DHC: A Density-based Hierarchical Clustering Method for Time Series Gene Expression Data

Daxin Jiang Jian Pei Aidong Zhang
Department of Computer Science and Engineering, State University of New York at Buffalo
Email: {djiang3, jianpei, azhang}@cse.buffalo.edu

Abstract

Clustering the time series gene expression data is an important task in bioinformatics research and biomedical applications. Recently, some clustering methods have been adapted or proposed. However, some concerns still remain, such as the robustness of the mining methods, as well as the quality and the interpretability of the mining results.

In this paper, we tackle the problem of *effectively clustering time series gene expression data* by proposing algorithm *DHC*, a *density-based, hierarchical clustering* method. We use a density-based approach to identify the clusters such that the clustering results are of high quality and robustness. Moreover, The mining result is in the form of a *density tree*, which uncovers the embedded clusters in a data set. The inner-structures, the borders and the outliers of the clusters can be further investigated using the *attraction tree*, which is an intermediate result of the mining. By these two trees, the internal structure of the data set can be visualized effectively. Our empirical evaluation using some real-world data sets show that the method is effective, robust and scalable. It matches the ground truth provided by bioinformatics experts very well in the sample data sets.

1 Introduction

DNA microarray technology [11, 12] has made it now possible to monitor simultaneously the expression levels for thousands of genes during important biological process [15] and across collections of related samples [1]. It is often an important task to find genes with similar expression patterns (co-expressed genes) from DNA microarray data. First, co-expressed genes may demonstrate a significant enrichment for function analysis of the genes [2, 17, 6, 13]. We may understand the functions of some poorly characterized or novel genes better by testing them together with the genes with known functions. Second, co-expressed genes with strong expression pattern correlations may indicate co-regulation and help uncover the regulatory elements in transcriptional regulatory networks [17]. Cluster techniques, which are essential in data mining process for exploring natural structure and identifying interesting patterns in underlying data, have proved to be useful in finding co-expressed genes.

In cluster analysis, one wishes to partition the given data set into groups based on the given features such that the data objects in the same group are more similar to each other than the data objects in other groups. Various clustering algorithms have been applied on gene expression data with promising results [6, 17, 16, 4, 13]. However, as indicated in some previous studies (e.g., [8, 16]), many conventional clustering algorithms originated from non-biological fields may suffer from some problems when mining gene expression data, such as having to specify the number of clusters, lacking of robustness to noises, and being weak to handle embedded clusters and highly intersected clusters. Recently, some specifically designed algorithms for clustering gene expression data have been proposed aiming at those problems [8, 4].

Distinguishing from other kinds of data, gene expression data usually have several characteristics. First, gene expression data sets are often of small size (in the scale of 10 thousands) comparing to some other large databases (e.g. multimedia databases and transaction databases). A gene expression data set often can be held into main memory. Second, for many dedicated microarray experiments, people are usually interested in the expression patterns of only a subset of all the genes. Other gene patterns are roughly considered insignificant, and thus become noise. Hence, in the gene expression data analysis, people are much more concerned with the effectiveness and interpretability of the clustering results than the efficiency of the clustering algorithm. How to group co-expressed genes together meaningfully and extract the useful patterns intelligently from noisy data sets are two major challenges for clustering gene expression data.

In this paper, we investigate the problems of *effectively clustering gene expression data* and make the following contributions. First, we analyze and examine a good number of existing clustering algorithms in the context of clustering gene expression data, and clearly identify the challenges. Second, we develop DHC, a *density-based, hierarchical clustering* method aiming at gene expression data. DHC is a density-based approach so that it effectively solves some problems that most distance-based approaches

cannot handle. Moreover, DHC is a hierarchical method. The mining result is in the form of a tree of clusters. The internal structure of the data set can be visualized effectively. At last, we conduct an extensive performance study on DHC and some related methods. Our experimental results show that DHC is effective. The mining results match the ground truth given by the bioinformatics experts nicely on real data sets. Moreover, DHC is robust with respect to noise and scalable with respect to database size.

The remainder of the paper is organized as follows. In Section 2, we analyze some important existing clustering methods in the context of clustering gene expression data, and identify the challenges. In Section 3, we discuss the density measurement for density-based clustering of gene expression, and develop algorithm DHC. The extensive experimental results are reported in Section 4. The paper is concluded in Section 5.

2 Related Work

Various clustering algorithms have been applied to gene expression data. It has been proved that the clustering is helpful to identify groups of co-expressed genes and corresponding expression patterns. Nevertheless, several challenges arise. In this section, we identify the challenges by a brief survey of some typical clustering methods.

2.1 Partition-based Algorithms

K-means [17] and SOM (Self Organizing Map) [16] are two typical partition-based clustering algorithms. Although useful, these algorithms suffer the following drawbacks as pointed out by [8, 14]. First, both K-means and SOM require the users to provide the number of clusters as a parameter. Since clustering is usually an explorative task in the initial analysis of gene expression data sets, such information is often unavailable. Another disadvantage of the partition-based approaches is that they force each data object into a cluster, which makes the partition-based approaches sensitive to outliers.

Recently, some new algorithms have been developed for clustering time-series gene expression data. They particularly addressed the problems of outliers and the number of clusters discussed above. For example, Ben-Dor et al. [4] introduced the idea of a "corrupted clique graph" data model and presented a heuristic algorithm CAST (for Cluster Affinity Search Technique) based on the data model. In [8], the authors described a two-step procedure (Adapt) to identify one cluster while the first step is to estimate the cluster center C_K and the second step is to estimate the radius R_K of the cluster. Once C_K and R_K are determined, a cluster is defined as $Cluster = \{g_i \in G | \|g_i - C_K\| < R_K\}$. Both algorithms extract clusters from the data set one after another until no more clusters can be found. Therefore, the algorithms can automatically determine the number of clusters, and genes not belonging to any cluster are regarded as outliers. However, the criteron for clusters defined by those algorithms are either based on some glabal parameters or dependent on some assumptions of the cluster structure of the data set. For example, CAST uses an *affinity threshold* parameter t to control the avaerage pairwise similarity between objects within a cluster. Adapt assumes that the cluster has the same radius in each direction in the high-dimensional object space. However, the clustering results may be quite sensitive to differenct parameter settings and the assumptions for cluster structure may not always hold. In particular, CAST and Adapt may not be effective with the embedded clusters and the highly intersected clusters, respectively.

2.2 Hierarchical Clustering

A hierarchical clustering algorithm does not generate a set of disjoint clusters. Instead, it generates a hierarchy of nested clusters that can be represented by a tree, called a *dendrogram*. Based on how the hierarchical decomposition is formed, hierarchical clustering algorithms can be further divided into *agglomerative algorithms* (i.e., bottom-up approaches, e.g., [6]) and divisive algorithms (top-down approaches, e.g., [2, 13]). In fact, the hierarchical methods are particularly favored by the biologists becasue they may give more insights to the structure of the clusters than the other methods.

However, the hierarchical methods also have some drawbacks. First, it is sometimes subtle to determine where to cut the dendrogram and derive clusters. Usually, this step is done by domain experts' visual inspection. Second, it is hard to tell the inner structure of a cluster from the dendrogram, e.g. which object is the medoid of the cluster and which objects are the borders of the cluster. Last, many hierarchical methods are considered lacking of robustness and uniqueness [16]. They may be sensitive to the order of input and small perturbations in the data.

2.3 Density-based clustering

Density-based clustering algorithms [7, 9, 3]characterize the data distribution by the *density* of each data object. Clustering is the process of identifying dense areas in the object space. Coventional density-based approaches, such as DBSCAN [7], classify a data object O as one of the *cores* of a cluster if O has more than $MinPts$ neighbors within neighborhood τ. Clusters are formed by connecting neighboring 'core' objects and those 'non-core' objects either serve as the boundaies of clusters or become outliers. Since the noises of the data set are typically randomly distributed, the density within a cluster should be significantly higher than that of the noises. Therefore, density-based approaches have the advantage of extracting clusters from a highly noisy environment, which is the case of time-series gene expression data.

However, the performance of DBSCAN is quite sensitive to the parameters of object density, namely, $MinPts$

and τ. For a complex data set, the appropriate parameters are hard to specify. Our experimental study has demonstrated that DBSCAN tends to reuslt in either a large number of trivial clusters or a few huge clusters merged by several smaller ones for time-series gene expression data. Other density-based approaches (e.g. Optics [3] and Denclue [9]) are more robust to their algorithm parameters. However, none of the exsited density-based algorithms provide a hierarchial cluster strucuture which gives people a thorough picture of the data distribution and help people understand the relationship between the clusters and the data objects well.

2.4 What are the challenges?

Based on the above analysis, to conduct effective clustering analysis over gene expression data, we need to develop an algorithm meeting the following requirements.

First, the clustering result should be highly interpretable and easy to visualize. As gene expression data is complicated, the interpretability and visualization become very important. Second, the clustering method should be able to determine the number of clusters automatically. Gene expression data sets are usually noisy. It is hard to guess the number of clusters. A method adaptive to the natural number of clusters is highly preferable. Third, the clustering method should be robust to noise, outliers, and the parameters. It is well recognized that the gene expression data are usually noisy and the rules behind the data are unknown. Thus, the method should be robust so that it can be used as the first step to explore the valuable patterns. Lastly but not at least, the clustering method should be able to handle embedded clusters and highly intersected clusters effectively. The structure of gene expression data is often complicated. It is unlikely that the data space can be clearly divided into several independent clusters. Instead, the user may be interested in both the clusters and the connections among the clusters. The clustering method should be able to uncover the whole picture.

3 The Algorithms

In this section, we propose *DHC*, an algorithm mining the cluster structure of a data set as a density tree. First, we dicuss how to define the density of objects properly and then we develop the algorithm. The algorithm works in two steps. First, all objects in a data set are organized into an *attraction tree*. Then, the attraction tree is summarized as clusters and dense areas, and a *density tree* is derived as the summary structure.

3.1 Definition of density

When clustering gene expression data, we want to group genes with similar expression patterns. Thus, we choose the correlation coefficient, which is capable of catching the similarity between two vectors based on their expression patterns but not on the absolute magnitudes. The correlation coefficient for two data objects O_i and O_j in a d-dimension space is defined as

$$similarity(O_i, O_j) = \rho(O_i, O_j) = \frac{\sum_{l=1}^{d}(O_{il}-\bar{O}_i)(O_{jl}-\bar{O}_j)}{\sqrt{\sum_{l=1}^{d}(O_{il}-\bar{O}_i)^2}\sqrt{\sum_{l=1}^{d}(O_{jl}-\bar{O}_j)^2}}.$$

where O_{il} is the l^{th} scalar of data object O_i and $\bar{O}_i$ is the mean of the scalars of data object O_i. Note that the correlation coefficient ρ ranges between -1 and 1. The larger the value, the more similar they are with each other.

We define the distance between objects O_i and O_j as

$$d(O_i, O_j) = \begin{cases} \frac{1}{\rho(O_i,O_j)} & \text{if } \rho(O_i, O_j) > 0 \\ +\infty & \text{otherwise} \end{cases}.$$

Given a radius r, the neighborhood of O_i w.r.t. r in a d-dimension space forms a hyper-sphere $\mathcal{S}_r^{O_i}$. The set of objects in the hyper-sphere is $\{O_j | d(O_i, O_j) \leq r\}$. The volume of the hyper-sphere is $V(\mathcal{S}_r^{O_i}) = \frac{\pi^{\frac{d}{2}}}{\Gamma[\frac{n}{2}+1]} \cdot r^d$. We ignore the global constant coefficient $\frac{\pi^{\frac{d}{2}}}{\Gamma[\frac{n}{2}+1]}$ and define $Vol(\mathcal{S}_r^{O_i}) = r^d$.

To precisely describe the distribution of neighbors of object O_i, we divide $\mathcal{S}_r^{O_i}$ into a series of *hyper-shells*, such that each hyper-shell occupies exactly a *unit* volume. The idea is demonstrated in Figure 1.

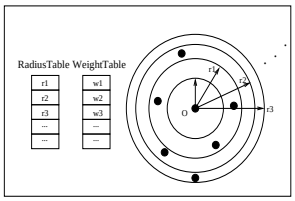


Figure 1: Hypersphere and hyper-shell w.r.t object O_i

The radius of the k^{th} hyper-shell is

$$r_k = k^{\frac{1}{d}} \qquad (Volume(\mathcal{S}_{r_k}^{O_i}) = k, \quad k = 1, 2, \cdots).$$

Then, we have

$$Shell_k^{O_i} = \mathcal{S}_{k+1}^{O_i} - \mathcal{S}_k^{O_i} = \{O_j | r_k < d(O_i, O_j) \leq r_{k+1}\}.$$

By hyper-shells, the neighborhood of O_i is discretized and forms a histogram $\mathcal{H}$, where each bin β_k contains the objects falling into hyper-shell $Shell_k$. For each β_k, we define a *weight* of the contribution from β_k to the density of O_i.

$$\beta_k = \|Shell_k^{O_i}\|, \qquad Weight(Shell_k^{O_i}) = \frac{1}{\log k}.$$

Now, we are ready to define the density of an object O.

$$density(O) = \sum_{k=1}^{\infty} Weight(Shell_k^O) \cdot \beta_k.$$

In our density definition, we do not simply set up a threshold τ and count the number of data objects within neighborhood τ as the density. On the contrary, we discretize the neighborhood of object O_i into a series of hyper-shells and calculate the density of an object as the sum of contributions from individual hyper-shells. Thus, our definition avoids the difficulty of choosing a good global threshold τ and accurately reflect the neighbor distribution around the specific object.

3.2 Building an attraction tree

As the first step of the mining, we organize all objects in the data set into a hierarchy based on their density. The resulting structure is called an *attraction tree*, since the tree is built by considering the attraction among objects in a data set. Intuitively, an object with high density "*attracts*" some other objects with lower density. We formalize the ideas as follows.

The attraction between two data objects O_1 and O_2 ($O_1 \neq O_2$) in a k-d space is defined as follows, where k is the dimensionality of the objects.

$$Attraction(O_1, O_2) = \frac{density(O_1) \cdot density(O_2)}{d(O_1, O_2)^{k-1}}.$$

The attraction is said *from O_i to O_j*, if $density(O_i) < density(O_j)$, denoted as $O_i \rightarrow O_j$. In the case that two objects are tie, we can artificially assign $O_i \rightarrow O_j$ for $(i < j)$. Thus, an object O is attracted by a set of objects $A(O)$ whose densities are larger than that of O, i.e., $A(O) = \{O_j | density(O_j) > density(O)\}$. We define the *attractor* of O as the object $O_j \in A(O)$ with the largest attraction to O, i.e.,

$$Attractor(O) = \arg \max_{O_j \in A(O)} Attraction(O_j, O).$$

The process of determining the attractor of each data object is as follows. For each data object O_i, we initialize O_i's attractor as itself. Then we search for $A(O_i)$ and compare the attraction for each $O_j \in A(O_i)$ to O_i. Finally the winner becomes the attractor of O_i. The only special case is O_{hd} that has the largest density, where $A(O_{hd})$ will be empty. In this case, we set O_{hd}'s attractor as itself.

The attraction relation from an object to another (i.e., $O_i \rightarrow O_j$) is a partial order. Based on this order, we can derive an *attraction tree* T. Each node has an object O such that

$$Parent(O) = \begin{cases} nil & \text{if } Attractor(O) = O, \\ attractor(O) & \text{otherwise.} \end{cases}$$

The tree construction process is as follows. First, each data object O_i is a singleton attraction tree T_i. Then, we scan the data set once. For each data object O_i, we find its attractor O_j. We insert T_i as a child of O_j. Thus the original singleton cluster trees merge with the others during the scanning process. When the scanning process is over, all data objects form an attraction tree reflecting the attraction hierarchy. One special case here is O_{hd} whose attractor is itself. The corresponding attraction tree T_{hd} cannot be a child of any others. Instead, O_{hd} is the root of the resulting attraction tree. All other data objects are its descendants.

3.3 Deriving a density tree

The attraction tree constructed in Section 3.2 includes every object in the data set. Thus, the tree can be bushy. To identify the really meaningful clusters and their hierarchical structures, we need to identify the clusters and prune the noise and outliers. This is done by a summarization of clusters and dense areas in the form of a density tree.

There are two kinds of nodes in a density tree, namely the collection nodes and the cluster nodes. A *collection node* is an internal node in the tree and represents a dense area of the data set. A *cluster node* represents a cluster that will not be decomposed further. Each node has the medoid of the dense area or the cluster as its representative.

Figure 2 is an example of a density tree. At the root of the tree, the whole data set is regarded as a dense area, and denoted as a root (collection) node A_0. This dense area consists of two dense sub-areas, i.e., A_1 and C_1. The dense sub-areas can be further decomposed to finer sub-dense areas, i.e., C_2, A_2. DHC recursively splits the dense sub-areas until the sub-areas meet some termination criteron. The sub-areas at the leaf level are represented by cluster nodes in the density tree. Each cluster node corresponds to one cluster in the data set.

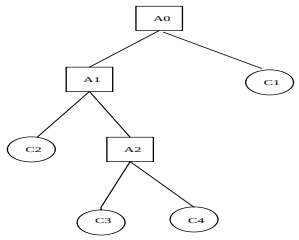


Figure 2: A density tree.

How to derive a density tree from an attraction tree? The basic idea is that, first, we have the whole data set as one dense area to split, then, we recursively split the dense areas until each dense sub-area contains only one cluster.

To determine the clusters, the dense areas and the bridges between them, we introduce two parameters: *similarity threshold* τ and *minimum number of object threshold* $MinPts$. For an edge O_iO_j in an attraction tree, where O_i is the parent of O_j, the attraction sub-tree T_{O_j} is a dense sub-area if and only if (1) $\|Neighbors(O_j, \tau)\| \geq MinPts$ and (2) $similarity(O_j, O_i) \leq \tau$. In other words, a dense area is identified if and only if there are at least $MinPts$ objects in the area and the similarity between the

center of the area and the center of the higher level area is no more than τ.

Once the dense areas are identified, we can derive the density tree using the dense areas and their centers attraction relation stored in the attraction tree. The algorithm is presented in Figure 3. Function $deriveDensityTree$ derives the density tree from the attraction tree $AttractionTree$. We maintain a queue $splitQueue$ to recursively split the dense areas. The $splitQueue$ is initialized with only one element, i.e., the $AttractionTree$. For each iteration, we extract an element from the $splitQueue$ which represents the dense area to split. Then we call the function $split$ to identify sub-dense areas in $splitTree$. If $splitTree$ cannot be further divided, function $split$ returns $splitTree$ unchanged. In this case, we will serialize the $splitTree$ area list of data objects and record it in the cluster list $clusters$. Otherwise, $split$ will return a square type node with split sub-areas as its children. In this case, we put all of the children as candidate split areas and put them into $splitQueue$. The iteration stops when the $splitQueue$ is empty, i.e., no more sub areas can be split.

3.4 Why is DHC effective and efficient?

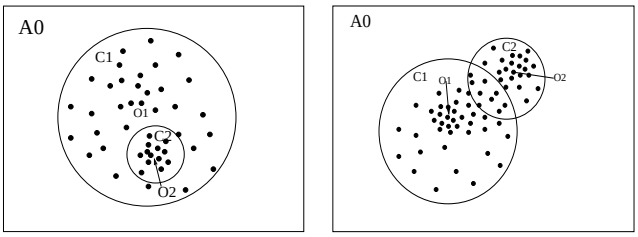
By measuring the density for each data object, DHC captures the natural distribution of the data. Intuitively, a group of highly co-expressed genes will form a dense area (cluster), and the gene with the highest density within the group becomes the medoid of the cluster. There may be many noise objects. However, they distribute sparsely in the object space and cannot show a high degree of co-expression, and thus have low densities.

In summary, DHC has some distinct advantages over some previously proposed methods. *Comparing to k-means and SOM, DHC does not need a parameter about the number of clusters, and the resulting clusters are not affected by outliers*. On the one hand, by locating the dense areas in the object space, DHC automatically detects the number of clusters. On the other hand, since DHC uses the expression pattern of the medoid to represent the average expression patterns of co-expressed genes, the resulting clusters will not be corrupted by outliers.

Comparing to CAST and Adapt, DHC can handle the embedded clusters and highly intersected clusters uniformly. Figure 4 shows an example. Figure 4(a) illustrates two embedded clusters, and Figure 4(b) shows two highly intersected clusters. In both figures, let A_0 be the whole data set, C_1 and C_2 be the two clusters in A_0, and O_1 and O_2 be the medoids of C_1 and C_2. Suppose $density(O_1) > density(O_2)$. After the $attract$ process and $constructTree$ process, in both situations, the following three facts hold: (1) O_1 will be the root of the attraction tree of the data set, since it has a higher density than any other data objects; (2) O_2 will be the root of a subtree that contains the data objects belonging to C_2, since O_2 is the medoid of C_2 and (3) O_2 will be attracted by some data object O_3 and become one of the children of O_3. Figure 5(a) demonstrates the generated attraction tree. After the

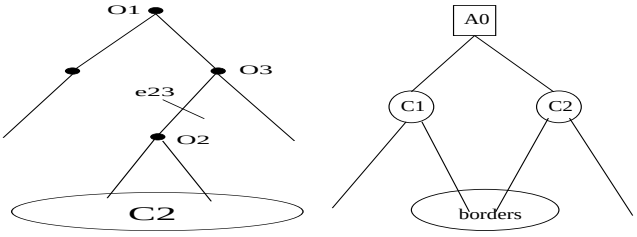
```
Proc deriveDensityTree(AttractionTree)
   splitQueue.add(AttractionTree)
  while (!splitQueue.isEmpty())
     spTree = splitQueue.extract()
     parentTree = spTree.parent
     node = splitTree(spTree)
     if (parentTree == NULL) then root = node
     else parentTree.addChild(node)
     end if
     if (node.type == CLUSTERTYPE) then
       c = node.serialize()
       clusters.add(c)
     else // node.type==COLLECTIONTYPE
       for each child chTree of node do
          chTree.parent = node
          node.remove(chTree)
          splitQueue.add(chTree)
       end for
     end if
  end while
End Proc


struct MaxCut(t,p,dist)
   tree = t; parent = p; distance = dist
end Struct MaxCut
Func splitTree(tree)
  finished = false;
  currentDistance = MAXDISTANCE
  While (! finished)
     cut = findMaxCut(tree,currentDistance)
     if (cut == null) then finished = true
     else
       currentDistance= cut.distance
       if (splitable(mc.tree,tree)) then
          cut.parent.remove(cut.tree)
          finished = true
       end if
     end if
  end while
  if (cut == null) then return tree
  else
    collection = new DensityTree(collection)
    collection.addChild(cut.tree)
    collection.addChild(tree)
     return
  end if
End Func
```

Figure 3: Algorithm DHC.



(a) Embedded cluster (b)Highly intersected cluster

Figure 4: An example of embedded cluster and highly intersected cluster.



(a) Attraction tree. (b) Derived density tree.

Figure 5: Attraction tree and derived density tree of C_1 and C_2.

$deriveDensity$ process, C_2 will be identified as a dense area and split away from the attraction tree from edge e_{23}. The derived density tree is demonstrated by Figure 5 (b). As can be seen, the embedded clusters and intersected clusters are treated uniformly.

Moreover, the result of DHC can be visualized as a hierarchical structure. Comparing to some conventional hierarchial approaches, the two tree approach (density tree and attraction tree) is easier to understand. First, the density tree summarizes the cluster structure of the data set, so there is no need to cut the dendrogram. Second, instead of putting all data objects at the leaf level, DHC puts the medoid at the tree root to represent the "core" of the cluster. Other data objects in the attraction tree are directly or indirectly attracted to the medoid level by level. The levels of the data objects reflect the similarity of the data objects w.r.t. the medoid. The lower level an data object stays, the farther it locates from the "core". Therefore, the attraction tree for each cluster clearly discloses the inner-cluster structure. Third, due to the low densities, outliers are attracted at the leave level of the attraction tree. A post-pruning process can be applied to discard the outliers. We can leave the user to set the prune threshold and thus retain the flexibility of judging the outliers.

4 Experimental Results

We test algorithm DHC on real data sets. Limited by space, we only report the results on two typical real gene expression data sets: (1) the Iyer's Data [10], which contains gene expression levels of 517 human genes in response to serum stimulation over 12 time points; and (2) the Cho's Data [5], which has been preprocessed to remove duplicate genes and genes peaking during multiple phases. It contains 386 gene expression patterns during the cell cycle of the budding yeast *S. cerevisiae.* Those two data sets are public on the web and have become sort of the benchmark data sets for clustering analysis of gene expression data.

In [6], Eisen et al. partitioned the Iyer's data set into 10 clusters according to the gene expression patterns. In [5], Cho et al. listed the functionally characterized genes whose transcripts display periodic fluctuation. Five putative true clusters, which are the sets of early G1-peaking, late G1-peaking, S peaking, G_2 peaking and M peaking genes are reported at http: //171.65.26.52/ yeast_cell_cycle/ functional_categories.html. We use the above partition as the ground truth to test and compare the performance of DHC with those of other clustering algorithms.

The algorithms are implemented on a Sun workstation with a 440 MHz CPU and 256 MB main memory.

4.1 Experiments on the attraction trees and the density trees

Figure 6 is the density tree generated by DHC on the Iyer's data set. It contains 8 clusters (i.e., the 8 leaf nodes with frames in the figure). We compare the 8 clusters with the ground truth. Each cluster corresponds to a cluster in the ground truth. We will provide more detailed results later. To illustrate the attraction tree, we plot Figure 7, which shows the attraction sub-tree of cluster 4. The cluster contains 13 genes, while gene 517 is the medoid of the cluster. As shown in the figure, genes at the second level (i.e., gene $498, 505, 514, 515, 516$) are directly attracted by gene 517, and other genes are indirectly attracted by gene 517. The leaf nodes in the tree represent the genes forming the border of the cluster.

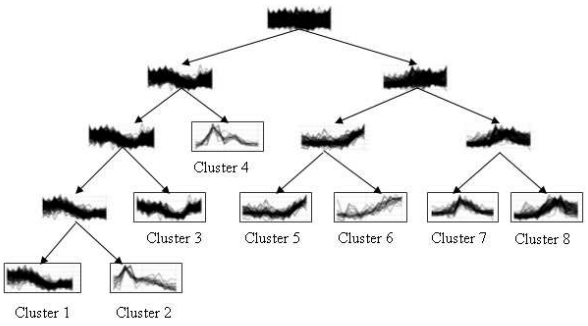


Figure 6: The density tree of the Iyer's Data.

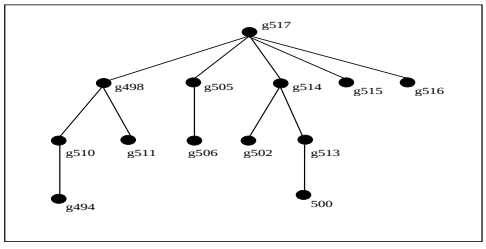


Figure 7: A attraction subtree of cluster 4 in the Iyer's data.

Figures 6 and 7 also demonstrate how the clustering results in DHC can be visualized at two levels. The higher level, the density tree, shows a comprehensive picture of

the cluster structure in the data set. When a user is interested in a cluster or a dense area, she can go to the lower level, the corresponding attraction subtree, which discloses the inner-cluster structures of data objects. The medoid of the cluster, which represents the expression pattern of the whole cluster, sits at the root of the attraction sub-tree. When the tree level goes down, data objects locate farther and farther away from the medoid of the cluster.

4.2 Identification of interesting patterns from noisy data

As mentioned above, a good clustering algorithm for gene expression data must be robust to noise and be able to automatically extract important genes and interesting patterns from noisy data sets. As discussed in Section 2, most distance-based algorithms may corrupt with the noise, i.e., the average profiles of clusters deviate from true interesting patterns. One distinct feature of DHC is that it captures the core area of a cluster that has a significantly higher density than noise. The experimental results suggest that the density tree structure remains robust and the root of the attraction subtree for each cluster still precisely identifies the corresponding pattern of interest, though in a highly noisy environment (e.g., 8 fold noises presents).

In this experiment, we measure the patterns identified from the original Iyer's data. Then, we add one fold, three fold and eight fold noises (permutation of real gene expression data) to the data set. We compute the average profiles of the 10 clusters given by the ground truth as the *standard patterns*, and test each clustering method the ability to identify those standard patterns.

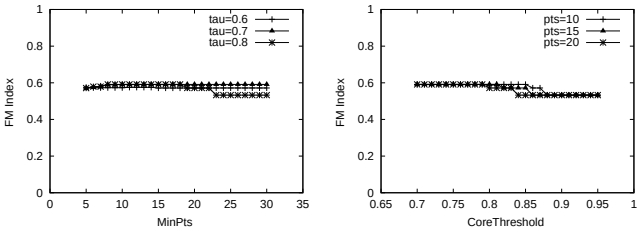
Suppose $\mathcal{C}_A = \{C_1, \ldots, C_m\}$ is the set of resulting clusters from algorithm $\mathcal{A}$, and $\mathcal{P} = \{P_1, \ldots, P_n\}$ is the set of standard gene expression patterns in the ground truth. First, we compute the *average profile* $\tilde{P}_i$ for each cluster C_i, and find the most similar standard pattern P_j that $\tilde{P}_i$ matches. If a set of average profiles $\tilde{P_{j_1}}$, $\tilde{P_{j_2}}$, ..., $\tilde{P_{j_k}}$ matches the same standard pattern P_j, we select the one with the highest similarity with P_j as P_j's *estimate pattern*. Then, we examine the similarity between each standard pattern P_j and its estimate pattern $\tilde{P}_j$. If $Similarity(P_j, \tilde{P}_j) \geq 0.9$, we call the standard pattern P_j is *identified* by the estimate pattern $\tilde{P}_j$.

In Figure 8, we highlight the identified patterns in bold. The value of each cell is the similarity between each standard pattern and its estimate pattern w.r.t. different clustering algorithms. IPN means the number of standard patterns identified by the algorithm. For the original data, our algorithm identified all patterns except P_5 and P_9. P_5 was not identified by any other clustering algorithms, either. P_9 was only barely identified by K-means. When noise rate increases, the behavior of distance-based algorithms become worse, and less and less standard patterns can be identified. However, DHC successfully identifies significant and interesting patterns reliably.

4.3 The robustness to the parameters

In the following experiments, we test the robustness of our algorithm against different settings of parameters. Our algorithm has two parameters, the similarity threshold τ and the minimum number of objects threshold $MinPts$. We test the mining results w.r.t. various parameter settings on Cho's data.

First we test the number of resulted clusters under a wide range of paramter settings. Our experiment shows that the number of clusters reported by DHC does not change much when we change the value of the similarity threshod and the minimum number of object threshold. Next, we test the quality of the mining result in terms of FM index w.r.t. the parameters settings. As shown in Figure 9, the FM index values are not sensitive to the parameter settings. These results strongly confirmed that DHC is robust w.r.t. the input parameters.



(a) FM index vs. $MinPts$ (b) FM index vs. $MinSim$.

Figure 9: The sensitivity of clustering results w.r.t. the input parameters.

4.4 Scalability

We test the scalability of our algorithm by adding noises to the Iyer's data. The experimental results show that DHC can process $10,340$ gene expressions in about $6,000$ seconds. It verifies that DHC is scalable in processing large gene expression data sets. Limited by space, the details are omitted here.

5 Conclusions

Clustering time series gene expression data is an important task in bioinformatics research and bio-medical applications. Although some methods have been adapted or proposed recently, several challenges remain, including the interpretability and visualization of the clustering results, the robustness of the mining results w.r.t. noise, outliers, and global parameters, and handling clusters with arbitrary shape and structures.

In this paper, we propose DHC, a density-based hierarchical clustering method. DHC first organizes all objects in a data set into an attraction tree according to the density-based connectivity. Then, clusters and dense areas (i.e., collections of clusters) are identified.

Noise Fold		Methods				Noise Fold		Methods			
		S O M	K-Means	CAST	D H C			S O M	K-Means	CAST	D H C
0	C1	**0.980**	**0.972**	**0.954**	**0.992**	1	C1	0.879	**0.965**	**0.953**	**0.992**
	C2	**0.990**	**0.948**	0.887	**0.957**		C2	**0.992**	**0.994**	0.696	**0.957**
	C3	-1.0	-1.0	**0.997**	**0.984**		C3	**0.973**	-1.0	**0.994**	**0.984**
	C4	**0.978**	**0.995**	**0.968**	**0.979**		C4	**0.985**	**0.973**	**0.925**	**0.979**
	C5	-1.0	-1.0	-1.0	-1.0		C5	-1.0	0.886	-1.0	-1.0
	C6	-1.0	**0.964**	**0.983**	**0.974**		C6	-1.0	**0.932**	**0.986**	**0.974**
	C7	**0.951**	-1.0	-1.0	**0.966**		C7	-1.0	-1.0	-1.0	**0.966**
	C8	-1.0	**0.962**	**0.999**	**0.970**		C8	**0.985**	**0.961**	**0.986**	**0.970**
	C9	0.893	**0.910**	0.729	-1.0		C9	-1.0	-1.0	0.844	-1.0
	C10	-1.0	**0.930**	**0.995**	**0.973**		C10	-1.0	-1.0	**0.975**	**0.973**
	IPN	**4**	**7**	**6**	**8**		IPN	**4**	**5**	**6**	**8**
3	C1	**0.978**	**0.985**	**0.938**	**0.992**	8	C1	**0.981**	-1.0	0.671	**0.992**
	C2	**0.972**	**0.963**	0.853	**0.957**		C2	**0.976**	**0.938**	**0.947**	**0.957**
	C3	-1.0	-1.0	**0.991**	-1.0		C3	-1.0	**0.936**	**0.950**	-1.0
	C4	**0.955**	-1.0	**0.987**	**0.979**		C4	-1.0	**0.950**	**0.980**	**0.979**
	C5	-1.0	0.860	-1.0	-1.0		C5	-1.0	-1.0	**0.920**	-1.0
	C6	-0.860	0.371	**0.969**	**0.974**		C6	-1.0	0.844	**0.987**	**0.974**
	C7	-1.0	**0.941**	-1.0	**0.966**		C7	**0.937**	-1.0	-1.0	**0.966**
	C8	-1.0	-1.0	**0.986**	**0.970**		C8	0.881	0.889	**0.994**	**0.970**
	C9	0.597	0.875	0.537	**0.905**		C9	-1.0	-1.0	0.860	**0.905**
	C10	0.791	0.365	**0.977**	**0.973**		C10	0.813	-1.0	0.895	**0.973**
	IPN	**3**	**3**	**6**	**8**		IPN	**3**	**3**	**6**	**8**

Figure 8: Pattern identification from the noise-added Iyer's Data.

As verified by our empirical evaluation, DHC is clearly more robust than some typical methods proposed previously, in terms of handling noise, outliers, structures of the clusters, and user-specified parameters. Moreover, the clustering results from DHC fits the ground truth better than most of the previously proposed methods in many cases. As a distinct feature, the mining results from DHC can be visualized and interpreted systematically. All these features make DHC a desirable method for bioinformatics data analysis.

References

[1] Golub T. R. *et al.* Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, Vol. 286(15):531–537, October 1999.

[2] Alon U. *et al.* Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide array. *Proc. Natl. Acad. Sci. USA*, Vol. 96(12):6745–6750, June 1999.

[3] Ankerst, M. *et al.* . OPTICS: Ordering Points To Identify the Clustering Structure. *Sigmod*, pages 49–60, 1999.

[4] Ben-Dor A. *et al.* Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

[5] Cho, R. *et al.* A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2:65–73, 1998.

[6] Eisen M.B. *et al.* Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA*, Vol. 95:14863–14868, 1998.

[7] M. *et al.* Ester. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of 2nd International Conference on KDD*, pages 226–231, Portland, OR, 1996.

[8] Frank De Smet *et al.* Adaptive quality-based clustering of gene expression profiles. *Bioinformatics*, 18:735–746, 2002.

[9] Hinneburg, A. *et al.* An efficient approach to clustering in large multimedia database with noise. *Proc. 4th Int. Con. on Knowledge discovery and data mining*, 1998.

[10] Iyer V.R. *et al.* The transcriptional program in the response of human fibroblasts to serum. *Science*, 283:83–87, 1999.

[11] Lockhart, D. *et al.* Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nat. Biotechnol*, 14:1675–1680, 1996.

[12] Schena,M., D. Shalon, R. Davis, and P.Brown. Quantitative monitoring of gene expression patterns with a compolementatry DNA microarray. *Science*, 270:467–470, 1995.

[13] Shamir R. *et al.* Click: A clustering algorithm for gene expression analysis. In *In Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB '00). AAAI Press.*, 2000.

[14] Sherlock G. Analysis of large-scale gene expression data. *Curr Opin Immunol*, 12(2):201–205, 2000.

[15] Spellman PT. *et al.* Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol Biol Cell*, 9(12):3273–3297, 1998.

[16] Tamayo P. *et al.* Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proc. Natl. Acad. Sci. USA*, Vol. 96(6):2907–2912, March 1999.

[17] Tavazoie, S. *et al.* Systematic determination of genetic network architecture. *Nature Genet*, pages 281–285, 1999.