of the n iterations of HMA corresponds to an r_ϵ equal to the ith smallest value in d^{n_ϵ} and results in three types of points: 1) a set of dense points that have cluster labels, 2) a set of nondense points that are within r_ϵ of at least one dense point, and 3) the rest of the nondense points. If each point in the second set is instead given the same cluster label as its nearest dense point (in a specific order to break ties as suggested in [44]), the resulting clustering is identical to that of the DBSCAN algorithm. Thus, DS (and the corresponding iteration of HMA with $i = n_c$) obtains the same clustering as DBSCAN for points of type 1 but does not cluster points of type 2, whereas DBSCAN clusters points of both types 1 and 2. Thus, in DBSCAN, it is possible to label points that are not dense but rather are on the periphery of a dense neighborhood.

3. Section 8 describes an implementation based on this approach.

If one were to build a hierarchy using DBSCAN instead of DS (as described in the next section), there are an additional number of problems. In the hierarchy, DBSCAN can let a less dense point be incorporated before a more dense point. In addition, the neighborhoods of DBSCAN are order dependent since a nondense point can be the neighbor of more than one dense point. Moreover, there is no way to control the number of points that are clustered by DBSCAN except by trial and error, whereas in DS, this number is directly provided by specifying f_{shave}.

5 Hierarchical DS (HDS)

In this section, we develop a technique called HDS that finds a good approximation of the HMA hierarchy and runs much more quickly than HMA. Conceptually, the core of HDS can simply be thought of as several calls of DS that calculate selected iterations of HMA. HDS exploits the ability of DS to independently compute any iteration of HMA. This allows HDS to compute only a key subset of HMA iterations. It further exploits the fact that since the HMA iteration cluster labels are nested, any subset of HMA iterations also forms a hierarchy.

Broadly speaking, HDS consists of two stages. First, DS is run several times[4] to compute a subset of the iterations in the HMA hierarchy. Conceptually, n_ϵ is given as input and held fixed across all runs of DS while f_{shave} is varied. This creates a set of cluster labels stored in a matrix $\mathbf{L}$, where $\mathbf{L}(i, j)$ is the label of the $i\text{th}$ point on the $j\text{th}$ run of HDS. Thus, a column of $\mathbf{L}$ corresponds to a particular iteration of HDS. The second stage of HDS involves remapping of the labels in $\mathbf{L}$ so that they correspond to the labels in the HMA hierarchy. We also apply an additional step to further smooth and denoise the cluster hierarchy in $\mathbf{L}$, resulting in a refined version of the HMA hierarchy referred to as the HDS hierarchy.

Note that because each run of DS is independent, the iterations of HDS can be run in any order. The HDS hierarchy is created in a top-down manner as compared to the HMA hierarchy, which is created bottom-up. The reason for this will be explained in the next section.

5.1 HDS Iterations and Initial Hierarchy Construction

Instead of going through all n iterations of the full HMA hierarchy, HDS uses a geometric scale to select a subset of these iterations. At each HDS level, the set of dense points from the previous level is reduced ("shaved") by a constant fraction r_{shave} $(0 < r_{shave} \leq 1)$. This *exponential shaving* process has the ability to perform finer shavings as clusters get smaller, thus preserving the ability to discover small-scale but highly dense structures. Exponential shaving also naturally leads to a scale-free model selection method described in Section 6.

For the exponential shaving method, the number of points $n_c(\mathrm{t})$ to cluster after t shavings is given by

$$n_c(t) = \lceil n \times (1 - r_{shave})^t \rceil. \tag{2}$$

4. Computationally efficient implementations of HDS do not actually run DS repeatedly but result in the same clustering; we describe HDS in this manner since this version of HDS is most easily understood.

That is, the clusters obtained on the $t\text{th}$ iteration of HDS is equivalent to running DS with $f_{shave} = 1 - (1 - r_{shave})^t$ and some n_ϵ given as input. Note that consecutive values of t can result in the same values of $n_c(\mathrm{t})$, particularly as t gets larger. We ignore such duplicates and only call the DS routine for unique values of $n_c(\mathrm{t})$. The cluster labels output by the $j\text{th}$ DS run (i.e., the $j\text{th}$ *iteration* of HDS) are saved in the $j\text{th}$ column of label matrix $\mathbf{L}$.

The final HDS iteration corresponds to the first t that gives $n_c = 1$. If we denote this t by t_{max}, then from (2), $t_{max} = \lceil -\frac{\log(n)}{\log(1-r_{shave})} \rceil$. Note that because duplicate n_c values are possible, the actual number of columns in $\mathbf{L}$ can be smaller than t_{max}.

HDS versus HMA iterations. Although each distinct iteration of HDS could be computed in any order, they are organized by decreasing values of $n_c(\mathrm{t})$. The first iteration of HDS maps to the last iteration of HMA; HMA iterations are bottom-up, whereas HDS iterations are defined top-down.

While conceptually, HDS consists of at most $\lceil -\frac{\log(n)}{\log(1-r_{shave})} \rceil$ calls to DS (and indeed, we have described it in these terms in the above section), there are various ways of speeding up HDS compared to this naive implementation. Two such methods, leading to 1) recursive and 2) streaming algorithms, are described in detail in [21] and [19], respectively. It is computationally advantageous for these variants if the HDS hierarchy is defined top-down, which is one of the main reasons for the difference from the bottom-up creation of the HMA hierarchy.

5.2 Extracting a Smoothed HMA Hierarchy

We now describe the second conceptual step of HDS: relabeling the cluster label matrix $\mathbf{L}$ to correspond to the cluster hierarchy that would have been obtained using HMA while applying further denoising, resulting in an HDS cluster hierarchy. In addition to denoising, the HMA and HDS cluster hierarchy need different interpretations due to the different directions in which they build the hierarchies. In the HDS hierarchy, points are removed from each cluster at each successive iteration. If a cluster splits into two distinct clusters, one can think of this as a parent cluster dividing into two child clusters; alternatively, this can be thought of as a cluster splitting into subcluster structures. If points are removed from a cluster but no splitting has occurred, then the clusters in the two iterations can be considered the same cluster viewed under different density requirements. If all points from a cluster are removed, one can think of this as a cluster "disappearing." Thus, the number of clusters can increase, stay the same, or decrease as one progresses through the cluster hierarchy.

Although the first step of HDS produces a subset of the full HMA iterations, the labels are not based on labels in previous iterations. Hence, there is no correspondence between the different iterations computed during the first stage of HDS. Thus, a relabeling of $\mathbf{L}$ needs to be performed. Such a relabeling can also be viewed as a "compaction" of the hierarchy generated by HDS since the total number of cluster IDs in the hierarchy is reduced.

In addition, we add a refinement that allows the cluster hierarchy learned by HDS to be even more compact and noise tolerant compared to the HMA cluster hierarchy. Let any very small child cluster be called a *particle*. Specifically, we define a *particle* as any dense child cluster of size less than n_{part} points. Particles are ignored when compacting $\mathbf{L}$.

Starting from the second column of $\mathbf{L}$ (i.e., the second iteration of HDS), relabeling of $\mathbf{L}$ proceeds as follows:

1. Find the unique cluster IDs at iteration $j-1$.
2. Repeat the following for each of the clusters found in step 1:
   2.1. If all points belonging to a cluster in iteration $j-1$ are either
      a. clustered in the same cluster in iteration j,
      b. are assigned to the don't care set $\mathcal{O}$, or
      c. are assigned to a cluster that is a particle,

      then we assign the child cluster at iteration j the label of the parent cluster at iteration $j-1$. That is, one can view the cluster on iteration j as a continuation of the corresponding cluster on iteration $j-1$, barring those points that are now part of $\mathcal{O}$ or a particle.

   2.2. If the condition in step 2.1 is not satisfied, then a cluster has split into two or more child clusters. Each of these child clusters is assigned a new cluster ID.

The relabeled label matrix $\mathbf{L}_{HDS}$ output by this procedure represents a smoothed HMA hierarchy, which we refer to as the *HDS hierarchy* (see Fig. 3a for an example of $\mathbf{L}_{HDS}$).

For $n_{part}=0$, the relabeled HDS hierarchy is identical to a subset of the HMA hierarchy. A larger n_{part} acts as a smoothing parameter, somewhat similar to the effect produced by a larger n_ϵ. However, there is a subtle but important difference between the two; while n_ϵ performs density smoothing, resulting in less significant dense regions getting ignored (Fig. 1), n_{part} has a smoothing effect on the HDS hierarchy, preventing inclusion of insignificant child clusters. The use of n_{part} in HDS is also similar to *runt pruning* used in [42].

Since the hierarchy compaction process is extremely fast ($O(n \log n)$), n_{part} can be selected interactively[5] by the user to smooth clustering. In Section 7, we describe an illustrative example of how this can be achieved in conjunction with a visualization framework. These properties make interactive HMA-type analysis for finding small dense regions practical on much larger data sets.

The supplemental material (which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2008.32) associated with Gupta et al. [24] contains a simple running example of the process described in this section needed to transform $\mathbf{L}$ into $\mathbf{L}_{HDS}$.

6 Model Selection

We now describe a method for performing automatic model selection on the clusters found by HDS. In this context, model selection refers to the ability to automatically select the number and location of the "best" clusters. We introduce a novel stability criterion in order to rank the clusters and a simple algorithm for selecting the best clusters in terms of this ranking.

5. In contrast, modifying n_ϵ for smoothing is too slow to be interactive for large problems.

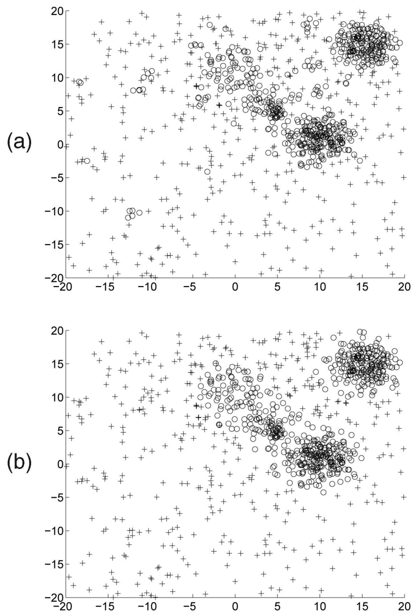


Fig. 1. The effect on clustering when n_ϵ is changed from (a) 5 to (b) 50, showing the robustness of DS with respect to parameter n_ϵ. "o" represent dense points, and "+" are outliers. Increasing n_ϵ by small amounts results in a smoothing effect on the density estimation that prunes the smallest dense regions and grows the larger ones, thus preserving the large-scale structures in the clustering. Data: Sim-2 (see Section 9.1).

6.1 Ranking Clusters

The compact hierarchy generated using the process described in Section 5.2 makes it easier to select clusters. For many applications, such a hierarchy by itself may be the end goal. However, in the absence of any supervision, a notion of cluster quality that we call *stability* can be used to rank clusters, where a higher stability gives a higher rank to a cluster. We define stability as the number of levels a cluster exists in the HDS hierarchy.

To derive the formula for stability, we first calculate the number of shavings required to reach the jth iteration of HDS. Starting from n points, the number of shavings t at the rate of r_{shave} required to get n_c points in HDS iteration t can be derived using (2):

$$t = \frac{\log(n_c) - \log(n)}{\log(1 - r_{shave})}. \tag{3}$$

Therefore, for a given cluster $\mathcal{C}$, the stability can be calculated as the number of shavings between the first and the last iteration that a cluster appears in:

$$\begin{aligned} Stab(\mathcal{C}) &= \frac{\log(n_c^e) - \log(n)}{\log(1 - r_{shave})} - \frac{\log(n_c^{s-1}) - \log(n)}{\log(1 - r_{shave})} \\ &= \frac{\log(n_c^e) - \log(n_c^{s-1})}{\log(1 - r_{shave})}, \end{aligned} \tag{4}$$

where s is the iteration of HDS where $\mathcal{C}$ first appears, and e is the last iteration where $\mathcal{C}$ survives (after which it either splits into child clusters or disappears completely). This notion of cluster stability has the following properties that makes ranking of clusters of various sizes and densities using stability robust and meaningful: 1) when ordering clusters by cluster stability, the ordering is independent of the shaving rate, and 2) it has the property of **Scale Invariance of Stability.** That is, Ordering clusters by stability results in the same ordering as when ordering clusters by the fraction of data shaved from the entire data set between the first iteration where $\mathcal{C}$ appears and the last iteration before $\mathcal{C}$ disappears. The first property follows from the fact that the denominator in (4) is a constant for all clusters $\mathcal{C}$. The second property is due to the fact that since the fraction of data shaved between two levels is constant, the fraction of points shaved between $Stab(\mathcal{C})$ levels is also constant and is given by the equation $f^{\mathcal{C}}_{shave} = (1 - r_{shave})^{Stab(\mathcal{C})}$, which can be derived from (4).

Note that if one were to use a linear shaving rate, i.e., shaving off the same number of points in each iteration, the second property would no longer hold true. This is one of the main reasons why we have chosen an exponential shaving rate instead of a linear shaving rate.

Because of our definition of cluster stability, we can now discover all the significant clusters in the relabeled hierarchy and can compare *all* clusters with each other, including clusters of different sizes, different densities, and parent and children clusters.

6.2 Selecting Clusters

Picking the most stable clusters proceeds iteratively as follows: First, make a list of all clusters eligible for selection (i.e., all clusters that are not particles). Second, pick the eligible cluster with the largest stability. Third, mark all parent and child clusters of the selected cluster as ineligible. Repeat steps 2 and 3 until there are no more eligible clusters.

The cluster members of the selected clusters are all the points assigned to that cluster on the first level the cluster appears in the HDS hierarchy. The points in the "don't care" set $\mathcal{O}$ are all the points that do not belong to any of the selected clusters.

HDS is able to select clusters that satisfy different notions of density, which is not possible using DS. An illustrative example highlighting this difference is given in Fig. 2, where HDS and DS are applied to the Sim-2 data set. For example, in Fig. 2c, DS is used to cluster 580 points, while in Fig. 2i, HDS with model selection is used to cluster 610 points. While the number of clustered points is roughly the same, HDS is able to find a larger variety of clusters since it allows different notions of density.

7 Visualization with HDS

Each row of the $n \times n_{iter}$ matrix $\mathbf{L}_{HDS}$ representing the HDS hierarchy contains the cluster label of each point in all the HDS iterations. We now sort the rows of $\mathbf{L}_{HDS}$ using a dictionary sort such that we give higher precedence to labels with smaller column indices. An example of the effect of this sorting is shown on a toy example in Fig. 3.

A simple yet powerful visualization of the HDS hierarchy can be achieved by plotting this sorted matrix, assigning separate colors to each distinct value in the matrix, and a background color to the values that are zero. Fig. 2g shows such a visualization for the 2D Sim-2 data, while Fig. 4a shows the same for the 6,151-dimensional Gasch data. Figs. 2h and 4b show visualization of clusters selected as described in Section 6 and labeled with their stability.

We call the combination of HDS, model selection, cluster selection, and visualization the *Auto-HDS* framework.

Note that just as in HMA, for a range of iterations of HDS (for example, Fig. 2d versus Fig. 2e), the number of clusters often does not change, and each of the clusters at iteration $j-1$ simply loses some points to the "don't care" cluster at iteration j. However, since HDS uses exponential shaving, the iterations of HDS are on a log scale (x-axis) as compared to HMA and therefore show the smaller denser clusters well. Furthermore, the length of the clusters approximately corresponds to the stability of the clusters, while the relative separation of two points or clusters along the y-axis is proportional to their distance in the original space, since dense points in the same clusters are closer in the dictionary sort; the process of labeling used by HDS results in a novel and amazingly simple projection of high-dimensional data density and clusters onto a 2D space that is at the same time tightly integrated with the clustering methodology.

The Auto-HDS visualization also enables easy visual verification of the cluster selection process. The cluster selection process selects clusters from the first level that they occur. This can be seen in the toy example in Fig. 3b, where the first level of cluster 4 occurs at the fourth column from left; thus, the member points of cluster 4 are $\mathbf{x}_3$, $\mathbf{x}_8$, and $\mathbf{x}_9$. Another example of the cluster selection process with visualization is shown in Figs. 2g and 2h. Fig. 2g shows the relabeled and sorted HDS hierarchy on the Sim-2 data, while Fig. 2h shows the corresponding clusters selected automatically, along with their stability values. Fig. 2i shows the clusters corresponding to Fig. 2h in the original 2D euclidean space. It can be seen that Auto-HDS finds five clusters and a compact hierarchy with only eight nodes and that the results match remarkably well with the actual dense regions and their relative positions in the Sim-2 data.

It is important to note why the hierarchy produced by Auto-HDS (e.g., in the Sim-2 example above) is extremely compact, a key property that distinguishes it from traditional hierarchical clustering methods. The Auto-HDS hierarchy at any level corresponds to only the dense subset of the data; that is, at any given level, the least dense points are assigned to the "don't care" set. Therefore, the number of clusters does not grow rapidly as one moves up or down the Auto-HDS levels; as we move up the levels, new clusters only appear when a cluster splits and old less dense clusters disappear. In contrast, traditional bottom-up hierarchical methods such as Agglomerative clustering often end up discovering numerous spurious clusters. A well-known example in bioinformatics is a program called *Cluster* (http://rana.lbl.gov/EisenSoftware.htm), which uses traditional agglomerative (hierarchical) clustering. When used along with *TreeView* (http://rana.lbl.gov/EisenSoftware.htm), biologists can prune the hierarchy and extract small relevant gene clusters but only through a tedious manual process driven by intuition.

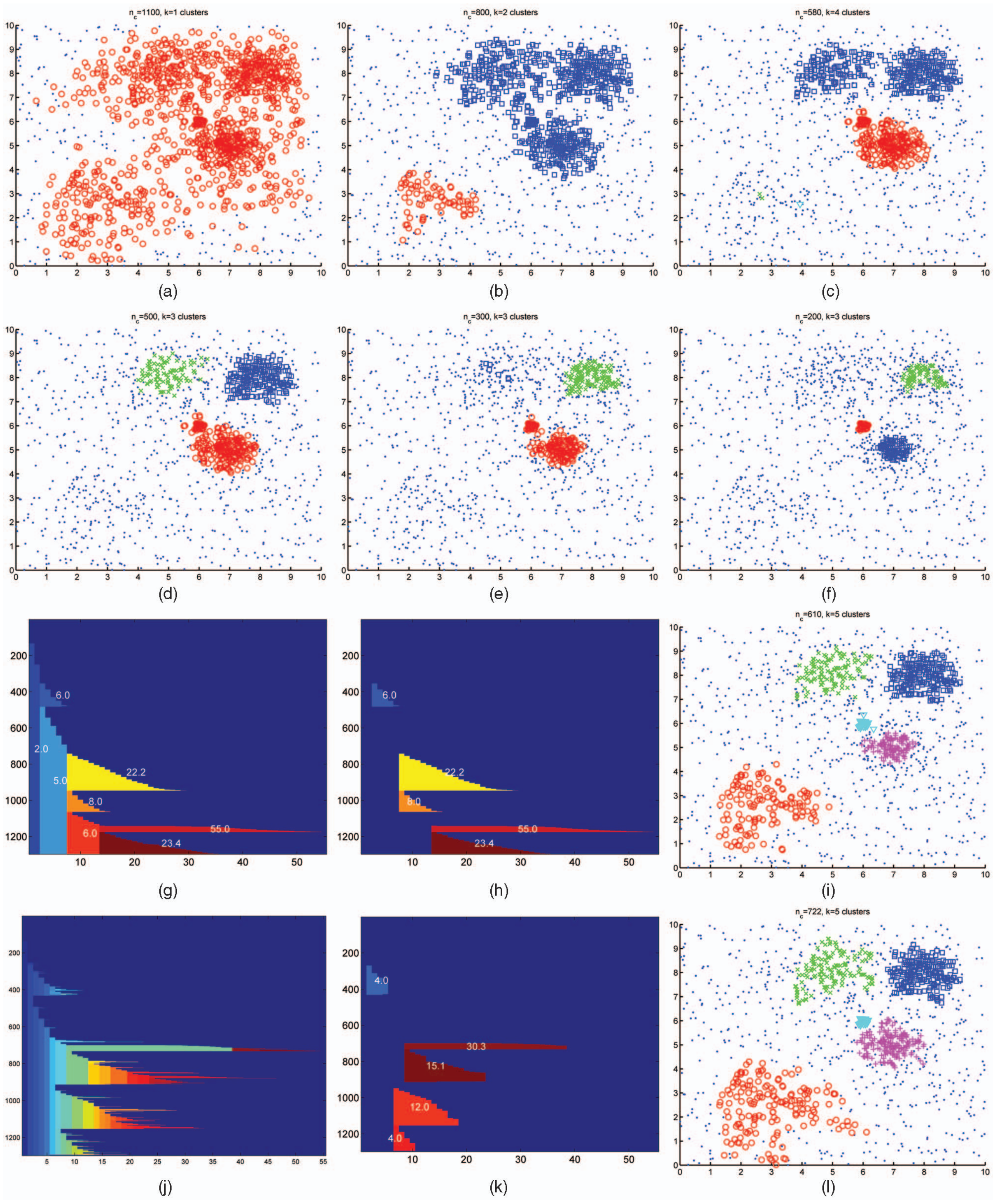


Fig. 2. (a)-(f) Effect of DS applied with varying n_c (f_{shave}) for $n_\epsilon = 20$, resulting in a hierarchy of clusters. For $n_\epsilon = 20$ and $n_{part} = 5$, HDS visualization after (g) cluster identification and (h) cluster selection are shown. Unlike DS, Auto-HDS can discover (i) clusters of different densities. The fourth row shows Auto-HDS results for $n_\epsilon = 7$: (j) shows the degradation of hierarchy compaction with $n_{part} = 0$, and (k) shows hierarchy compaction and cluster selection using $n_{part} = 30$ that result in (l) clusters very similar to those in (i). An r_{shave} of 0.1 was used for HDS. Data set: Sim-2 (see Section 9.1).

Auto-HDS visualization can also aid in choosing the two HDS smoothing parameters n_ϵ and n_{part}. It is possible to prevent small insignificant clusters from being found by selecting either a larger value of n_ϵ or n_{part}. If the user changes n_{part}, Auto-HDS clustering can be updated fairly quickly (usually interactively within seconds), since the hierarchy compaction process is very fast. In contrast, changing n_ϵ requires the regeneration of the HDS clustering. It is

x_i	*Cluster IDs*
$\mathbf{x_1}$	1 1 2 2 0 0 0 0
$\mathbf{x_2}$	1 0 0 0 0 0 0 0
$\mathbf{x_3}$	1 1 3 4 0 0 0 0
$\mathbf{x_4}$	1 0 0 0 0 0 0 0
$\mathbf{x_5}$	1 1 0 0 0 0 0 0
$\mathbf{x_6}$	1 1 3 5 5 0 0 0
$\mathbf{x_7}$	1 1 3 0 0 0 0 0
$\mathbf{x_8}$	1 1 3 4 4 4 0 0
$\mathbf{x_9}$	1 1 3 4 4 4 4 0
$\mathbf{x_{10}}$	1 1 0 0 0 0 0 0

(a)

x_i	*Cluster IDs*
$\mathbf{x_2}$	1 0 0 0 0 0 0 0
$\mathbf{x_4}$	1 0 0 0 0 0 0 0
$\mathbf{x_5}$	1 1 0 0 0 0 0 0
$\mathbf{x_{10}}$	1 1 0 0 0 0 0 0
$\mathbf{x_1}$	1 1 2 2 0 0 0 0
$\mathbf{x_7}$	1 1 3 0 0 0 0 0
$\mathbf{x_3}$	1 1 3 4 0 0 0 0
$\mathbf{x_8}$	1 1 3 4 4 4 0 0
$\mathbf{x_9}$	1 1 3 4 4 4 4 0
$\mathbf{x_6}$	1 1 3 5 5 0 0 0

(b)

Fig. 3. Example of the dictionary sort on $\mathbf{L}_{HDS}$ for $n = 10$, $n_{iter} = 8$, and $n_{part} = 0$. (a) The unsorted label matrix $\mathbf{L}_{HDS}$. (b) The result of the sorting.

therefore possible to choose a very small n_ϵ to start with, obtain a noisy clustering, and then slowly increase n_{part} until the clusters obtained stop changing substantially. Once the clustering appears stable and satisfactory, the user can then use this n_{part} to select a larger n_ϵ and run the HDS clustering from scratch. An example of using this approach for finding good clustering on the Sim-2 data is shown in Figs. 2g, 2h, 2i, 2j, 2k, and 2l, where a small n_ϵ was first used to generate a "noisy" hierarchy (Fig. 2j) and was subsequently smoothed using a larger n_{part} (Fig. 2k). A larger n_ϵ was later found to be sufficient for obtaining a good hierarchy (Fig. 2h). The hierarchy of clusters found using the two alternatives, i.e., a larger n_ϵ versus a larger n_{part}, shows very similar clustering results (Fig. 2i versus Fig. 2l), organized in a very similar topology (Fig. 2h versus Fig. 2k).

To summarize, the visualization provides a powerful compact informative hierarchy and a spatially relevant 2D projection of a high-dimensional density distribution. While many visualization tools are built on top of clustering results, the Auto-HDS visualization directly corresponds to the clustering methodology. The visual feedback provided also allows the user to go back, if desired, and adjust the smoothing parameters n_{part} and n_ϵ. Typically, however, the results are quite stable over a range of smoothing parameter values.

8 Gene DIVER: A Scalable Java Implementation

The Gene DIVER product website is at http://www.ideal.ece.utexas.edu/~gunjan/genediver.

The results presented in this paper were mostly produced using a Recursive HDS implementation in Matlab, which is ideal for rapid prototyping and testing.[6] We also implemented a highly scalable and memory-efficient version of Auto-HDS in Java called *Gene DIVER*. With an $O(n)$ memory usage, Gene DIVER is scalable on modest computers for reasonably large clustering problems and provides a sophisticated SWING-based user interface that not only makes it usable by nonprogrammers but also provides the ability to perform interactive clustering. The Java implementation allows Gene DIVER to run on most platforms. Some of the features in Gene DIVER that make it highly usable for clustering in bioinformatics include

1. low memory usage by processing only one row at a time,
2. ability to reuse previous clustering results on a given data set, resulting in much faster clustering when parameters are updated by the user,
3. allowing the user to skip clustering the least dense data, which again helps in speeding up clustering,
4. ability to select from various distance measures,
5. ability to use a user-supplied distance matrix,
6. allowing a user to interactively browse relevant clusters of genes, including browsing functional categories matching a given gene cluster using FunSpec [37],
7. ability to explore individual genes in a cluster using BioGRID (http://www.thebiogrid.org/), a recently updated and popular database of gene functions for many genomes, and
8. ability to "zoom" the visualization to individual clusters, a feature especially useful for high-throughput biological data sets where individual pure gene clusters are often small, while the number of genes can be large.

Gene DIVER also provides a command-line interface that enables other applications to call Gene DIVER.

The supplemental material (which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2008.32) accompanying this paper provides more details on using Gene DIVER [25]. These include instructions on how to use the Gene DIVER user interface, descriptions of the visualization features, details of the biological databases that can be accessed automatically via Gene DIVER, and more comments on selecting parameters to explore and cluster data sets.

9 Experimental Evaluation

9.1 Data Sets

We tested our framework on two real and one artificial data sets. In Table 2, which summarizes the properties of these data sets, *true* k corresponds to the number of known classes in the data set, while k_A refers to the number of clusters automatically found by Auto-HDS. The Sim-2 data set was generated using five 2D Gaussians of different variances (which roughly correspond to the clusters in Fig. 2i) and a uniform distribution. Two of the Gaussians had relatively small mass, and one of them had very low variance. This data set is useful for verifying algorithms since the true clusters and their spatial distributions are known exactly. The Gasch data set [17], a widely used benchmark for testing clustering algorithms on microarray data, consists of 6,151 genes of yeast *Saccharomyces cervisiae* responding to diverse environmental conditions over 173 microarray experiments. These experiments were designed to measure the response of the yeast strain over various forms of stress such as temperature shock, osmotic shock, starvation, and exposure to various toxins. Each experiment was categorized into one of 12 different categories based on the experiment label. Many of the

6. Recursive HDS is a computationally efficient version of the HDS described in [21].

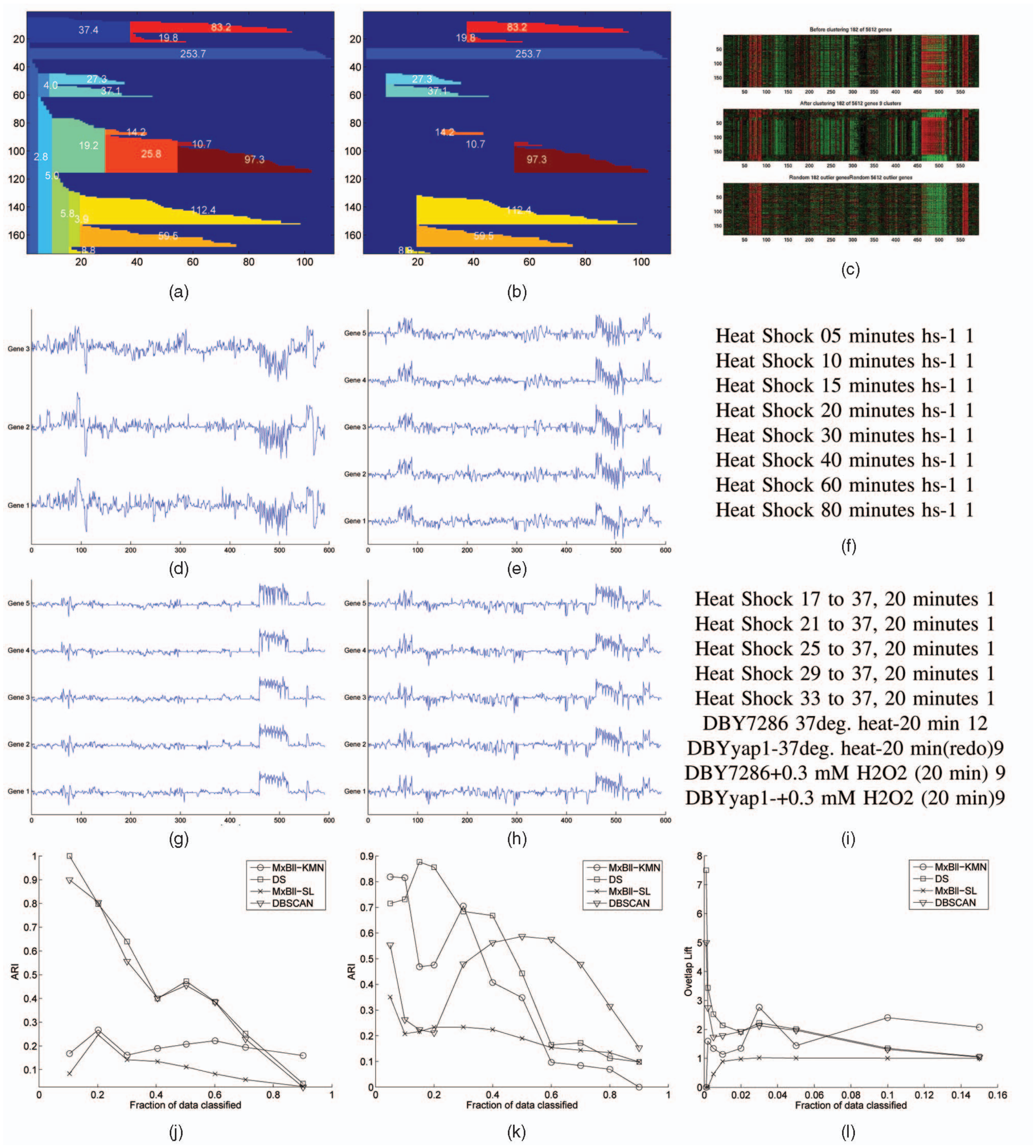


Fig. 4. (a) and (b) Demonstration of Auto-HDS clustering and visualization of Gasch experiments showing the effectiveness of the 2D projection of the 6,151-dimensional Gasch data; related clusters form "siblings" that are located close to each other; their size, density, and stability differences are easy to compare. ARI comparisons of DS with other methods on the (j) Gasch and (k) Sim-2 data sets. (f) and (i) Sibling clusters discovered by Auto-HDS on the Gasch data set. (d),(e), (g), and (h) Expression patterns across 591 experiments (x-axis) for genes of the clusters discovered by Auto-HDS are highly correlated. (c) The same trend is visible across the 182 genes clustered from the Lee data set using $n_\epsilon = 4$, and they are shown using a popular red-green format used for visualizing gene-expression data; red represents underexpressed genes, while green implies overexpressed genes. The x-axis represents the 591 experiments, while the y-axis represents the 182 genes. Within the plot are three subplots; the top and middle plots respectively are shown before and after sorting genes using rows of HDS label matrix $\mathbf{L}$, while the bottom row consists of 182 random genes for comparison. (l) Comparison of DS with other benchmarks on the Lee data set using Overlap Lift. Results for MaxBall K-Means were averaged over 100, 10, and 3 trials for the Sim-2, Gasch, and Lee data sets.

categories such as "temperature," "cold shock," and "heat shock" are closely related. Furthermore, each of the 173 experiments have a description associated with them. The Lee data set [32] consists of 591 gene-expression experiments on yeast obtained from the Stanford Microarray database [18] (http://genome-www5.stanford.edu/) and also contains a *Gold* standard based on Gene Ontology (GO) annotations (http://www.geneontology.org). The

Gold standard contains 121,406 pairwise links (out of a total of 15,744,466 gene pairs) between 5,612 genes in the Lee data that are known to be functionally related. The Gold standard was generated using levels[7] 6-10 of the GO biological process.

9.2 Evaluation Criteria

We use the following three criteria for performing evaluations against labeled data (note that the labels were only used for evaluations; Auto-HDS, DS, and all benchmarks were executed in a completely unsupervised setting):

1. *Adjusted Rand Index* (ARI). This criterion was proposed in [27] as a normalized version of Rand Index and returns 1 for a perfect agreement between clusters and class labels and 0 when the clustering is as bad as random assignments. ARI can be used on the Gasch Array and the simulated Sim-2 data set since the true class labels are available.
2. *P Value.* We use P value to evaluate individual clusters of Yeast genes for the Lee data set. *Funspec*[8] is a popular Yeast database query interface on the Web that computes cluster P values for individual clusters using the hypergeometric distribution, representing the probability that the intersection of a given list of genes with any given functional category occurs by random chance.
3. *Overlap Lift.* The GO annotation Gold standard provides labels for the Lee data set in the form of a set of pairwise links between functionally related genes; one could also view these labels as an undirected graph. It is not possible to use ARI with such a set of labels. Furthermore, the P value described above is only relevant for individual clusters. For evaluating the overall clustering quality on the Lee data set using the GO annotation Gold standard, we can compute the statistical significance of all the clusters simultaneously using *Overlap Lift*, which we introduce as follows: A cluster containing w genes in one cluster creates $w(w-1)/2$ links between genes, since every point within the cluster is linked to every other point. Therefore, k clusters of size $\{w_j\}_{j=1}^{k}$ would result in a total of $l_c = \sum_{j=1}^{k} w_j(w_j-1)/2$ links. The fraction of pairs in the Gold standard that are linked f_{linked} is known (e.g., for the Lee data set, $f_{linked} = 121{,}406/15{,}744{,}466 = 0.007711$). If we construct a null hypothesis as randomly picking l_c pairs out of $n(n-1)/2$ possible pairs, we can expect $l_{null} = f_{linked} l_c$ pairs to be correctly linked. A good clustering should result in more correctly linked pairs than l_{null}. If l_{true} is the number of correct links observed (which will always be $\leq l_c$) in our clustering, then the Overlap Lift is computed as the ratio l_{true}/l_{null}, which represents how many more times correct links are observed as compared to random chance. A larger ratio implies better clustering.

Note that the points in the background or the "don't care" clusters were excluded from the evaluation.

7. Note that the term "level" is used here only in the context of the GO annotation and should not be confused with levels of HDS.
8. http://funspec.med.utoronto.ca/.

9.3 Benchmark Algorithms

Most labeled evaluation measures for clustering are sensitive to the number of clusters discovered and the percentage of data clustered. To get around this problem, we ensure that the benchmark algorithms use the same n_c and k as our methods by applying the following procedure that we call *MaxBall*:

1. For a given benchmark clustering algorithm, find k clusters $\{\mathcal{C}_j\}_{j=1}^{k}$, where k corresponds to the number of clusters found by DS for a particular n_c.
2. Compute cluster center $\mathbf{c}_j$ for cluster $\mathcal{C}_j$ as the mean of the cluster's member points.
3. Assign each of the n points to the cluster center among $\{\mathbf{c}_j\}_{j=1}^{k}$ that is closest[9] to it.
4. Select n_c points closest to their assigned cluster centers as the final clustering. Reassign the remaining $(n-n_c)$ points to the "don't care" set.

Using *MaxBall*, we modified K-Means and Agglomerative clustering as follows:

K-Means. Since the centers output by K-Means are means of the k clusters generated, we can directly apply step 3 of MaxBall to obtain a clustering of n_c points. We refer to the resultant algorithm as *MaxBall K-Means*. Since K-Means uses the Squared euclidean distance, it is not suitable for clustering gene-expression data. However, it is easy to modify K-Means to work with Pearson Distance, a biologically relevant measure ((5) in the Appendix). This modification is similar to that of spherical K-Means [13], except that the recomputed centers are required to be z-scored (just as the points are z-scored in (5) in the Appendix) after each iteration. This modified version of K-Means is used to run experiments on Lee and Gasch data sets.

Agglomerative. One way to obtain k clusters from Agglomerative clustering is to split the cluster dendrogram at a level that gives exactly k clusters. Applying the MaxBall procedure to clusters found using Agglomerative Single Link results in *MaxBall-SL*, while other variants such as Agglomerative Complete Link and Average Link result in *MaxBall-CL* and *MaxBall-AL*, respectively. The performances of average- and complete-link derivatives were comparable to that of the single-link derivate. Therefore, for brevity, we present results on Auto-HDS for Single Link and Complete Link and for DS against Single Link. The MaxBall procedure was also applied on DS in order to compare it with Auto-HDS.

DBSCAN. For the sake of discussion, we define coverage as the fraction of points clustered (i.e., n_c/n). As discussed earlier, the clustering obtained using DBSCAN is similar to that of DS and identical to a special case of HMA described in [44]. However, in contrast with DS, it is not possible to control the coverage directly in DBSCAN, which is essential for a fair comparison against other methods. Therefore, for the two DBSCAN parameters, we set $MinPts$ to four as recommended in [15] and then perform a search for an Eps that results in the desired fraction of data in clusters.

Comparing DS and Auto-HDS with benchmarks. For DS, comparisons with other benchmarks were performed across a range of coverages. Since varying the coverage for

9. Using the same distance measure as that used by DS.

TABLE 3
Comparisons of the Benchmarks with Auto-HDS Using ARI on Gasch and Sim-2 Data

Dataset:	Gasch	Sim-2
Auto-HDS	**0.3509**	**0.6985**
MxBll-DS	0.1533	0.5414
MxBll-KMN	0.2301	0.6433
MxBll-SL	0.1304	0.1948
MxBll-CL	0.2511	0.5114
DBSCAN	0.0234	0.5711

For Gasch data, $k = 11$, and the coverage was 62.4 percent. For Sim-2 data, $k = 5$, and the coverage was 48.4 percent.

DS results in varying k, the corresponding k is used as an input to the benchmark algorithms except for DBSCAN.

For Auto-HDS, which is a deterministic algorithm, it is not possible to vary n_c; the corresponding k and n_c output by Auto-HDS are used along with the MaxBall procedure described earlier to obtain the results summarized in Table 3, except for DBSCAN for which we used the same procedure to control the coverage as that for DBSCAN comparisons with DS. Note that the number of clusters k cannot be controlled for DBSCAN.

9.4 Results

General results. Figs. 4j and 4k compare DS with the other three benchmark algorithms on the Gasch and Sim-2 data set using ARI over a range of fraction of data clustered (x-axis), respectively, while Fig. 4l shows a performance comparison on the Lee data set using Overlap Lift. In general, for lower coverages that correspond to dense regions, DS tended to perform very well. Auto-HDS, which selects clusters automatically, performed better than the other methods for detecting the most significant dense regions in the data, and the discovered k matched well with the number of classes (Table 2). Furthermore, Auto-HDS clusters also matched well with the true labels in the target classes. This can be seen in the highly correlated expression patterns across gene experiments (e.g., Figs. 4d, 4e, 4g, and 4h), when evaluating against known functional categories (Table 4) or class labels (Figs. 4f, 4i, 2i, and 2l and Table 3). It should be stressed that since k is discovered by our framework and is given as an input to the benchmarks (except for DBSCAN where it is not possible to directly control k), they are not a viable alternative to our framework for finding dense regions automatically. Also, DBSCAN tended to oversplit the clusters for the Sim-2 data set, resulting in a much larger number of clusters (between 17 and 48) than the number of classes, which was five.

Robust model parameters. Auto-HDS is also very robust to the choice of the two major model parameters n_ϵ and n_{part} (Fig. 2i versus Fig. 2l). For high-dimensional gene-expression data, usually small values for both work well. Furthermore, r_{shave} is only a speed parameter and does not affect the shape of the HMA hierarchy discovered; smaller values of the shaving rate r_{shave} give slightly higher resolution hierarchies and more precise cluster boundaries. For all experiments, we used r_{shave} in the range of 0.01 and 0.05.

Clustering Gasch experiments. The hierarchy found by Auto-HDS on the extremely high-dimensional Gasch data set is quite compact and easy to interpret (Fig. 4a). Many of the 11 clusters discovered by Auto-HDS (Fig. 4b) contain highly correlated experimental descriptions, while others that form siblings have closely related descriptions. For example, a particularly interesting pair of sibling clusters $\mathcal{A}$ and $\mathcal{B}$ are shown in Figs. 4f and 4i. Both clusters contain heat shock experiments. However, the heat shock experiments in cluster $\mathcal{A}$ involve a constant heat (37 degrees) and variable time, while the heat shock experiments in cluster $\mathcal{B}$ involve variable heat and constant time. Additional such examples can be found at our website.

TABLE 4
Example High-Purity Clusters from Auto-HDS, Lee Data Set

C. Id.	$\lvert\mathcal{C}\rvert$	Cat(Cov.)	p-val
2	8	Nucleosomal protein complex (8/8)	<1e-14
3	7	PF00674-DUP (6/7)	1.132e-14
5	11	glycolysis (7/16)	3.175e-14
5	11	cytoplasm (11/554)	3.175e-14
6	120	Cytoplasmic ribosomes (111/138)	<1e-14
7	7	PF00660-SRP1_TIP1 (6/30)	<1e-14
7	7	stress (5/175)	<1e-14

Note that Funspec returns multiple categories (column 3) with low P values for a given cluster; (x/y) stands for the coverage, where we found x out of y known members of a category.

Clustering genes, Lee. Automating clustering for biological data sets in general, and gene-expression data sets in particular, is a hard problem that motivated the design of Auto-HDS. One of the critical issues facing biologists analyzing the vast stream of biological data is that obtaining pure gene clusters often requires significant manual pruning of the clustering results. With $n_\epsilon = 4$ and $n_{part} = 2$, Auto-HDS found nine clusters in the Lee data set formed by 182 out of 5,612 genes. After pruning such vast numbers of genes, most of the clusters were very pure when evaluated using FunSpec and show very small P values. Some of the high-purity clusters with extremely low P values are summarized in Table 4. More details on these clusters are available online on our website. The most surprising among these clusters is Cluster 6, where 111 out of the 120 genes in the cluster belong to a known biological process category —*Cytoplasmic ribosomes* that has only 138 known members. Given that there are 5,612 genes to pick from, this accuracy is remarkable.

Another popular approach for quickly verifying the quality of gene-expression clustering is by visualizing the clustered genes in the original feature space. For four of the clusters in Table 4, Figs. 4d, 4e, 4g, and 4h show the gene-expression level of a sample of genes from the corresponding cluster across 591 experiments in the Lee data set. Clearly, the genes are highly correlated. A high degree of correlation is also visible in the middle subplot in Fig. 4c, where all the 182 genes were sorted using the discovered HDS hierarchy.

9.5 Additional Results

Expanded results from the above set of experiments are available online.[10] Additional experiments have been performed by other researchers on other types of biological data using the Gene DIVER software. Work by one of the authors and a colleague on clustering phylogenetic data is

10. http://www.ideal.ece.utexas.edu/~gunjan/hds/readme.html.

included in the supplemental material (which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2008.32) [36]. The goal of the experiments is to identify coconserved functional gene groupings by clustering phylogenetic profiles of 20 different species of yeast. Similar to the results presented above, the results on this data set indicate that Auto-HDS can find compact pure clusters that can be easily explored and potentially utilized for predicting new functions for less understood genes. In comparison, traditional methods that do not prune out noise are less easily interpretable.

In addition, [7] utilizes Gene DIVER to construct a 3D map of human and mouse genomes. The authors use Gene DIVER to identify salient regions in the map and to validate the preservation of these regions in the map across the two species.

10 Possible Extensions to Auto-HDS

One extension of traditional clustering algorithms is semisupervised clustering, which is able to make use of various kinds of labeled information. This information may be in the form of class labels or various kinds of constraints (e.g., must-link constraints, which list points that must be in the same cluster) [6]. A semisupervised Auto-HDS is possible.

The selection of most stable clusters in Auto-HDS is currently constrained to avoid selecting a parent and a child cluster simultaneously. This strategy maximizes the number of most stable *and* distinct clusters discovered and, when clustering genes, is well suited for selecting all of the most specific functional groupings of genes identified in the data. However, other equally valid selection approaches exist that would be better suited for other cluster mining goals. For example, if the goal is to find the most stable clusters *and* their topology, then performing an unconstrained selection of the k most stable clusters would result in a set of clusters that are related to each other in a hierarchical manner—a concept that is exploited in [7].

Finding overlapping gene clusters is also of great interest to biologists. Though the splitting criteria used in Auto-HDS results in nonoverlapping clusters, it is possible to *extend* the final set of selected clusters by selecting a splitting point from an earlier HDS iteration, resulting in overlapping clusters. We are currently working toward incorporating a user-interactive version of this process into Gene DIVER. Other features in the works include the ability to automatically label clusters with known gene function annotations and the ability to cluster and visualize gene network data.

Finally, there is a connection between Auto-HDS (and density-based clustering in general) and outlier detection algorithms that seek to find rarely occurring events. In particular, the local outlier factor [8] is an outlier detection technique that uses a notion of density to define outliers. While density-based clustering and outlier detection are not the same problem, it is worth investigating whether one can adapt techniques used in Auto-HDS (e.g., visualization) for outlier detection problems.

11 Concluding Remarks

In this paper, we have introduced Auto-HDS, a framework that is well suited for unsupervised learning on large data sets such as those in bioinformatics, where the solutions of interest are often hidden in a small subset of the data. A key property of our framework is the ability to find a compact hierarchy of clusters. This stems from the ability to ignore less dense points while generating the hierarchy. Auto-HDS is also able to automatically discover the size, the number, and the location of dense clusters. In particular, remarkably pure clusters of functionally related genes have been discovered by Auto-HDS from both microarray data and phylogenetic profile data.

The Gene DIVER product, a highly scalable and memory-efficient implementation of Auto-HDS, includes several key innovations that make interactive clustering of large biological data sets practical on modest computers. This tool is already being used by other researchers, and we hope that it will eventually become a popular and versatile clustering tool for bioinformatics applications.

Appendix

Pearson Distance for Biological Data Sets

An example of a symmetric distance measure is *Pearson Distance* (d_p) [20] computed as $(1 - p)$, where p is the *Pearson Correlation*, a popular similarity measure for clustering gene-expression and other biological data [39], [34]. It can be shown that Pearson Distance is equal to the Squared euclidean distance between z-scored[11] points normalized by $2(d - 1)$:

$$d_p(\mathbf{x}, \mathbf{y}) = \frac{\|z(\mathbf{x}) - z(\mathbf{y})\|^2}{2(d - 1)}, \tag{5}$$

where z represents the z-scoring function. Pearson Correlation is popular among biologists for clustering genes since it captures correlation that is invariant to linear scaling; it is useful for a variety of high-throughput biological data sets such as gene expression, protein expression, phylogenetic profiles, and protein mass spectroscopy, among others. We use the Pearson Distance on the biological data presented in this paper. In particular, it is reasonable to do so with HMA (and for our Auto-HDS framework derived from HMA) for the following reasons: 1) the triangle inequality property is exploited indirectly in the graph-traversal process that connects the clusters in HMA, 2) $\sqrt{d_p}$ gives a semimetric that is the euclidean distance between z-scored points, and 3) the clustering using $\sqrt{d_p}$ and d_p would be identical for the density kernel used in HMA (1) since the relative ordering between points is the same for $\sqrt{d_p}$ and d_p. For the same reasons, it can be shown that $(1 - \text{cosine}$ similarity) [13] would also be an appropriate distance measure with HMA and Auto-HDS.

Acknowledgments

This research was supported by US National Science Foundation Grants IIS-0325116, IIS-0307792, and IIS-0713142. The authors are grateful to Edward Marcotte, Insuk Lee, and Kriston L. McGary (http://polaris.icmb.utexas.edu) for providing many valuable insights.

11. Normally performed between points across a dimension. Here, we perform it between dimensions for each data point.

REFERENCES

[1] M. Ankerst, M.M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering Points to Identify the Clustering Structure," *Proc. ACM SIGMOD '99,* pp. 49-60, 1999.

[2] A. Banerjee, S. Basu, C. Krumpelman, J. Ghosh, and R. Mooney, "Model Based Overlapping Clustering," *Proc. ACM SIGKDD '05,* pp. 100-106, 2005.

[3] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. Modha, "A Generalized Maximum Entropy to Bregman Co-Clustering and Matrix Approximation," *J. Machine Learning Research,* vol. 8.

[4] A. Banerjee, I. Dhillon, J. Ghosh, and S. Sra, "Clustering on the Unit Hypersphere Using Von Mises-Fisher Distributions," *J. Machine Learning Research,* vol. 6, pp. 1345-1382, 2005.

[5] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh, "Clustering with Bregman Divergences," *J. Machine Learning Research,* vol. 6, pp. 1705-1749, 2005.

[6] S. Basu, A. Banerjee, and R.J. Mooney, "Semi-Supervised Clustering by Seeding," *Proc. 19th Int'l Conf. Machine Learning (ICML '02),* pp. 27-34, 2002.

[7] M. Bellis and J. Hennetin, "Application of Gene DIVER to the Study of Geometrical Representations of Gene Expression Covariation," *IEEE/ACM Transaction Computational Biology and Bioinformatics,* supplement 3, 2008.

[8] M.M. Breunig, H.-P. Kriegel, R.T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," *Proc. ACM SIGMOD '00,* pp. 93-104, 2000.

[9] S.V. Chakaravathy and J. Ghosh, "Scale Based Clustering Using a Radial Basis Function Network," *IEEE Trans. Neural Networks,* vol. 2, no. 5, pp. 1250-1261, Sept. 1996.

[10] H. Cho, I.S. Dhillon, Y. Guan, and S. Sra, "Minimum Sum-Squared Residue Co-Clustering of Gene Expression Data," *Proc. Fourth SIAM Int'l Conf. Data Mining (SDM '04),* pp. 114-125, Apr. 2004.

[11] I. Dhillon, S. Mallela, and D. Modha, "Information-Theoretic Co-Clustering," *Proc. ACM SIGKDD '03,* pp. 89-98, 2003.

[12] I.S. Dhillon, E.M. Marcotte, and U. Roshan, "Diametrical Clustering for Identifying Anti-Correlated Gene Clusters," *Bioinformatics,* vol. 19, pp. 1612-1619, 2003.

[13] I.S. Dhillon and D.S. Modha, "Concept Decompositions for Large Sparse Text Data Using Clustering," *Machine Learning,* vol. 42, no. 1-2, pp. 143-175, Jan.-Feb. 2001.

[14] A. Enright, S. Van Dongen, and C. Ouzounis, "An Efficient Algorithm for Large-Scale Detection of Protein Families," *Nucleic Acids Research,* vol. 30, no. 7, pp. 1575-1584, 2002.

[15] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proc. ACM SIGKDD '96,* pp. 226-231, 1996.

[16] B. Everitt, *Cluster Analysis.* Heinemann Educational Books, 1974.

[17] A.P. Gasch et al., "Genomic Expression Programs in the Response of Yeast Cells to Environmental Changes," *Molecular Biology of the Cell,* vol. 11, no. 3, pp. 4241-4257, Dec. 2000.

[18] J. Gollub et al., "The Stanford Microarray Database: Data Access and Quality Assessment Tools," *Nucleic Acids Research,* vol. 31, pp. 94-96, 2003.

[19] G. Gupta, "Robust Methods for Locating Multiple Dense Regions in Complex Datasets," PhD dissertation, Univ. of Texas at Austin, Dec. 2006.

[20] G. Gupta and J. Ghosh, "Robust One-Class Clustering Using Hybrid Global and Local Search," *Proc. 22nd Int'l Conf. Machine Learning (ICML '05),* pp. 273-280, Aug. 2005.

[21] G. Gupta, A. Liu, and J. Ghosh, "Automatic Hierarchical Density Shaving and Gene DIVER," Technical Report IDEAL-TR05, Dept. Electrical and Computer Eng., Univ. of Texas at Austin, http://www.lans.ece.utexas.edu/techreps.html, 2006.

[22] G. Gupta, A. Liu, and J. Ghosh, "Clustering and Visualization of High-Dimensional Biological Datasets Using a Fast HMA Approximation," *Proc. Artificial Neural Networks in Eng. Conf. (ANNIE '06),* Nov. 2006.

[23] G. Gupta, A. Liu, and J. Ghosh, "Hierarchical Density Shaving: A Clustering and Visualization Framework for Large Biological Datasets," *Proc. IEEE ICDM Workshop Data Mining in Bioinformatics (DMB '06),* pp. 89-93, Dec. 2006.

[24] G. Gupta, A. Liu, and J. Ghosh, "An Extended Example of Creating L_{HDS}," *IEEE/ACM Trans. Computational Biology and Bioinformatics,* supplement 4, 2008.

[25] G. Gupta, A. Liu, and J. Ghosh, "Gene DIVER: Gene Density Interactive Visual ExploreR," *IEEE/ACM Trans. Computational Biology and Bioinformatics,* supplement 1, 2008.

[26] T. Hastie et al., "Gene Shaving as a Method for Identifying Distinct Sets of Genes with Similar Expression Patterns," *Genome Biology,* vol. 1, pp. 1-21, 2000.

[27] L. Hubert and P. Arabie, "Comparing Partitions," *J. Classification,* pp. 193-218, 1985.

[28] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data.* Prentice Hall, 1988.

[29] H. Jenq-Neng, L. Shyh-Rong, and A. Lippman, "Nonparametric Multivariate Density Estimation: A Comparative Study," *Science,* vol. 42, no. 10, pp. 2795-2810, Oct. 1994.

[30] D. Jiang, J. Pei, and A. Zhang, "DHC: A Density-Based Hierarchical Clustering Method for Time Series Gene Expression Data," *Proc. Third IEEE Int'l Symp. BioInformatics and BioEngineering (BIBE '03),* p. 393, 2003.

[31] L. Lazzeroni and A.B. Owen, "Plaid Models for Gene Expression Data," *Statistica Sinica,* vol. 12, no. 1, pp. 61-86, Jan. 2002.

[32] I. Lee, S.V. Date, A.T. Adai, and E.M. Marcotte, "A Probabilistic Functional Network of Yeast Genes," *Science,* vol. 306, pp. 1555-1558, 2004.

[33] S.C. Madeira and A.L. Oliveira, "Biclustering Algorithms for Biological Data Analysis: A Survey," *IEEE/ACM Trans. Computational Biology and Bioinformatics,* vol. 1, no. 1, pp. 24-45, Jan.-Mar. 2004.

[34] R. Mansson, P. Tsapogas, M.A. et al., "Pearson Correlation Analysis of Microarray Data Allows for the Identification of Genetic Targets for Early B-Cell Factor," *J. Biological Chemistry,* vol. 279, no. 17, pp. 17905-17913, Apr. 2004.

[35] E.M. Marcotte, I. Xenarios, A.M. van Der Bliek, and D. Eisenberg, "Localizing Proteins in the Cell from Their Phylogenetic Profiles," *Proc. Nat'l Academy of Sciences USA,* vol. 97, no. 22, pp. 12115-12120, Oct. 2000.

[36] K.L. McGary and G. Gupta, "Discovering Functionally Related Genes in Yeast Using Gene DIVER on Phylogenetic Profile Data," *IEEE/ACM Trans. Computational Biology and Bioinformatics,* supplement 2, 2008.

[37] M.D. Robinson, J. Grigull, N. Mohammad, and T.R. Hughes, "FunSpec: A Web-Based Cluster Interpreter for Yeast," *BMC Bioinformatics,* vol. 35, no. 3, Nov. 2002.

[38] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller, "Rich Probabilistic Models for Gene Expression," *Bioinformatics,* vol. 17, no. 1, pp. 243-252, 2003.

[39] R. Sharan and R. Shamir, "Click: A Clustering Algorithm with Applications to Gene Expression Analysis," *Proc. Eighth Int'l Conf. Intelligent Systems for Molecular Biology (ISMB '00),* pp. 307-316, 2000.

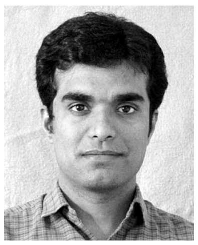
[40] D. Stoll, J. Bachmann, M.F. Templin, and T.O. Joos, "Microarray Technology: An Increasing Variety of Screening Tools for Proteomic Research," *Drug Discovery Today: TARGETS,* vol. 3, no. 1, pp. 24-31, Feb. 2004.

[41] A. Strehl and J. Ghosh, "Relationship-Based Clustering and Visualization for High-Dimensional Data Mining," *INFORMS J. Computing,* vol. 15, no. 2, pp. 208-230, 2003.

[42] W. Stuetzle, "Estimating the Cluster Tree of a Density by Analyzing the Minimal Spanning Tree of a Sample," *J. Classification,* vol. 20, pp. 25-47, 2003.

[43] J.B. Tenenbaum, V. de Silva, and J.C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science,* vol. 290, pp. 2319-2323, 2000.

[44] D. Wishart, "Mode Analysis: A Generalization of Nearest Neighbour Which Reduces Chaining Effects," *Proc. Colloquium Numerical Taxonomy,* pp. 282-308, Sept. 1968.



Gunjan Gupta received the PhD degree from the Department of Electrical and Computer Engineering, University of Texas at Austin, in 2006. His research interests include data mining and learning in situations where the data is large, high-dimensional, and noisy. He has worked as a data miner for over a decade at several companies, including Standard and Poors, i2 Technologies, and Infosys Technologies. He is currently a machine learning scientist at Amazon.com, Seattle, Wash. He is a member of the IEEE.

Alexander Liu received the BS, MS, and PhD degrees in electrical and computer engineering from the University of Texas at Austin in 2001, 2004, and 2009, respectively. He has been supported by a number of fellowships, including the Texas Excellence Award for Scholarship and Leadership, a Microelectronics and Computer Development Fellowship, and the Thrust Fellowship. His research interests include data mining, particularly cost-sensitive learning.

Joydeep Ghosh received the BTech degree from the Indian Institute of Technology (IIT), Kanpur, in 1983 and the PhD degree from the University of Southern California in 1988). He joined the faculty of the University of Texas at Austin (UT-Austin) in 1988, where he is currently the Schlumberger centennial chair professor of electrical and computer engineering in the Department of Electrical and Computer Engineering. He is the founder-director of the Intelligent Data Exploration and Analysis Laboratory (IDEAL). He has authored or coauthored more than 200 refereed papers, including more than 50 full-length journal articles. He has received 10 best paper awards, including the 2005 UT-Coop Society's Best Research Paper across all departments, the Best Theory Paper at SDM 04, the Best Applications Paper at ANNIE '97, and the 1992 Darlington Award for best paper among all IEEE CAS publications. He is a program cochair for 2006 SIAM International Conference on Data Mining, and the founding chair of the IEEE Computer Intelligence Society's Technical Committee on Data Mining. He is a fellow of the IEEE.