Job: Molecular Biotechnology
Chapter: Karypis/MB05-0033
Pub Date: 2005
Operator: SV
Date: 6/05
Revision: 1st Pass

REVIEW

Data Clustering in Life Sciences

*Ying Zhao and George Karypis**

Abstract

Clustering has a wide range of applications in life sciences and over the years has been used in many areas ranging from the analysis of clinical information, phylogeny, genomics, and proteomics. The primary goal of this article is to provide an overview of the various issues involved in clustering large biological datasets, describe the merits and underlying assumptions of some of the commonly used clustering approaches, and provide insights on how to cluster datasets arising in various areas within life-sciences. We also provide a brief introduction to CLUTO, a general purpose toolkit for clustering various datasets, with an emphasis on its applications to problems and analysis requirements within life sciences.

Index Entries: Clustering algorithms; similarity between objects; microarray data; Cluto.

1. Introduction

Clustering is the task of organizing a set of objects into meaningful groups. These groups can be disjoint, overlapping, or organized in some hierarchical fashion. The key element of clustering is the notion that the discovered groups are *meaningful*. This definition is intentionally vague, as what constitutes meaningful is to a large extent, application dependent. In some applications this may translate to groups in which the pairwise similarity between their objects is maximized, and the pairwise similarity between objects of different groups is minimized. In some other applications this may translate to groups that contain objects that share some key characteristics, although their overall similarity is not the highest. Clustering is an exploratory tool for analyzing large datasets, and has been used extensively in numerous application areas.

Clustering has a wide range of applications in life sciences and over the years has been used in many areas ranging from the analysis of clinical information, phylogeny, genomics, and proteomics. For example, clustering algorithms applied to gene expression data can be used to identify co-regulated genes and provide a genetic fingerprint for various diseases. Clustering algorithms applied on the entire database of known proteins can be used to automatically organize the different proteins into close- and distant-related families, and identify subsequences that are mostly preserved across proteins ***(1–5)***. Similarly, clustering algorithms applied to the tertiary structural datasets can be used to perform a similar organization and provide insights in the rate of change between sequence and structure ***(6,7)***.

The primary goal of this article is to provide an overview of the various issues involved in clustering large datasets, describe the merits and underlying assumptions of some of the commonly used clustering approaches, and provide insights on how to cluster datasets arising in various areas within life sciences. Toward this end, the article is organized, broadly, in three parts. The first part (*see* **Headings 2. to 4.**) describes the various types of clustering algorithms developed over the years,

*Author to whom all correspondence and reprint requests should be addressed. Department of Computer Science and Engineering, University of Minnesota, and Digital Technology Center and Army HPC Research Center, Minneapolis, MN 55455. E-mail: karypis@cs.umn.edu.

Job: Molecular Biotechnology
Chapter: Karypis/MB05-0033
Pub Date: 2005
Operator: SV
Date: 6/05
Revision: 1st Pass

the various methods for computing the similarity between objects arising in life sciences, and methods for assessing the quality of the clusters. The second part (*see* **Heading 5.**) focuses on the problem of clustering data arising from microarray experiments and describes some of the commonly used approaches. Finally, the third part (*see* **Heading 6.**) provides a brief introduction to Cluto, a general purpose toolkit for clustering various datasets, with an emphasis on its applications to problems and analysis requirements within life sciences.

2. Types of Clustering Algorithms

The topic of clustering has been extensively studied in many scientific disciplines and a variety of different algorithms have been developed ***(8–20)***. Two recent surveys on the topics ***(21,22)*** offer a comprehensive summary of the different applications and algorithms. These algorithms can be categorized along different dimensions based either on the underlying methodology of the algorithm, leading to *partitional* or *agglomerative* approaches; the structure of the final solution, leading to *hierarchical* or *nonhierarchical* solutions; the characteristics of the space in which they operate, leading to *feature* or *similarity* approaches; or the type of clusters that they discover, leading to *globular* or *transitive* clustering methods.

2.1. Agglomerative and Partitional Algorithms

Partitional algorithms, such as *K*-means ***(9,23)***, *K*-medoids ***(9,11,13)***, probabilistic ***(10,24)***, graph-partitioning based ***(9,25–27)***, or spectral based ***(28)***, find the clusters by partitioning the entire dataset into either a predetermined or an automatically derived number of clusters.

Partitional clustering algorithms compute a *k*-way clustering of a set of objects either directly or through a sequence of repeated bisections. A direct *k*-way clustering is commonly computed as follows. Initially, a set of *k* objects is selected from the datasets to act as the *seeds* of the *k* clusters. Then, for each object, its similarity to these *k* seeds is computed, and it is assigned to the cluster corresponding to its most similar seed. This forms the initial *k*-way clustering. This clustering is then repeatedly refined so that it optimizes a desired clustering criterion function. A *k*-way partitioning through repeated bisections is obtained by recursively applying the above algorithm to compute two-way clustering (i.e., bisections). Initially, the objects are partitioned into two clusters, then one of these clusters is selected and is further bisected, and so on. This process continues *k*–1 times, leading to *k* clusters. Each of these bisections is performed so that the resulting two-way clustering solution optimizes a particular criterion function.

Criterion functions used in the partitional clustering reflect the underlying definition of the "goodness" of clusters. The partitional clustering can be considered as an optimization procedure that tries to create high-quality clusters according to a particular criterion function. Many criterion functions have been proposed ***(9,29,30)*** and some of them are described later in **Heading 6**. Criterion functions measure various aspects of intracluster similarity, intercluster dissimilarity, and their combinations. These criterion functions use different views of the underlying collection, by either modeling the objects as vectors in a high-dimensional space or by modeling the collection as a graph.

Hierarchical agglomerative algorithms find the clusters by initially assigning each object to its own cluster and then repeatedly merging pairs of clusters until a certain stopping criterion is met. Consider an *n*-object dataset and the clustering solution that has been computed after performing merging steps. This solution will contain exactly *n*–1 clusters, as each merging step reduces the number of clusters by one. Now, given this *(n–1)*-way clustering solution, the pair of clusters that is selected to be merged next is the one that leads to an (*n*–*l*–1)-way solution that optimizes a particular criterion function. That is, each one of the $(n–1) \times (n–l–1)/2$ pairs of possible merges is evaluated, and the one that leads to a clustering solution that has the maximum (or minimum) value of the particular criterion function is selected. Thus, the criterion function is *locally* optimized within each particular stage of agglomerative al-

gorithms. Depending on the desired solution, this process continues until either there are only clusters left, or when the entire agglomerative tree has been obtained.

The three basic criteria to determine which pair of clusters to be merged next are single-link ***(31)***, complete-link ***(32)***, and group average (UPGMA [unweighted pair group method with arithmetic mean]) ***(9)***. The single-link criterion function measures the similarity of two clusters by the maximum similarity between any pair of objects from each cluster, whereas the complete-link criterion uses the minimum similarity. In general, both the single-link and the complete-link approaches do not work very well because they either base their decisions to a limited amount of information (single-link) or assume that all the objects in the cluster are very similar to each other (complete-link). On the other hand, the group average approach measures the similarity of two clusters by the average of the pairwise similarity of the objects from each cluster and does not suffer from the problems arising with single-link and complete-link. In addition to these three basic approaches, a number of more sophisticated schemes have been developed, such as CURE ***(18)***, ROCK ***(19)***, CHAMELEON ***(20)***, that have been shown to produce superior results.

Finally, hierarchical algorithms produce a clustering solution that forms a dendrogram, with a single all-inclusive cluster at the top and single-point clusters at the leaves. In contrast, in nonhierarchical algorithms there tends to be no relation between the clustering solutions produced at different levels of granularity.

2.2. Feature-Based and Similarity-Based Clustering Algorithms

Another distinction between the different clustering algorithms is whether or not they operate on the object's feature space or operate on a derived similarity (or distance) space. *K*-means–based algorithms are the prototypical examples of methods that operate on the original feature space. In this class of algorithms, each object is represented as a multidimensional feature vector, and the clustering solution is obtained by iteratively optimizing the similarity (or distance) between each object and its cluster centroid. On the other hand, similarity-based algorithms compute the clustering solution by first computing the pairwise similarities between all the objects and then use these similarities to drive the overall clustering solution. Hierarchical agglomerative schemes, graph-based schemes, as well as *K*-medoid, fall into this category. The advantages of similarity-based methods is that they can be used to cluster a wide variety of datasets, provided that reasonable methods exist for computing the pairwise similarity between objects. For this reason, they have been used to cluster both sequential ***(1,2)*** as well as graph datasets ***(33,34)***, especially in biological applications. On the other hand, there has been limited work in developing clustering algorithms that operate directly on the sequence or graph datasets ***(35)***.

However, similarity-based approaches have two key limitations. First, their computational requirements are high as they need to compute the pairwise similarity between all the objects that need to be clustered. As a result, such algorithms can only be applied to relatively small datasets (a few thousand objects), and they cannot be effectively used to cluster the datasets arising in many fields within life sciences. The second limitation of these approaches is that by not using the object's feature space and relying only on the pairwise similarities, they tend to produce suboptimal clustering solutions, especially when the similarities are low relative to the cluster sizes. The key reason for this is that these algorithms can only determine the overall similarity of a collection of objects (i.e., a cluster) by using measures derived from the pairwise similarities (e.g., average, median, or minimum pairwise similarities). However, such measures, unless the overall similarity between the members of different clusters is high, are quite unreliable as they cannot capture what is *common* between the different objects in the collection.

Clustering algorithms that operate in the object's feature space can overcome both of these limitations. Because they do not require the precomputation of the pairwise similarities, fast partitional

Job: Molecular Biotechnology
Chapter: Karypis/MB05-0033
Pub Date: 2005
Operator: SV
Date: 6/05
Revision: 1st Pass

algorithms can be used to find the clusters, and because their clustering decisions are made in the object's feature space, they can potentially lead to better clusters by correctly evaluating the similarity between a collection of objects. For example, in the context of clustering protein sequences, the proteins in each cluster can be analyzed to determine the conserved blocks, and use only these blocks in computing the similarity between the sequences (an idea formalized by profile HMM approaches; ***36***,***37***). Recent studies in the context of clustering large high-dimensional datasets done by various groups ***(38–41)*** show the advantages of such algorithms over those based on similarity.

2.3. Globular and Transitive Clustering Algorithms

Besides the operational differences between various clustering algorithms, another key distinction between them is the type of clusters that they discover. There are two general types of clusters that often arise in different application domains. What differentiates these types is the relationship between the cluster's objects and the dimensions of their feature space.

The first type of clusters contains objects that exhibit a strong pattern of conservation along a subset of their dimensions. That is, there is a subset of the original dimensions in which a large fraction of the objects agree. For example, if the dimensions correspond to different protein motifs, then a collection of proteins will form a cluster, if there exists a subset of motifs that is present in a large fraction of the proteins. This subset of dimensions is often referred to as a *subspace*, and the above stated property can be viewed as the cluster's objects and its associated dimensions forming a *dense subspace*. Of course, the number of dimensions in these dense subspaces, and the density (i.e., how large is the *fraction* of the objects that share the same dimensions) will be different from cluster to cluster. Exactly what is this variation in subspace size and density (and the fact that an object can be part of multiple disjoint or overlapping dense subspaces) is what complicates the problem of discovering this type of clusters. There are a number of application areas in which such clusters give rise to meaningful groupings of the objects (i.e., domain experts will tend to agree that the clusters are correct). Such areas includes clustering documents based on the terms they contain, clustering customers based on the products they purchase, clustering genes based on their expression levels, and clustering proteins based on the motifs they contain.

The second type of clusters contains objects in which again there exists a subspace associated with that cluster. However, unlike the earlier case, in these clusters there will be subclusters that may share a very small number of the subspace's dimension, but there will be *a strong path* within that cluster that will connect them. By "strong path" we mean that if A and B are two subclusters that share only a few dimensions, then there will be another set of subclusters $X_1,X_2,...,X_k$, that belong to the cluster, such that each of the subcluster pairs (A,X_1), (X_1,X_2),...,(X_k,B) will share many of the subspace's dimensions. What complicates cluster discovery in this setting is that the connections (i.e., shared subspace dimensions) between subclusters within a particular cluster will tend to be of different strength. Examples of this type of clusters include protein clusters with distant homologies or clusters of points that form spatially contiguous regions.

Our discussion so far focused on the relationship between the objects and their feature space. However, these two classes of clusters can also be understood in terms of the object-to-object similarity graph. The first type of clusters will tend to contain objects in which the similarity between all pairs of objects will be high. On the other hand, in the second type of clusters there will be a lot of objects whose direct pairwise similarity will be quite low, but these objects will be connected by many paths that stay within the cluster that traverse high similarity edges. The names of these two cluster types were inspired by this similarity-based view, and they are referred to as *globular* and *transitive* clusters, respectively.

The various clustering algorithms are in general suited for finding either globular or transitive clusters. In general, clustering criterion-driven

partitional clustering algorithms such as *K*-means and its variants and agglomerative algorithms using the complete-link or the group-average method are suited for finding globular clusters. On the other hand, the single-link method of the agglomerative algorithm and graph-partitioning-based clustering algorithms that operate on a nearest neighbor similarity graph are suited for finding transitive clusters. Finally, specialized algorithms, called *subspace* clustering methods, have been developed to explicitly find either globular or transitive clusters by operating directly in the object's feature space ***(42–44)***.

3. Methods for Measuring the Similarity Between Objects

In general, the method used to compute the similarity between two objects depends on two factors. The first factor has to do with how the objects are actually being represented. For example, the similarity between two objects represented by a set of attribute-value pairs will be entirely different from the method used to compute the similarity between two DNA sequences and two three-dimensional protein structures. The second factor is much more subjective and has to do with the actual goal of clustering. Different analysis requirements may give rise to entirely different similarity measures and different clustering solutions. This section focuses on discussing various methods for computing the similarity between objects that address both of these factors.

The diverse nature of biological sciences and the shear complexity of the underlying physicochemical and evolutionary principles that need to be modeled, gives rise to numerous clustering problems involving a wide range of different objects. The most prominent of them are the following:

Multidimensional Vectors: Each object is represented by a set of attribute-value pairs. The meaning of the attributes (also referred to as variables or features) is application dependent and includes datasets like those arising from various measurements (e.g., gene expression data), or from various clinical sources (e.g., drug response, disease states).

Sequences: Each object is represented as a sequence of symbols or events. The meaning of these symbols or events also depends on the underlying application and includes objects, such as DNA and protein sequences, sequences of secondary structure elements, temporal measurements of various quantities, such as gene expressions, and historical observations of disease states.

Structures: Each object is represented as a two- or three-dimensional structure. The primary examples of such datasets include the spatial distribution of various quantities of interest within various cells, and the three-dimensional geometry of chemical molecules such as enzymes and proteins.

The rest of this section describes some of the most popular methods for computing the similarity for all these types of objects.

3.1. Similarity Between Multidimensional Objects

There are a variety of methods for computing the similarity between two objects that are represented by a set of attribute-value pairs. These methods, to a large extent, depend on the nature of the attributes themselves and the characteristics of the objects that we need to model by the similarity function.

From the point of similarity calculations, there are two general types of attributes. The first one consists of attributes whose range of values is *continuous*. This includes both integer-valued and real-valued variables, as well as attributes whose allowed set of values are thought to be part of an ordered set. Examples of such attributes include gene expression measurements, ages, disease severity levels. On the other hand, the second type consists of attributes that take values from an unordered set. Examples of such attributes include various gender, blood type, and tissue type. We will refer to the first type as *continuous* attributes and to the second type as *categorical* attributes. The primary difference between these two types of attributes is that in the case of continuous attributes, when there is a mismatch on the value taken by a particular attribute in two different objects, the dif-

ference of the two values is a meaningful measure of distance, whereas in categorical attributes, there is no easy way to assign a distance between such mismatches.

In the rest of this section, we present methods for computing the similarity assuming that all the attributes in the objects are either continuous or categorical. However, in most real applications, objects will be represented by a mixture of such attributes, therefore the described approaches need to be combined.

3.1.1. Continuous Attributes

When all the attributes are continuous, each object can be considered to be a vector in the attribute space. That is, if n is the total number of attributes, then each object v can be represented by an n-dimensional vector $(v_1,v_2,...,v_n)$, where v_i is the value of the ith attribute.

Given any two objects with their corresponding vector–space representations v and u, a widely used method for computing the similarity between them is to look at their distance as measured by some norm of their vector difference. That is,

$$\mathrm{dis}_r(v,u) = \|v - u\|_r \qquad (1)$$

where r is the norm used, and $\|\cdot\|$ is used to denote vector norms. If the distance is small, then the objects will be similar, and the similarity of the objects will decrease as their distance increases.

The two most commonly used norms are the one-norm and the two-norm. In the case of the one-norm, the distance between two objects is give by

$$\mathrm{dis}_1(v,u) = \|v - u\|_1 = \sum_i^n |v_i - u_i| \qquad (2)$$

where $|\cdot|$ denotes absolute values. Similarly, in the case of the two-norm, the distance is given by

$$\mathrm{dis}_2(v,u) = \|v - u\|_2 = \sqrt{\sum_{i=1}^n (v_i - u_i)^2} \qquad (3)$$

Note that the one-norm distance is also called the *Manhattan* distance, whereas the two-norm distance is nothing more than the *Euclidean* distance between the vectors. Those distances may become problematic when clustering high-dimensional data, because in such datasets, the similarity between two objects is often defined along a small subset of dimensions.

An alternate way of measuring the similarity between two objects in the vector–space model is to look at the angle between their vectors. If two objects have vectors that point to the same direction (i.e., their angle is small), then these objects will be considered similar, and if their vectors point to different directions (i.e., their angle is large), then these vectors will be considered dissimilar. This angle-based approach for computing the similarity between two objects emphasizes the relative values that each dimension takes within each vector, and not their overall length. That is, two objects can have an angle of zero (i.e., point to the identical direction), even if their Euclidean distance is arbitrarily large. For example, in a two-dimensional space, the vectors $v = (1,1)$, and $u = (1000,1000)$ will be considered to be identical, as their angle is zero. However, their Euclidean distance is close to $1000\sqrt{2}$.

Because the computation of the angle between two vectors is somewhat expensive (requiring inverse trigonometric functions), we do not measure the angle itself but its cosine function. The cosine of the angle between two vectors v and u is given by

$$\mathrm{sim}(v,u) = \cos(v,u) = \frac{\sum_{i=1}^n v_i u_i}{\|v\|_2 \; \|u\|_2} \qquad (4)$$

This measure will be plus one, if the angle between v and u is zero, and minus one, if their angle is 180 degrees (i.e., point to opposite directions). Note that a surrogate for the angle between two vectors can also be computed using the Euclidean distance, but instead of computing the distance between v and u directly, we need to first scale them to be of unit length. In that case, the Euclidean distance measures the *chord* between the two vectors in the unit hypersphere.

In addition to the above linear algebra inspired methods, another widely used scheme for determining the similarity between two vectors uses the Pearson correlation coefficient, which is given by

$$\mathrm{sim}(v,u) = \mathrm{corr}(v,u) = \frac{\sum_{i=1}^n (v_i - \bar{v})(u_i - \bar{u})}{\sqrt{\sum_{i=1}^n (v_i - \bar{v})^2} \sqrt{\sum_{i=1}^n (v_i - \bar{u})^2}} \qquad (5)$$

where $\bar{v}$ and $\bar{u}$ are the mean of the values of the *v* and *u* vectors, respectively. Note that Pearson's correlation coefficient is nothing more than the cosine between the mean-subtracted *v* and *u* vectors. As a result, it does not depend on the length of the $(v - \bar{v})$ and $(u - \bar{u})$ vectors, but only on their angle.

Our discussion so far on similarity measures for continuous attributes focused on objects in which all the attributes were homogeneous in nature. A set of attributes are called *homogeneous* if all of them measure quantities that are of the same type. As a result changes in the values of these variables can be easily correlated across them. Quite often, each object will be represented by a set of inhomogeneous attributes. For example, if we would like to cluster patients, then some of the attributes describing each patient can measure things like age, weight, height, calorie intake. If we use some of these methods to compute the similarity, we will essentially make the assumption that equal magnitude changes in all variables are identical. However, this may not be the case. If the age of two patients is 50 yr, that represents something that is significantly different if their calorie intake difference is 50 cal. To address these problems, the various attributes need to be first normalized before using any of these similarity measures. Of course, the specific normalization method is attribute dependent, but its goal should be to make differences across different attributes comparable.

3.1.2. Categorical Attributes

If the attributes are categorical, special similarity measures are required, as distances between their values cannot be defined in an obvious manner. The most straightforward way is to treat each categorical attribute individually and define the similarity based on whether two objects contain the exact same value for each categorical attribute. Huang ***(45)*** formalized this idea by introducing dissimilarity measures between objects with categorical attributes that can be used in any clustering algorithms. Let *X* and *Y* be two objects with *m* categorical attributes, and X_i and Y_i be the values of the *i*th attribute of the two objects, the dissimilarity measure between *X* and *Y* is defined to be the number of mismatching attributes of the two objects. That is,

$$d(X,Y) = \sum_{i=1}^{m} S(X_i, Y_i)$$

where

$$S(X_i, Y_i) = \begin{cases} 0 & (X_i = Y_i) \\ 1 & (X_i \neq Y_i) \end{cases}$$

A normalized variant of the above dissimilarity is defined as follows

$$d(X,Y) = \sum_{i=1}^{m} \frac{n_{X_i} + n_{Y_i}}{n_{X_i} n_{Y_i}} S(X_i, Y_i)$$

where $n_{X_i}(n_{Y_i})$ is the number of times the value X_i (Y_i) appears in the ith attribute of the entire dataset. If two categorical values are common across the dataset, they will have low weights, so that the mismatch between them will not contribute significantly to the final dissimilarity score. If two categorical values are rare in the dataset, then they are more informative and will receive higher weights according to the formula. Hence, this dissimilarity measure emphasizes the mismatches that happen for rare categorical values than for those involving common ones.

One of the limitations of the preceding method is that two values can contribute to the overall similarity only if they are the same. However, different categorical values may contain useful information in the sense that even if their values are different, the objects containing those values are related to some extent. By defining similarities just based on matches and mismatches of values, some useful information may be lost. A number of approaches have been proposed to overcome this limitation ***(19,46–48)*** by using additional information between categories or relationships between categorical values.

3.2. Similarity Between Sequences

One of the most important applications of clustering in life sciences is clustering sequences, for example, DNA or protein sequences. Many clustering algorithms have been proposed to enhance sequence database searching, organize sequence

databases, generate phylogenetic trees, or guide multiple sequence alignment. In this specific clustering problem, the objects of interest are biological sequences, which consist of a sequence of *symbols*, which could be nucleotides, amino acids, or secondary structure elements (SSEs). Biological sequences are different from the objects we have discussed so far, in the sense that they are not defined by a collection of attributes. Hence, the similarity measures we discussed so far are not applicable to biological sequences.

Over the years, a number of different approaches have been developed for computing similarity between two sequences ***(49)***. The most common ones are the alignment-based measures, which first compute an optimal alignment between two sequences (either globally or locally), and then determine their similarity by measuring the degree of agreement in the aligned positions of the two sequences. The aligned positions are usually scored using a symbol-to-symbol scoring matrix, and in the case of protein sequences, the most commonly used scoring matrices are PAM ***(50,51)*** or BLOSUM ***(52)***.

The global sequence alignment (Needleman-Wunsch alignment; ***53***) aligns the entire sequences using dynamic programming. The recurrence relations are the following ***(49)***. Given two sequences S_1 of length n and S_2 of length m, and a scoring matrix S, let score(i, j) be the of the optimal alignment of prefixes and $S_1[1...i]$ and $S_2[1...j]$.

The base conditions are

$$\text{score}(0, j) = \sum_{1 \le k \le j} S(\_, S_2(k))$$

and

$$\text{score}(i, 0) = \sum_{1 \le k \le i} S(S_1(k), \_)$$

Then, the general recurrence is

$$\text{score}(_{0,\,j}) = \max \begin{cases} \text{score}(i-1, j-1) + S(S_1(i), S_2(i)) \\ \text{score}(i-1, j) + S(S_1(i), \_) \\ \text{score}(i, j-1) + S(\_, S_2(j)) \end{cases}$$

where '_' represents a space, S is the scoring matrix to specify the matching score for each pair of symbols, and score(n,m) is the optimal alignment score.

These global similarity scores are meaningful when we compare similar sequences with roughly the same length (e.g., protein sequences from the same protein family). However, when sequences are of different lengths and are quite divergent, the alignment of the entire sequences may not make sense, in which case, the similarity is commonly defined on the conserved subsequences. This problem is referred to as the *local alignment problem*, which seeks to find the pair of substrings of the two sequences that has the highest global alignment score among all possible pairs of substrings. Local alignments can be computed optimally via a dynamic programming algorithm, originally introduced by Smith and Waterman ***(54)***. The base conditions are score(0,j) = 0 and score(i,0) = 0, and the general recurrence is given by

$$\text{score}(_{0,\,j}) = \max \begin{cases} 0 \\ \text{score}(i-1, j-1) + S(S_1(i), S_2(i)) \\ \text{score}(i-1, j) + S(S_1(i), \_) \\ \text{score}(i, j-1) + S(\_, S_2(j)) \end{cases}$$

The local sequence alignment score corresponds to the cells of the dynamic programming table that have the highest value. Note that the recurrence for local alignments is very similar to that for global alignments only with minor changes, which allow the alignment to begin from any location (i, j) ***(49)***.

Alternatively, local alignments can be computed approximately via heuristic approaches, such as FASTA ***(55,56)*** or BLAST ***(57)***. The heuristic approaches achieve low time complexities by first identifying promising locations in an efficient way, and then applying a more expensive method on those locations to construct the final local sequence alignment. The heuristic approaches are widely used for searching protein databases because of their low time complexity. Description of these algorithms is beyond the scope of this article, and the interested reader should follow the references.

Most existing protein clustering algorithms use the similarity measure based on the local alignment methods (i.e., Smith-Waterman, BLAST

Job: Molecular Biotechnology
Chapter: Karypis/MB05-0033
Pub Date: 2005
Operator: SV
Date: 6/05
Revision: 1st Pass

and FASTA [GeneRage; ***2***, ProtoMap ***58***]). These clustering algorithms first obtain the pairwise similarity scores of all pairs of sequences. Then they either normalize the scores by the self-similarity scores of the sequences to obtain a percentage value of identicalness ***(59)***, or transform the scores to binary values based on a particular threshold ***(2)***. Other methods normalize the row similarity scores by taking into account other sequences in the dataset. For example, ProtoMage ***(58)*** first generates the distribution of the pairwise similarities between sequence A and the other sequences in the database. Then the similarity between sequence A and sequence B is defined as the expected value of the similarity score found for A and B, based on the overall distribution. A low expected value indicates a significant and strong connection (similarity).

Au: ref 58 states ProtoMap not ProtoMage

3.3. Similarity Between Structures

Methods for computing the similarity between the three-dimensional structures of two proteins (or other molecules) are intrinsically different from any of the approaches that we have seen so far for comparing multidimensional objects and sequences. Moreover, unlike the previous data types for which there are well-developed and widely accepted methods for measuring similarities, the methods for comparing three-dimensional structures are still evolving, and the entire field is an active research area. Providing a comprehensive description of the various methods for computing the similarity between two structures requires a chapter (or a book) of its own, and is far beyond the scope of this article. For this reason, our discussion in the rest of this section will primarily focus on presenting some of the issues involved in comparing three-dimensional structures, in the context of proteins, and outlining some of the approaches that have been proposed for solving them. The reader should refer to the chapter by Johnson and Lehtonen ***(60)*** that provide an excellent introduction on the topic.

Au: ref 60 does not have an author; please complete reference

The general approach, which almost all methods for computing the similarity between a pair of three-dimensional protein structures follow, is to try to *superimpose* the structure of one protein on top of the structure of the other protein, so that certain key features are mapped very close to each other in space. Once this is done, then the similarity between two structures is computed by measuring the *fit* of the superposition. This fit is commonly computed as the *root mean square deviations* (RMSD) of the corresponding features. To some extent, this is similar in nature to the alignment performed for sequence-based similarity approaches, but it is significantly more complicated as it involves three-dimensional structures with substantially more degrees of freedom. There are a number of different variations for performing this superposition that have to do with (1) the features of the two proteins that are sought to be matched, (2) whether or not the proteins are treated as rigid or flexible bodies, (3) how the equivalent set of features from the two proteins are determined, and (4) the type of superposition that is computed.

In principle, when comparing two protein structures we can treat every atom of each amino acid side chain as a feature and try to compute a superposition that matches all of them as well as possible. However, this usually does not lead to good results because the side chains of different residues will have different number of atoms with different geometries. Moreover, even the same amino acid types may have side chains with different conformations, depending on their environment. As a result, even if two proteins have very similar backbones, a superposition computed by looking at all the atoms may fail to identify this similarity. For this reason, most approaches try to superimpose two protein structures by focusing on the C_a atoms of their backbones, whose locations are less sensitive on the actual residue type. Besides these atom-level approaches, other methods focus on SSEs and superimpose two proteins so that their SSEs are geometrically aligned with each other.

Most approaches for computing the similarity between two structures treat them as rigid bodies, and try to find the appropriate geometric transformation (i.e., rotation and translation) that leads to

the best superposition. Rigid-body geometric transformations are well-understood and they are relatively easy to compute efficiently. However, by treating proteins as rigid bodies we may get poor superpositions when the protein structures are significantly different, although they are part of the same fold. In such cases, allowing some degree of flexibility tends to produce better results, but also increases the complexity. In trying to find the best way to superimpose one structure on top of the other in addition to the features of interest we must identify the pairs of features from the two structures that will be mapped against each other. There are two general approaches for doing that. The first approach relies on an initial set of equivalent features (e.g., C_a atoms or SSEs) being provided by domain experts. This initial set is used to compute an initial superposition and then additional features are identified using various approaches based on dynamic programming or graph theory ***(53,61,62)***. The second approach tries to automatically identify the correspondence between the various features by various methods including structural comparisons based on matching C_a-atoms contact maps ***(63)***, or on the optimal alignment of SSEs ***(64)***.

Finally, as it was the case with sequence alignment, the superposition of three-dimensional structures can be done globally, whose goal is to superimpose the entire protein structure, or locally, which seeks to compute a good superposition involving a subsequence of the protein.

4. Assessing Cluster Quality

Clustering results are hard to evaluate, especially for high-dimensional data and without a prior knowledge of the objects' distribution, which is quite common in practical cases. However, assessing the quality of the resulting clusters is as important as generating the clusters. Given the same dataset, different clustering algorithms with various parameters or initial conditions will give very different clusters. It is essential to know whether the resulting clusters are valid and how to compare the quality of the clustering results, so that the right clustering algorithm can be chosen and the best clustering results can be used for further analysis.

Another related problem is answering the question, how many clusters are there in the dataset? An ideal clustering algorithm should be the one that can automatically discover the natural clusters present in the dataset based on the underlying cluster definition. However, there are no such universal cluster definitions and clustering algorithms suitable for all kinds of datasets. As a result, most existing algorithms require either the number of clusters to be provided as a parameter as it is done in the case of *K*-means, or a similarity threshold that will be used to terminate the merging process in the case of agglomerative clustering. However, in general, it is hard to know the right number of clusters or the right similarity threshold without a prior knowledge of the dataset.

One possible way to automatically determine the number of clusters *k* is to compute various clustering solutions for a range of values of *k*, *score* the resulting clusters based on some particular metric and then select the solution that achieves the best score. A critical component of this approach is the method used to measure the quality of the cluster. To solve this problem, numerous approaches have been proposed in a number of different disciplines including pattern recognition, statistics, and data mining. The majority of them can be classified into two groups: *external quality measures* and *internal quality measures*.

The approaches based on external quality measures require *a priori* knowledge of the natural clusters that exist in the dataset, and validate a clustering result by measuring the agreement between the discovered clusters and the known information. For instance, when clustering gene expression data, the known functional categorization of the genes can be treated as the natural clusters, and the resulting clustering solution will be considered correct if it leads to clusters that preserve this categorization. A key aspect of the external quality measures is that they use information other than that used by the clustering algorithms. However, such reliable *a priori* knowledge is usually not available when analyzing real datasets—after all, clustering is used as a tool to discover such knowledge in the first place.

The basic idea behind internal quality measures is rooted from the definition of clusters. A meaningful clustering solution should group objects into various clusters, so that the objects within each cluster are more similar to each other than the objects from different clusters. Therefore, most of the internal quality measures evaluate the clustering solution by looking at how similar the objects are within each cluster and how well the objects of different clusters are separated. For example, the pseudo F statistic suggested by Calinski and Harabasz ***(65)*** uses the quotient between the intracluster average squared distance and intercluster average squared distance. If we have X as the centroid (i.e., mean vector) of all the objects, X_j as the centroid of the objects in cluster C_j, k as the total number of clusters, X_j as the total number of objects, and as the squared Euclidean distance between two object-vectors x and y, then the pseudo F statistic is defined as follows:

$$F = \frac{\frac{\sum_{i=1}^{n} d(x_i, X) - \sum_{j=1}^{k} \sum_{x \in C_j} d(x, X_j)}{k-1}}{\frac{\sum_{j=1}^{k} \sum_{x \in C_j} d(x, X_j)}{n-k}}$$

One of the limitations of the internal quality measures is that they often use the same information both in discovering and in evaluating the clusters. Recall from **Heading 2.** that some clustering algorithms produce clustering results by optimizing various clustering criterion functions. If the same criterion functions were used as the internal quality measure, then the overall clustering assessment process does nothing more than *assessing* how effective the clustering algorithms was in optimizing the particular criterion function, and provides no independent confirmation about the degree to which the clusters are meaningful.

An alternative way for validating the clustering results is to see how stable they are when adding noise to the data, or subsampling it ***(66)***. This approach performs a sequence of subsamplings of the dataset and uses the same clustering procedure to produce clustering solutions for various subsamples. These various clustering results are then compared to see the degree to which they agree. The stable clustering solution should be the one that gives similar clustering results across the different subsamples. This approach can also be easily used to determine the *correct* number of clusters in hierarchical clustering solutions. The stability test of clustering is performed at each level of the hierarchical tree, and the number of clusters k will be the largest k value that still can produce stable clustering results.

Finally, a recent approach, with applications to clustering gene expression datasets, assesses the clustering results of gene expression data by looking at the predictive power for one experimental condition from the clustering results based on the other experimental conditions ***(67)***. The key idea behind this approach is that if one condition is left out, then the clusters generated from the remaining conditions should exhibit lower variation in the left-out condition than randomly formed clusters. Yeung et al. ***(67)*** defined the *figure of merit* (FOM) to be the summation of intracluster variance for each one of the clustering instances in which one of the conditions was not used during cluster (i.e., left-out condition). Among the various clustering solutions, they prefer the one that exhibits the least variation, and their experiments showed that in the context of clustering gene expression data, this method works quite well. The limitation of this approach is that it is not applicable to dataset in which all the attributes are independent. Moreover, this approach is only applicable to low dimensional datasets, as computing the intracluster variance for each dimension is quite expensive when the number of dimensions is very large.

5. Case Study: Clustering Gene Expression Data

Recently developed methods for monitoring genomewide mRNA expression changes such as oligonucleotide chips ***(68)*** and cDNA microarrays ***(69)***, are especially powerful as they allow us to quickly and inexpensively monitor the expression levels of a large number of genes at different time points, for different conditions, tissues, and organ-

isms. Knowing when and under what conditions a gene or a set of genes is expressed often provides strong clues as to their biological role and function.

Clustering algorithms are used as an essential tool to analyze these datasets and provide valuable insight on various aspects of the genetic machinery. There are four distinct classes of clustering problems that can be formulated from the gene expression datasets, each addressing a different biological problem. The first problem focuses on finding *co-regulated genes* by grouping together genes that have similar expression profiles. These co-regulated genes can be used to identify promoter elements by finding conserved areas in their upstream regions. The second problem focuses on finding *distinctive tissue types* by grouping together tissues whose genes have similar expression profiles. These tissue groups can then be further analyzed to identify the genes that best distinguish the various tissues. The third clustering problem focuses on finding *common inducers* by grouping together conditions for which the expression profiles of the genes are similar. Finding such groups of common inducers will allow us to substitute different "trigger" mechanisms that still elicit the same response (e.g., similar drugs, or similar herbicides or pesticides). Finally, the fourth clustering problem focuses on finding *organisms that exhibit similar responses over a specified set of tested conditions*, by grouping together organisms for which the expression profiles of their genes (in an ortholog sense) are similar. This would allow us to identify organisms with similar responses to chosen conditions (e.g., microbes that share a pathway).

In the rest of this subheading we briefly review the approaches behind cDNA and oligonucleotide microarrays, and discuss various issues related to clustering such gene expression datasets.

5.1. Overview of Microarray Technologies

DNA microarrays measure gene expression levels by exploiting the preferential binding of complementary, single-stranded nucleic acid sequences. cDNA microarrays, developed at Stanford University, are glass slides, to which single-stranded DNA molecules are attached at fixed locations (spots) by high-speed robotic printing ***(70)***. Each array may contain tens of thousands of spots, each of which corresponds to a single gene. mRNA from the sample and from control cells is extracted and cDNA is prepared by reverse transcription. Then, cDNA is labeled with two fluorescent dyes and washed over the microarray so that cDNA sequences from both populations hybridize to their complementary sequences in the spots. The amount of cDNA from both populations bound to a spot can be measured by the level of fluorescence emitted from each dye. For example, the sample cDNA is labeled with a red dye and the control cDNA is labeled with a green dye. Then, if the mRNA from the sample population is in abundance, the spot will be red; if the mRNA from the control population is in abundance, it will be green; if sample and control bind equally the spot will be yellow; if neither binds, it will appear black. Thus, the relative expression levels of the genes in the sample and control populations can be estimated from the fluorescent intensities and colors for each spot. After transforming the raw images produced by microarrays into relative fluorescent intensity with some image processing software, the gene expression levels are estimated as log-ratios of the relative intensities. A gene expression matrix can be formed by combining multiple microarray experiments of the same set of genes but under different conditions, where each row corresponds to a gene and each column corresponds to a condition (i.e., a microarray experiment) ***(70)***.

The Affymetrix GeneChip oligonucleotide array contains several thousand single-stranded DNA oligonucleotide probe pairs. Each probe pair consists of an element containing oligonucleotides that perfectly match the target (PM probe) and an element containing oligonucleotides with a single base mismatch (MM probe). A probe set consists of a set of probe pairs corresponding to a target gene. Similarly, the labeled RNA is extracted from sample cell and hybridizes to its complementary sequence. The expression level is measured by determining the difference between the PM and MM probes. Then, for each gene (i.e., probe

set) average difference or log average can be calculated, where average difference is defined as the average difference between the PM and MM of every probe pair in a probe set and log average is defined as the average log ratios of the PM/MM intensities for each probe pair in a probe set.

5.2. Data Preparation and Normalization

Many sources of systematic variation may affect the measured gene expression levels in microarray experiments ***(71)***. For the GeneChip experiments, scaling/normalization must be performed for each experiment before combining them together, so that they can have the same Target Intensity (TGT). The scaling factor of each experiment is determined by the array intensity of the experiment and the common TGT, where the array intensity is a composite of the average difference intensities across the entire array.

For cDNA microarray experiments, two fluorescent dyes are involved and cause more systematic variation, which makes normalization more important. In particular, this variation could be caused by differences in RNA amounts, differences in labeling efficiency between the two fluorescent dyes, and image acquisition parameters. Such biases can be removed by a constant adjustment to each experiment to force the distribution of the log-ratios to have a median of zero. Because an experiment corresponds to one column in the gene expression array, this global normalization can be done by subtracting the mean/median of the gene expression levels of one experiment from the original values, so that the mean value for this experiment (column) is zero.

However, there are other sources of systematic variation that global normalization may not be able to correct. Yang et al. ***(71)*** pointed out that dye biases can depend on spot overall intensity and location on the array. Given the red and green fluorescence intensities (R, G) of all the spots in one slide, they plotted the log intensity ratio $M = \log R/G$ vs the mean log-intensity $A = \log \sqrt{RG}$, which shows clear dependence of the log ratio M on overall spot intensity A. Hence, an intensity-related normalization was proposed, where the original log-ratio is subtracted by $C(A)$. $C(A)$ is a scatter-plot smoother fit to the M vs A plot using robust locally linear fits.

5.3. Similarity Measures

In most microarray clustering applications our goal is to find clusters of genes or clusters of conditions. A number of different methods have been proposed for computing these similarities, including Euclidean distance-based similarities, correlation coefficients, and mutual information.

The use of correlation coefficient-based similarities is primarily motivated by the fact that while clustering gene expression datasets we are interested on how the expression levels of different genes are related under various conditions. The correlation coefficient values between genes (Eq. 5) can be used directly or transformed to absolute values if genes of both positive and negative correlations are important in the application.

An alternate way of measuring the similarity is to use the mutual information between a pair of genes. The mutual information between two information sources A and B represent how much information the two sources contain for each other. D'Haeseleer et al. ***(72)*** used mutual information to define the relationship between two conditions A and B. This was done by initially discretizing the gene expression levels into various bins, and using this discretization to compute the Shannon entropy of conditions A and B as follows

$$S_A = -\sum_i p_i \log p_i$$

where p_i is the frequency of each bin. Given these entropy values, then the mutual information between A and B is defined as

$$M(A,B) = S_A + S_B - S_{A,B}$$

A feature common to many similarity measures used for microarray data is that they almost never consider the length of the corresponding gene or condition vectors, which is the actual value of the differential expression level, but focus only on various measures of relative change or how these relative measures are correlated between two genes or conditions ***(67,73,74)***. The reason for this is twofold. First, there is still significant experimental errors in measuring the expression level

of a gene, and is not reliable to use it "as is." Second, in most cases we are only interested on how the different genes change across the different conditions (i.e., either upregulated or downregulated) and we are not interested in the exact amount of this change.

5.4. Clustering Approaches for Gene Expression Data

Since the early days of the development of the microarray technologies, a wide range of existing clustering algorithms have been used, and novel new approaches have been developed for clustering gene expression datasets. The most effective traditional clustering algorithms are based either on the group-average variation of the agglomerative clustering methodology, or the *K*-means approach applied to unit-length gene or condition expression vectors. Unlike other applications of clustering in life sciences, such as the construction of phylogenetic trees, or guide trees for multiple sequence alignment, there is no biological reason that justifies that the structure of the correct clustering solution is in the form of a tree. Thus, agglomerative solutions are inherently suboptimal when compared to partitional approaches, which allow for a wider range of feasible solutions at various levels of cluster granularity. However, despite this, the agglomerative solutions tend to produce reasonable and biologically meaningful results, and allow for an easy visualization of the relationships between the various genes or conditions in the experiments.

The ease of visualizing the results has also led to the extensive use of self-organizing maps (SOM) for gene expression clustering ***(73,75)***. The SOM method starts with a geometry of "nodes" of a simple topology (e.g., grid and ring) and a distance function on the nodes. Initially, the nodes are mapped randomly into the gene expression space, in which the *i*th coordinate represents the expression level in the *i*th condition. At each following iteration, a data point *P* (i.e., a gene expression profile) is randomly selected and the data point *P* will attract nodes to itself. The nearest node N_p to *P* will be identified and moved the most, and other nodes will be adjusted depending on their distances to the nearest node N_p toward the data point *P*. The advantages of using SOMs are its structured approach, which makes visualization very easy. However, the method requires the user to specify the number of clusters as well as the grid topology, including the dimensions of the grid and the number of clusters in each dimension.

From the successes obtained in using *K*-means and group-average-based clustering algorithms, as well as other similar algorithms ***(76,77)***, it appears that the clusters in the context of gene expression datasets are globular in nature. This should not be surprising as researchers are often interested in obtaining clusters whose genes have similar expression patterns/profiles. Such a requirement automatically lends itself to globular clusters, in which the pairwise similarity between most object pairs is quite high. However, as the dimensionality of these datasets continue to increase (primarily by increasing the number of conditions that are analyzed), requiring consistency across the entire set of conditions will be unrealistic. Many new algorithms have been proposed recently to tackle the problem of clustering gene expression data with high dimensionality, among which global dimension reduction ***(78,79)*** and finding clusters in subspaces (i.e., subsets of dimensions) ***(80–84)*** are two widely used techniques.

The basic idea of global dimension reduction is to compress the entire gene/condition matrix to represent genes by vectors in a compressed space of low dimensionality, such that the biologically interesting results can be extracted ***(78,79,85)***. Horn and Axel ***(78)*** and Ding et al. ***(79)*** proposed to use singular value decomposition (SVD) to compress the data and then apply traditional clustering algorithms (such as *k*-means). They also showed that their algorithms can find meaningful clusters on cancer cells ***(86)***, leukemia dataset ***(87)***, and yeast cell cycle dataset ***(88)***. Similar ideas have been applied to other high-dimensional clustering problems, such as text clustering, and shown to be effective as well ***(89)***.

Finding clusters in subspaces tackle this problem differently by redefining the problem of clustering as finding clusters whose internal similarities become apparent in subspaces or clusters

Job: Molecular Biotechnology
Chapter: Karypis/MB05-0033
Pub Date: 2005
Operator: SV
Date: 6/05
Revision: 1st Pass

that preserve certain expression patterns among the dimensions in subspaces ***(80–84)***. The various algorithms differ from one another in how they model the desired clusters, the optimization algorithm and clustering algorithm that generate the desired clusters, and whether the algorithms allow genes that belong to more than one cluster (i.e., overlapping clusters). Cheng and Church ***(81)*** assume each expression value in the matrix is the addition of three components: the background level, the row effect, and the column effect. Thus, they use minimum mean squared residue as the objective function to find clusters in subspaces that have small deviations with respect to the rows in the cluster, the columns in the subspace, and the background defined by the cluster. The Plaid model ***(82)*** assumes the entire matrix is a sum of overlapping layers (i.e., clusters) and the global background, to better handle the overlapping clusters. Thus, the goal is to estimate the mean expression values of each layer and the probabilities of each entry in the matrix belonging to each layer, such that the calculated expression values are most consistent with the observed ones. Both algorithms ***(81,82)*** use a greedy algorithm to find clusters one by one. In particular, after finding one cluster, the entries that participate in the cluster are adjusted to eliminate the effect of the cluster. The algorithm will iterate several times until K (a user-defined parameter) clusters are found or no significant cluster can be found further. The two algorithms just mentioned treated rows and columns equally during the clustering process and investigators often refer to this type of clustering algorithms as biclustering ***(81–83)***. Another type of clustering algorithms finding clusters in subspaces (projected clustering algorithms; ***80***,***84***) treats the objects to be clustered and the relevant dimensions differently and often finds disjointed clusters. PROCLUS ***(80)*** is one of such algorithms. The whole clustering process is similar to the *k*-medoids method, which assigns each object to its closest cluster medoid. However, when calculating the distance between the object to its medoid, PROCLUS selects the most conserved l (a user-defined parameter) dimensions for each medoid based on its nearest neighbors and only uses this set of dimensions to calculate the distances. The HARP algorithm is an agglomerative projected clustering algorithm ***(84)***. It evaluates the quality of a cluster as the sum of the relevance index of each dimension, where the relevance index is defined by comparing the local variance (i.e., among the objects in the cluster) with the global variance of that dimension. The dimensions that have high relevance index values are the "signature" dimensions for identifying the members in the cluster. It will merge the two clusters that can result in a cluster with the highest relevance index sum. In the end, each cluster can be represented as the dimensions with highest relevance index values.

AU: necessary to spell out HARP? If so, please provide.

Finally, the clustering algorithms we discussed so far treat the dimensions (i.e., conditions) independently. In other words, if we permute the dimensions arbitrarily, the clustering result will remain the same. Sometimes, especially when the conditions are time points, methods that can capture the temporal relationships between the conditions are more suitable ***(90–93)***. Filkov et al. ***(90)*** showed that using correlation coefficient as the similarity measure can be misleading in time-series analysis and proposed to compare two genes by looking at the degree of agreement of slopes of the gene expression levels between two time points. Liu and Muller ***(91)*** applied the dynamic time-warping technique on gene expression data. There are also model-based clustering algorithms for temporal datasets, which assume each cluster as a stochastic model ***(92,93)***. Because the model-based clustering algorithm itself is a complicated topic and not the focus of this article, we will not discuss these algorithms in detail and readers can refer to ref. ***94*** for a comprehensive review.

6. Cluto: A Clustering Toolkit

We now turn our focus on providing a brief overview of CLUTO (release 2.1), a software package for clustering low- and high-dimensional datasets and for analyzing the characteristics of the various clusters, which has been developed by our group and is available at http://www.cs.umn.edu/~karypis/cluto. CLUTO has been developed as

Job: Molecular Biotechnology
Chapter: Karypis/MB05-0033
Pub Date: 2005
Operator: SV
Date: 6/05
Revision: 1st Pass

a general purpose clustering toolkit. CLUTO's distribution consists of both stand-alone programs (vcluster and scluster) for clustering and analyzing these clusters, as well as a library through which an application program can access directly the various clustering and analysis algorithms implemented in CLUTO. wCLUTO ***(95)***, a web-enabled version of the stand-alone application Cluto, designed to apply clustering methods to genomic information, is also available at http://cluto.ccgb.umn.edu. To date, CLUTO has been successfully used to cluster datasets arising in many diverse application areas including information retrieval, commercial datasets, scientific datasets, and biological applications.

CLUTO implements three different classes of clustering algorithms that can operate either directly in the object's feature space or in the object's similarity space. The clustering algorithms provided by Cluto are based on the partitional, agglomerative, and graph-partitioning paradigms. CLUTO's partitional and agglomerative algorithms are able to find clusters that are primarily globular, whereas its graph-partitioning and some of its agglomerative algorithms are capable of finding transitive clusters.

A key feature in most of CLUTO's clustering algorithms is that they treat the clustering problem as an optimization process that seeks to maximize or minimize a particular *clustering criterion function*, defined either globally or locally over the entire clustering solution space. CLUTO provides a total of seven different criterion functions that have been shown to produce high-quality clusters in low- and high-dimensional datasets. The equations of these criterion functions are shown in **Table 1**, and they were derived and analyzed ***(30,96)***. In addition to these criterion functions, CLUTO provides some of the more traditional local criteria (e.g., single-link, complete-link, and group-average) that can be used in the context of agglomerative clustering.

Table 1

An important aspect of partitional-based criterion-driven clustering algorithms is the method used to optimize this criterion function. CLUTO uses a randomized incremental optimization algorithm that is greedy in nature, has low computational requirements, and produces high-quality clustering solutions ***(30)***. Moreover, CLUTO's graph-partitioning-based clustering algorithms use high-quality and efficient multilevel graph-partitioning algorithms derived from the METIS and hMETIS graph- and hypergraph-partitioning algorithms ***(97,98)***. Moreover, CLUTO's algorithms have been optimized for operating on very large datasets, in terms of the number of objects, as well as in terms of the number of dimensions. This is especially true for CLUTO's algorithms for partitional clustering. These algorithms can quickly cluster datasets with several tens of thousands objects and several thousands of dimensions. Moreover, because most high-dimensional datasets are very sparse, CLUTO directly takes into account this scarcity and requires memory that is roughly linear on the input size.

In the rest of this section we present a short description of CLUTO's stand-alone programs followed by some illustrative examples of how it can be used for clustering biological datasets.

6.1. Usage Overview

The vcluster and scluster programs are used to cluster a collection of objects into a predetermined number of clusters *k*. The vcluster program treats each object as a vector in a high-dimensional space, and it computes the clustering solution using one of five different approaches. Four of these approaches are *partitional* in nature, whereas the fifth approach is *agglomerative*. On the other hand, the scluster program operates on the similarity space between the objects but can compute the overall clustering solution using the same set of five different approaches.

Both the vcluster and scluster programs are invoked by providing two required parameters on the command line along with a number of optional parameters. Their overall calling sequence is as follows:

vcluster [optional parameters] *MatrixFile NClusters*
scluster [optional parameters] *GraphFile NClusters*

Table 1
Mathematical Definition of CLUTO's Clustering Criterion Functions[a]

Criterion function	Optimization function	
I_1	$\text{maximize} \sum_{i=1}^{k} \frac{1}{n_i} \left(\sum_{v,u \in S_i} \text{sim}(v,u) \right)$	(6)
I_2	$\text{maximize} \sum_{i=1}^{k} \sqrt{\sum_{v,u \in S_i} \text{sim}(v,u)}$	(7)
$\mathcal{E}_1$	$\text{minimize} \sum_{i=1}^{k} n_i \frac{\sum_{v \in S_i, u \in S} \text{sim}(v,u)}{\sqrt{\sum_{v,u \in S_i} \text{sim}(v,u)}}$	(8)
$\mathcal{G}_1$	$\text{minimize} \sum_{i=1}^{k} \frac{\sum_{v \in S_i, u \in S} \text{sim}(v,u)}{\sum_{v,u \in S_i} \text{sim}(v,u)}$	(9)
$\mathcal{G}'_1$	$\text{minimize} \sum_{i=1}^{k} n_i^2 \frac{\sum_{v \in S_i, u \in S} \text{sim}(v,u)}{\sum_{v,u \in S_i} \text{sim}(v,u)}$	(10)
$\mathcal{H}_1$	$\text{maximize} \frac{I_1}{\mathcal{E}_1}$	(11)
$\mathcal{H}_2$	$\text{maximize} \frac{I_2}{\mathcal{E}_1}$	(12)

[a]The notation in these equations is as follows: k is the total number of clusters, S is the total objects to be clustered, S_i is the set of objects assigned to the ith cluster, n_i is the number of objects in the ith cluster, v and u represent two objects, and sim(v, u) is the similarity between two objects.

MatrixFile is the name of the file that stores the *n* objects that need to be clustered. In vcluster, each of these objects is considered to be a vector in an *m*-dimensional space. The collection of these objects is treated as an $n \times m$ matrix, whose rows correspond to the objects, and whose columns correspond to the dimensions of the feature space. Similarly, *GraphFile* is the name of the file that stores the adjacency matrix of the similarity graph between the *n* objects to be clustered. The second argument for both programs, *NClusters*, is the number of clusters that is desired.

Fig 1

Figure 1 shows the output of vcluster for clustering a matrix into 10 clusters. From this figure we see that vcluster initially prints information about the matrix, such as its name, the number of rows (*#Rows*), the number of columns (*#Columns*), and the number of non-zeros in the matrix (*#NonZeros*). Next, it prints information about the values of the various options that it used to compute the clustering, and the number of desired clusters (*#Clusters*). Once it computes the clustering solution, it displays information regarding the quality of the overall clustering solution, as well as the quality of each cluster, using a variety of internal quality measures. These measures include the average pairwise similarity between each object of the cluster and its standard deviation (ISim and ISdev), and the average similarity between the objects of each cluster to the objects in the other clusters and their standard deviation (ESim and ESdev). Finally, vcluster reports the time taken by the various phases of the program.

6.2. Summary of Biological Relevant Features

The behavior of vcluster and scluster can be controlled by specifying more than 30 different optional parameters. These parameters can be broadly categorized into three groups. The first

```
prompt% vcluster sports.mat 10
*******************************************************************************
vcluster (CLUTO 2.0) Copyright 2001-02, Regents of the University of Minnesota

Matrix Information ----------------------------------------------------------
  Name: sports.mat, #Rows: 8580, #Columns: 126373, #NonZeros: 1107980

Options ---------------------------------------------------------------------
  CLMethod=RB, CRfun=I2, SimFun=Cosine, #Clusters: 10
  RowModel=None, ColModel=IDF, GrModel=SY-DIR, NNbrs=40
  Colprune=1.00, EdgePrune=-1.00, VtxPrune=-1.00, MinComponent=5
  CSType=Best, AggloFrom=0, AggloCRFun=I2, NTrials=10, NIter=10

Solution --------------------------------------------------------------------

------------------------------------------------------------------------
10-way clustering: [I2=2.29e+03] [8580 of 8580]
------------------------------------------------------------------------
cid  Size  ISim  ISdev   ESim  ESdev  |
------------------------------------------------------------------------
  0   364 +0.166 +0.050 +0.020 +0.005 |
  1   628 +0.106 +0.041 +0.022 +0.007 |
  2   793 +0.102 +0.036 +0.018 +0.006 |
  3   754 +0.100 +0.034 +0.021 +0.006 |
  4   845 +0.095 +0.035 +0.023 +0.007 |
  5   637 +0.079 +0.036 +0.022 +0.008 |
  6  1724 +0.059 +0.026 +0.022 +0.007 |
  7   703 +0.049 +0.018 +0.016 +0.006 |
  8  1025 +0.054 +0.016 +0.021 +0.006 |
  9  1107 +0.029 +0.010 +0.017 +0.006 |
------------------------------------------------------------------------

Timing Information ----------------------------------------------------------
   I/O:                                 0.920 sec
   Clustering:                         12.440 sec
   Reporting:                           0.220 sec
*******************************************************************************
```

Fig. 1. Output of vcluster for matrix *sports.mat* and a 10-way clustering.

group controls various aspects of the clustering algorithm, the second group controls the type of analysis and reporting that is performed on the computed clusters, and the third set controls the visualization of the clusters. Some of the most important parameters are shown in **Table 2**, and are described in the context of clustering biological datasets in the rest of this section.

Table 2

6.3. Clustering Algorithms

The *-clmethod* parameter controls the type of algorithms to be used for clustering. The first two methods (i.e., "rb" and "direct") follow the partitional paradigm described in **Subheading 2.1**. The difference between them is the method they use to compute the *k*-way clustering solution. In the case of "rb", the *k*-way clustering solution is computed by a sequence of repeated bisections, whereas in the case of "direct", the entire *k*-way clustering solution is computed at one step. CLUTO's traditional agglomerative algorithm is implemented by the "agglo" option, whereas the "graph" option implements a graph-partitioning-based clustering algorithm, which is well-suited for finding transitive clusters. The method used to define the similarity between the objects is specified by the *-sim* parameter, and supports the cosine ("cos"), correlation coefficient ("corr"), and a Euclidean distance-derived similarity ("dist"). The clustering criterion function that is used by the partitional and agglomerative algorithms is controlled by the *-crfun* parameter. The first seven criterion functions (described in **Table 1**) are used by both partitional and agglomerative,

Job: Molecular Biotechnology
Chapter: Karypis/MB05-0033
Pub Date: 2005
Operator: SV
Date: 6/05
Revision: 1st Pass

Table 2
Key Parameters of Cluto's Clustering Algorithms

Parameter	Values	Function
-clmethod	rb, direct, agglo, graph	Clustering Method
-sim	cos, corr, dist	Similarity measures
-crfun	I_1, I_2, E_1, G_1, G'_1, H_1, H_2, slink, wslink, clink, wclink, upgma	Criterion Function
-agglofrom	(int)	Where to start agglomeration
-fulltree		Builds a tree within each cluster
-showfeatures		Display cluster's feature signature
-showtree		Build a tree on top of clusters
-labeltree		Provide key features for each tree node
-plottree	(filename)	Plots the agglomerative tree
-plotmatrix	(filename)	Plots the input matrices
-plotclusters	(filename)	Plots cluster–cluster matrix
-clustercolumn		Simultaneously cluster the features

whereas the last five (single-link, weighted-single-link, complete-link, weighted-complete-link, and group-average) are only applicable to agglomerative clustering.

A key feature of CLUTO is that allows you to combine partitional and agglomerative clustering approaches. This is done by the *-agglofrom* parameter in the following way. The desired *k*-way clustering solution is computed by first clustering the dataset into clusters ($m > k$), and then uses an agglomerative algorithm to group some of these clusters to form the final *k*-way clustering solution. The number of clusters *m* is the value supplied to *-agglofrom*. This approach was motivated by the two-phase clustering approach of the CHAMELEON algorithm ***(20)***, and was designed to allow the user to compute a clustering solution that uses a different clustering criterion function for the partitioning phase from that used for the agglomeration phase. An application of such an approach is to allow the clustering algorithm to find non-globular clusters. In this case, the partitional clustering solution can be computed using a criterion function that favors globular clusters (e.g., "i2"), and then combine these clusters using a single-link approach (e.g., "wslink") to find nonglobular but well-connected clusters.

6.4. Building Tree for Large Datasets

Hierarchical agglomerative trees are used extensively in life sciences as they provide an intuitive way to organize and visualize the clustering results. However, there are two limitations with such trees. First, hierarchical agglomerative clustering may not be the *optimal* way to cluster data in which there is no biological reason to suggest that the objects are related with each other in a tree-fashion. Second, hierarchical agglomerative clustering algorithms have high computational and memory requirements, making them impractical for datasets with more than a few thousand objects.

To address these problems CLUTO provides the *-fulltree* option that can be used to produce a complete tree using a hybrid of partitional and agglomerative approaches. In particular, when *-fulltree* is specified, CLUTO builds a complete hierarchical tree that preserves the clustering solution that was computed. In this hierarchical clustering solution, the objects of each cluster form a subtree, and the different subtrees are merged to get an all-inclusive cluster at the end. Furthermore, the individual trees are combined in a meaningful way, therefore to accurately represent the similarities within each tree.

Job: Molecular Biotechnology
Chapter: Karypis/MB05-0033
Pub Date: 2005
Operator: SV
Date: 6/05
Revision: 1st Pass

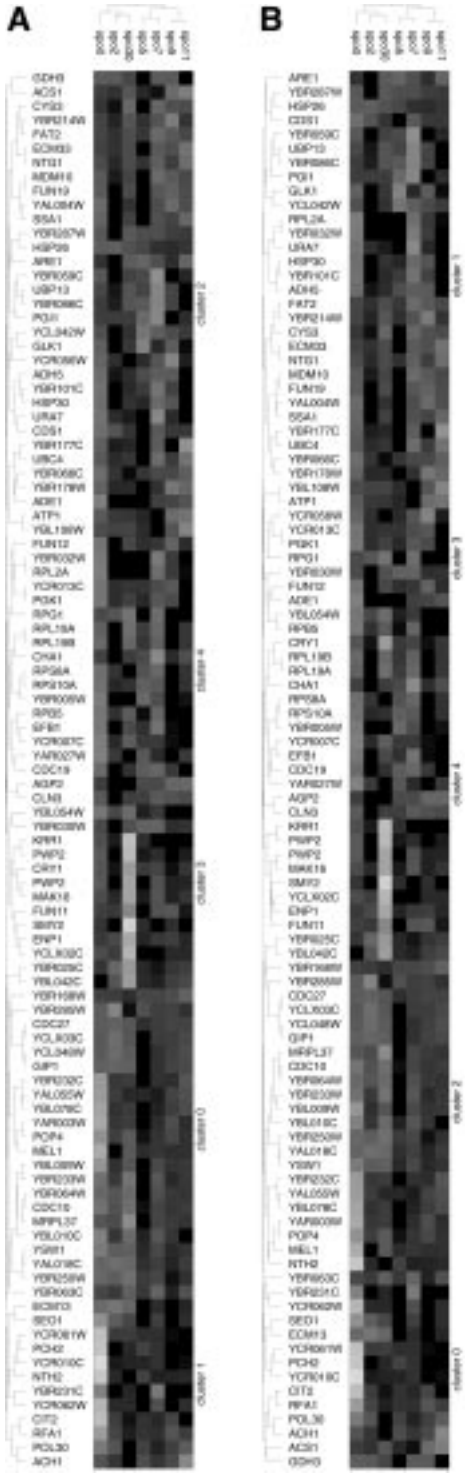

Fig. 2. (**A**) The clustering solution produced by the agglomerative method. (**B**) The clustering solution produced by the repeated-bisecting method and *–fulltree*.

Figure 2 shows the trees produced on a sample gene expression dataset. The first tree was obtained using the agglomerative clustering algorithm, whereas the second tree was obtained using the repeated-bisecting method in which the *-fulltree* was specified.

Fig 2

6.5. Analyzing the Clusters

In addition to the core clustering algorithms, CLUTO provides tools to analyze each of the clusters and identify what are the features that best describe and discriminate each one of the clusters. To some extent, these analysis methods try to identify the dense subspaces in which each cluster is formed. This is accomplished by the *-showfeatures* and *-labeltree* parameters.

Figure 3 shows the output produced by vcluster when *-showfeatures* was specified for a dataset consisting of protein sequences and the 5mers that they contain. Looking at this figure, we can see that the set of descriptive and discriminating features are displayed right after the table that provides statistics for the various clusters. For each cluster, vcluster displays three lines of information. The first line contains some basic statistics for each cluster corresponding to the cluster-id (cid), number of objects in each cluster (Size), the average pairwise similarity between the cluster's objects (ISim), and the average pairwise similarity to the rest of the objects (ESim). The second line contains the five most descriptive features, whereas the third line contains the five most discriminating features. The features in these lists are sorted in decreasing descriptive or discriminating order.

Fig 3

Right next to each feature, vcluster displays a number that in the case of the descriptive features is the percentage of the within-cluster similarity that this particular feature can explain. For example, for the 0th cluster, the 5mer "GTSMA" explains 58.5% of the average similarity between the objects of the 0th cluster. A similar quantity is displayed for each one of the discriminating features, and is the percentage of the dissimilarity between the cluster and the rest of the objects that this feature can explain. In general there is a large overlap between descriptive and discriminating features, with the only difference being that the

```
*******************************************************************************
vcluster (CLUTO 2.0) Copyright 2001-02, Regents of the University of Minnesota

Matrix Information ------------------------------------------------------------
Name: peptide5.mat, #Rows: 1539, #Columns: 2965, #NonZeros: 50136

Options -----------------------------------------------------------------------
  CLMethod=RB, CRfun=I2, SimFun=Cosine, #Clusters: 10
  RowModel=None, ColModel=IDF, GrModel=SY-DIR, NNbrs=40
  Colprune=1.00, EdgePrune=-1.00, VtxPrune=-1.00, MinComponent=5
  CSType=Best, AggloFrom=0, AggloCRFun=I2, NTrials=10, NIter=10

Solution ----------------------------------------------------------------------

------------------------------------------------------------------------------
10-way clustering: [I2=5.03e+02] [1539 of 1539], Entropy: 0.510, Purity: 0.638
------------------------------------------------------------------------------
cid  Size  ISim  ISdev   ESim  ESdev  Entpy Purty |    1    2    3    4    5
------------------------------------------------------------------------------
  0    50 +0.439 +0.139 +0.001 +0.001 0.061 0.980 |    0    0    1   49    0
  1   109 +0.275 +0.113 +0.019 +0.010 0.156 0.945 |    0    3    3  103    0
  2    99 +0.246 +0.089 +0.001 +0.001 0.096 0.970 |    0    1   96    2    0
  3   227 +0.217 +0.074 +0.010 +0.005 0.129 0.956 |    0    3    7  217    0
  4   121 +0.197 +0.092 +0.001 +0.001 0.258 0.893 |    0  108    6    7    0
  5   127 +0.113 +0.064 +0.001 +0.001 0.425 0.780 |   99    5   21    2    0
  6   112 +0.109 +0.101 +0.001 +0.001 0.553 0.661 |    1   74   12   25    0
  7   202 +0.043 +0.040 +0.001 +0.002 0.741 0.421 |    4    6   78   85   29
  8   268 +0.029 +0.029 +0.001 +0.001 0.880 0.302 |   81   47   81   55    4
  9   224 +0.027 +0.024 +0.001 +0.001 0.860 0.312 |    1   57   70   37   59
------------------------------------------------------------------------------
------------------------------------------------------------------------------
10-way clustering solution - Descriptive & Discriminating Features...
------------------------------------------------------------------------------
Cluster   0, Size:    50, ISim: 0.439, ESim: 0.001
      Descriptive:  GTSMA 58.5%, HGTHV 37.7%, PSTVV  0.4%, LGASG  0.4%, KELKK  0.3%
   Discriminating:  GTSMA 29.6%, HGTHV 19.1%, GDSGG  2.7%, DSGGP  2.5%, TAAHC  2.0%

Cluster   1, Size:   109, ISim: 0.275, ESim: 0.019
      Descriptive:  SGGPL  8.6%, RPYMA  7.6%, VLTAA  7.2%, TAAHC  6.9%, DSGGP  6.3%
   Discriminating:  RPYMA  5.5%, PHSRP  4.3%, SRPYM  4.3%, HSRPY  3.9%, KGDSG  3.3%

Cluster   2, Size:    99, ISim: 0.246, ESim: 0.001
      Descriptive:  HEFGH 13.2%, CGVPD  6.5%, RCGVP  6.3%, PRCGV  6.1%, VAAHE  5.3%
   Discriminating:  HEFGH  6.8%, CGVPD  3.3%, RCGVP  3.2%, PRCGV  3.1%, GDSGG  2.9%

Cluster   3, Size:   227, ISim: 0.217, ESim: 0.010
      Descriptive:  GDSGG  8.3%, DSGGP  6.7%, CQGDS  5.7%, QGDSG  5.6%, TAAHC  4.4%
   Discriminating:  CQGDS  3.5%, QGDSG  3.3%, NSPGG  2.6%, GDSGG  2.3%, CGGSL  2.1%

Cluster   4, Size:   121, ISim: 0.197, ESim: 0.001
      Descriptive:  CGSCW 13.9%, GSCWA 10.2%, SCWAF  8.4%, KNSWG  5.3%, GCNGG  5.0%
   Discriminating:  CGSCW  7.1%, GSCWA  5.2%, SCWAF  4.3%, GDSGG  2.9%, KNSWG  2.7%

Cluster   5, Size:   127, ISim: 0.113, ESim: 0.001
      Descriptive:  DTGSS  6.0%, FDTGS  4.0%, TGSSD  3.0%, IGTPP  2.7%, GTPPQ  2.6%
   Discriminating:  DTGSS  3.1%, GDSGG  2.8%, DSGGP  2.6%, TAAHC  2.0%, SGGPL  2.0%

Cluster   6, Size:   112, ISim: 0.109, ESim: 0.001
      Descriptive:  KDELR  2.2%, IEASS  1.6%, RWAVL  1.6%, TFLKR  1.4%, EEKIK  1.3%
   Discriminating:  GDSGG  2.8%, DSGGP  2.6%, TAAHC  2.0%, SGGPL  2.0%, LTAAH  1.4%

Cluster   7, Size:   202, ISim: 0.043, ESim: 0.001
      Descriptive:  NSPGG 46.9%, HELGH 12.1%, ALLEV  7.0%, VLAAA  2.6%, GYVDA  1.7%
   Discriminating:  NSPGG 24.5%, HELGH  5.1%, ALLEV  3.6%, GDSGG  2.9%, DSGGP  2.6%

Cluster   8, Size:   268, ISim: 0.029, ESim: 0.001
      Descriptive:  QACRG 13.7%, IQACR  7.5%, DTGAD  2.9%, VDTGA  2.5%, LDTGA  1.2%
   Discriminating:  QACRG  7.1%, IQACR  3.9%, GDSGG  2.9%, DSGGP  2.6%, TAAHC  2.1%

Cluster   9, Size:   224, ISim: 0.027, ESim: 0.001
      Descriptive:  LAAIA  4.3%, TDNGA  3.0%, LKTAV  1.4%, TQYGG  1.1%, GFRRL  1.1%
   Discriminating:  GDSGG  2.9%, DSGGP  2.6%, LAAIA  2.2%, TAAHC  2.1%, SGGPL  2.0%
------------------------------------------------------------------------------

Timing Information ------------------------------------------------------------
   I/O:                          0.040 sec
   Clustering:                   0.470 sec
   Reporting:                    0.040 sec
*******************************************************************************
```

Fig. 3. Output of vcluster for matrix *sports.mat* and a 10-way clustering that shows the descriptive and discriminating features of each cluster.

Job: Molecular Biotechnology
Chapter: Karypis/MB05-0033
Pub Date: 2005
Operator: SV
Date: 6/05
Revision: 1st Pass

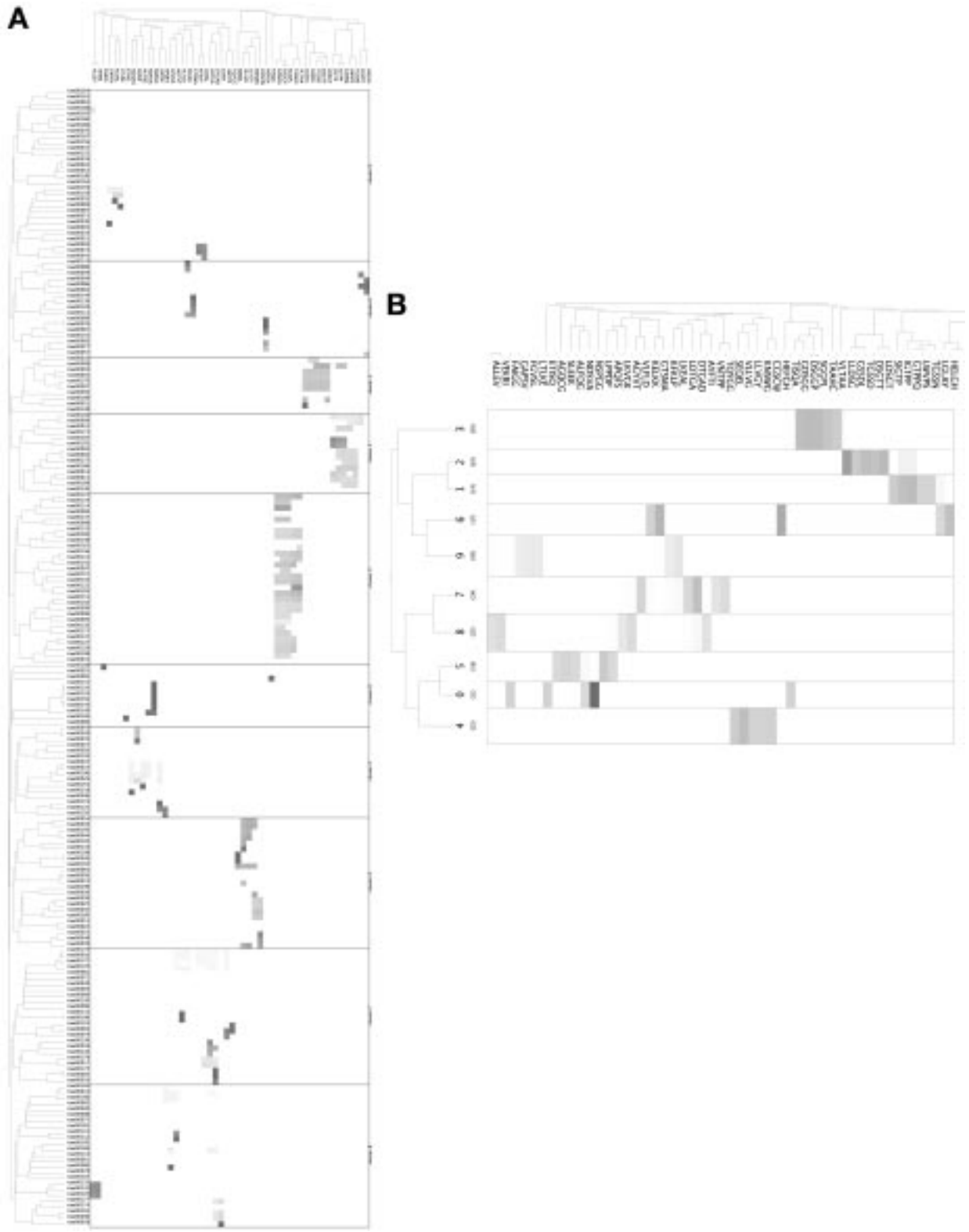

Fig. 4. Various visualizations generated by the-*plotmatrix* (a) and -*plotcluster* (b) parameter.

percentages associated with the discriminating features are typically smaller than the corresponding percentages of the descriptive features. This is because some of the descriptive features of a cluster may also be present in a small fraction of the objects that do not belong to the cluster.

6.6. Visualizing the Clusters

Fig 4

CLUTO's programs can produce a number of visualizations that can be used to see the relationships between the clusters, objects, and features. You have already seen one of them in **Fig. 2** that was produced by the *-plotmatrix* parameter. The same parameter can be used to visualize sparse high-dimensional datasets. This is illustrated in **Fig. 4A** for the protein dataset used earlier. As we can see from that plot, vcluster shows the rows of the input matrix reordered in such a way so that the rows assigned to each one of the 10 clusters are numbered consecutively. The columns of the displayed matrix are selected to be the union of the most descriptive and discriminating features of each cluster, and are ordered according to the tree produced by an agglomerative clustering of the columns. Also, at the top of each column, the label of each feature is shown. Each non-zero positive element of the matrix is displayed by a different shade of red. Entries that are bright red correspond to large values and the brightness of the entries decreases as their value decrease. Also note that in this visualization both the rows and columns have been reordered using a hierarchical tree.

Finally, **Fig. 4B** shows the type of visualization that can be produced when *-plotcluster* is specified for a sparse matrix. This plot shows the clustering solution shown at **Fig. 4A** by replacing the set of rows in each cluster by a single row that corresponds to the centroid vector of the cluster. The *-plotcluster* option is particularly useful for displaying very large datasets, as the number of rows in the plot is only equal to the number of clusters.

7. Future Research Directions in Clustering

Despite the huge body of research in cluster analysis, there are a number of open problems and research opportunities, especially in the context of clustering datasets arising in life sciences. Existing clustering algorithms for sequence and structure datasets operate on the object's similarity space. As discussed in **Subheading 2.2.**, such algorithms are quite limiting as they cannot scale to very large datasets, cannot be used to find clusters that have conserved features (e.g., sequence or structural motifs), and cannot be used to provide a description as to why a set of objects was assigned to the same cluster that is native to the object's features. The only way to overcome these shortcomings is to develop algorithms that operate directly on either the sequences or structures. Thus, opportunities for future research can be broadly categorized into three groups: (1) development of computationally efficient and scalable algorithms for large sequence and structure datasets; (2) development of clustering algorithms for sequence and structure datasets that operate directly on the object's native representation; and (3) development of clustering algorithms that can provide concise explanations on the characteristics of the objects that were assigned to each cluster.

References

1. Linial, M., Linial, N., Tishby, J., and Golan, Y. (1997) Global self-organization of all known protein sequences reveals inherent biological structures. *J. Mol. Biol.* **268,** 539–556.
2. Enright, A. J. and Ouzounis, C. A. (2000) GeneRAGE: a robust algorithm for sequence clustering and domain detection. *Bioinformatics* **16(5),** 451–457.
3. Mian, I. S. and Dubchak, I. (2000) Representing and reasoning about protein families using generative and discriminative methods. *J. Comput. Biol.* **7(6),** 849–862.
4. Spang, R. and Vingron, M. (2001) Limits of homology of detection by pairwise sequence comparison. *Bioinformatics* **17(4),** 338–342.
5. Kriventseva, E., Biswas, M., and Apweiler, R. (2001) Clustering and analysis of protein families. *Curr. Opin. Struct. Biol.* **11,** 334–339.
6. Eidhammer, I., Jonassen, I., and Taylor, W. R. (2000) Structure comparison and stucture patterns. *J. Comput. Biol.* **7,** 685–716.
7. Shatsky, M., Fligelman, Z. Y., Nussinov, R., and Wolfson, H. J. (2000) Alignment of flexible protein structures, in *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, (Altman, R., et al., eds.), AAAI Press, Menlo Park, CA, pp. 329–343.

8. Lee, R. C. T. (1981) Clustering analysis and its applications, in *Advances in Information Systems Science* (Toum, J. T., ed.), Plenum, New York.
9. Jain, A. K. and Dubes, R. C. (1988) *Algorithms for Clustering Data*, Prentice Hall, 1998.
10. Cheeseman, P. and Stutz, J. (1996) Bayesian classification (autoclass): theory and results, in *Advances in Knowledge Discovery and Data Mining* (Fayyad, U. M., Piatetsky-Shapiro, G., Smith, P., and Uthurusamy, R., eds.), AAAI/MIT Press, pp. 153–180.
11. Kaufman, L. and Rousseeuw, P. J. (1990) *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, New York.
12. Jackson, J. E. (1991) *A User's Guide to Principal Components*, John Wiley & Sons, New York.
13. Ng, R. and Han, J. (1994) Efficient and effective clustering method for spatial data mining, in *Proceedings of the 20th VLDB Conference* (Bocca, J. B., Jarke, M., and Zaniolo, C., eds.), San Francisco, CA, pp. 144–155.
14. Berry, M. W., Dumais, S. T., and O'Brien, G. W. (1995) Using linear algebra for intelligent information retrieval. *SIAM Rev.* **37,** 573–595.
15. Zhang, T., Ramakrishnan, R., and Linvy, M. (1996) Birch: an efficient data clustering method for large databases, in *Proceedings of 1996 ACM-SIGMOD International Conference on Management of Data*, ACM Press, New York, NY, pp. 103–114.
16. Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996) A density-based algorithm for discovering clusters in large spatial databases with noise, in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (Simoudis, E., Han, J., and Fayyad, U., eds.), AAAI Press, Menlo Park, CA, pp. 226–231.
17. Wang, X., Wang, J. T. L., Shasha, D., Shapiro, B., Dikshitulu, S., Rigoutsos, I., and Zhang, K. (1997) Automated discovery of active motifs in three dimensional molecules, in *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining* (Heckerman, D., Mannila, H., Pregibon, D., and Uthurusamy, R., eds.), AAAI Press, Menlo Park, CA, pp. 89–95.
18. Guha, S., Rastogi, R., and Shim, K. (1998) CURE: an efficient clustering algorithm for large databases, in *Proceedings of 1998 ACM-SIGMOD International Conference on Management of Data*, ACM Press, New York, NY, pp. 73–84.
19. Guha, S., Rastogi, R., and Shim, K. (1999) ROCK: a robust clustering algorithm for categorical attributes, in *Proceedings of the 15th International Conference on Data Engineering*, IEEE, Washington-Brussels-Tokyo, pp. 512–521.
20. Karypis, G., Han, E. H., and Kumar, V. (1999) Chameleon: a hierarchical clustering algorithm using dynamic modeling. *IEEE Comput.* **32(8),** 68–75.
21. Jain, A. K., Murty, M. N., and Flynn, P. J. (1999) Data clustering: a review. *ACM Comput. Surv.* **31(3),** 264–323.
22. Han, J., Kamber, M., and Tung, A. K. H. (2001) Spatial clustering methods in data mining: a survey, in *Geographic Data Mining and Knowledge Discovery* (Miller, H. and Han, J., eds.), Taylor & Francis.
23. MacQueen, J. (1967) Some methods for classification and analysis of multivariate observations, in *Proceedings of the 5th Symposium Math. Statist. Prob.*, pp. 281–297.
24. Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977) Maximum likelihood from incomplete data via the em algorithm. *J. Roy. Stat. Soc.* **39**.
25. Zahn, K. (1971) Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. Comput.* **(C-20),** 68–86.
26. Han, E. G., Karypis, G., Kumar, V., and Mobasher, B. (1998) Hypergraph based clustering in high-dimensional data sets: a summary of results. *Bull. Tech. Committee Data Eng.* **21(1)**.
27. Strehl, A. and Ghosh, J. (2000) Scalable approach to balanced, high-dimensional clustering of market-baskets, in *Proceedings of HiPC*, Springer Verlag, pp. 525–536.
28. Boley, D. (1998) Principal direction divisive partitioning. *Data Mining Knowl. Discov.* **2(4),**
29. Duda, R. O., Hart, P. E., and Stork, D. G. (2001) *Pattern Classification*, John Wiley & Sons, New York.
30. Zhao, Y. and Karypis, G. (2001) Criterion functions for document clustering: experiments and analysis. Technical report TR #01–40, Department of Computer Science, University of Minnesota, Minneapolis. Available at http://cs.umn.edu/ ~karypis/publications.
31. Sneath, P. H. and Sokal, R. R. (1973) *Numerical Taxonomy*, Freeman, London, UK.
32. King, B. (1967) Step-wise clustering procedures. *J. Am. Stat. Assoc.* **69,** 86–101.
33. Johnson, M. S., Sutcliffe, M. J., and Blundell, T. L. (1990) Molecular anatomy: phyletic relationships derived from 3-dimensional structures of proteins. *J. Mol. Evol.* **30,** 43–59.
34. Taylor, W. R., Flores, T. P., and Orengo, C. A. (1994) Multiple protein structure alignment. *Protein Sci.* **3,** 1858–1870.
35. Jonyer, I., Holder, L. B., and Cook, D. J. (2000) Graph-based hierarchical conceptual clustering in structural databases. *Int. J. Artific. Intell. Tools.*
36. Eddy, S. R. (1996) Hidden markov models. *Curr. Opin. Struct. Biol.* **6,** 361–365.
37. Eddy, S. R. (1998) Profile hidden markov models. *Bioinformatics* **14,** 755–763.
38. Aggarwal, C. C., Gates, S. C., and Yu, P. S. (1999) On the merits of building categorization systems by supervised clustering, *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Chaudhuri, S. and

Madigan, D., eds.), ACM Press, New York, NY, pp. 352–356.
39. Cutting, D. R., Pedersen, J. O., Karger, D. R., and Tukey, J. W. (1992) Scatter/gather: a cluster-based approach to browsing large document collections, in *Proceedings of the ACM SIGIR*, ACM Press, New York, NY, pp. 318–329.
40. Larsen, B. and Aone, C. (1999) Fast and effective text mining using linear-time document clustering, in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Chaudhuri, S. and Madigan, D., eds.), ACM Press, New York, NY, pp. 16–22.
41. Steinbach, M., Karypis, G., and Kumar, V. (2000) A comparison of document clustering techniques, in *KDD Workshop on Text Mining*, ACM Press, New York, NY, pp. 109–110.
42. Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (1998) Automatic sub- space clustering of high dimensional data for data mining applications, in *Proceedings of 1998 ACM-SIGMOD International Conference on Management of Data*, ACM Press, New York, NY, pp. 94–105.
43. Burdick, D., Calimlim, M., and Gehrke, J. (2001) Mafia: a maximal frequent itemset algorithm for transactional databases, in *Proceedings of the 17th International Conference on Data Engineering*, IEEE, Washington-Brussels-Tokyo, pp. 443–452.
44. Nagesh, H., Goil, S., and Choudhary, A. (1999) Mafia: efficient and scalable subspace clustering for very large data sets. Technical report TR #9906-010, Northwestern University.
45. Huang, Z. (1997) A fast clustering algorithm to cluster very large categorical data sets in data mining, in *Proceedings of SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, Tucson, Arizona.
46. Ganti, V., Gehrke, J., and Ramakrishnan, R., (1999) Cactus-clustering categorical data using summaries, in *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining* (Chaudhuri, S. and Madigan, D., eds.), ACM Press, New York, NY, pp. 73–83.
47. Gibson, D., Kleinberg, J., and Rahavan, P. (1998) Clustering categorical data: an approach based on dynamical systems, in *Proceedings of the 24th International Conference on Very Large Databases* (Gupta, A., Shmueli, O., and Widom, J., eds.), pp. 311–323, San Francisco, CA, Morgan Kaufmann.
48. Ryu, T. W. and Eick, C. F. (1998) A unified similarity measure for attributes with set or bag of values for database clustering, in *The 6th International Workshop on Rough Sets, Data Mining and Granular Computing*, Research Triangle Park, NC.
49. Gusfield, D. (1997) *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, NY.
50. Dayhoff, M. O., Schwartz, R. M., and Orcutt, B. C. (1978) A model of evolutionary change in proteins. *Atlas Protein Sequence Struct.* **5,** 345–352.
51. Schwartz, R. M. and Dayhoff, M. O. (1978) Matrices for detecting distant relationships. *Atlas Protein Sequence Struct.* **5,** 353–358.
52. Henikoff, S. and Henikoff, J. G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA* **89,** 10,915–10,919.
53. Needleman, S. B. and Wunsch, C. D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* **48,** 443–453.
54. Smith, T. F. and Waterman, M. S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.* **147,** 195–197.
55. Lipman, D. J. and Pearson, W. R. (1985) Rapid and sensitive protein similarity searches. *Science* **227,** 1435–1441.
56. Pearson, W. R. and Lipman, D. J. (1988) Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA* **85,** 2444–2448.
57. Altschul, S., Gish, W., Miller, W., Myers, E., and Lipman, D. (1990) Basic local alignment search tool. *J. Mol. Biol.* **215,** 403–410.
58. Yona, G., Linial, N., and Linial, M. (2000) Protomap: automatic classification of protein sequences and hierarchy of protein families. *Nucleic Acids Res.* **28,** 49–55.
59. Bolten, E., Schliep, A., Schneckener, S., Schomburg, D., and Schrader, R. (2001) Clustering protein sequences-structure prediction by transitive homology. *Bioinformatics* **17,** 935–941.
60. Johnson, M. S. and Lehtonen, J. V. (2000) Comparison of protein three dimensional structure, in *Bioinformatics: Sequences, Structure and Databanks* (Higgins, D, eds.), Oxford University Press, 2000.
61. Grindley, H. M., Artymiuk, P. J., Rice, D. W., and Willet, P. (1993) Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm. *J. Mol. Biol.* **229,** 707–721.
62. Mitchell, E. M., Artymiuk, P. J., Rice, D. W., and Willet, P. (1989) Use of techniques derived from graph theory to compare secondary structure motifs in proteins. *J. Mol. Biol.* **212,** 151–166.
63. Holm, L. and Sander, C. (1993) Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.* **233,** 123.
64. Kleywegt, G. J. and Jones, T. A. (1997) Detecting folding motifs and similarities in protein structures. *Methods Enzymol.* **277,** 525–545.
65. Calinski, T. and Harabasz, J. (1974) A dendrite method for cluster analysis. *Commun. Stat.* **3,** 1–27.
66. Ben-Hur, A., Elisseeff, A., and Guyon, I. (2000) A stability based method for discovering structure in clustered data, in *Pacific Symposium on Biocomputing*, vol. 7, World Scientific Press, Singapore, pp. 6–17.

67. Yeung, K. Y., Haynor, D. R., and Ruzzo, W. L. (2001) Validating clustering for gene expression data. *Bioinformatics* **17(4),** 309–318.
68. Fodor, S. P., Rava, R. P., Huang, X. C., Pease, A. C., Holmes, C. P., and Adams, C. L. (1993) Multiplexed biochemical assays with biological chips. *Nature* **364,** 555, 556.
69. Schena, M., Shalon, D., Davis, R. W., and Brown, P. O. (1995) Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science* **270**.
70. Brazma, A., Robinson, A., Cameron, G., and Ashburner, M. (2000) One-stop shop for microarray data. *Nature* **403,** 699–701.
71. Yang, Y. H., Dudoit, S., Luu, P., and Speed, T. P. (2001) Normalization for cDNA microarray data, in *Proceedings of SPIE International Biomedical Optics Symposium* (Bittner, M. L., Chen, Y., Dorsel, A. N., and Dougherty, E. R., eds.), vol. 4266, SPIE, Bellingham, WA, pp. 141–152.
72. D'Haeseleer, P., Fuhrman, S., Wen, X., and Somogyi, R. (1998) Mining the gene expression matrix: inferring gene relationships from large scale gene expression data, in, *Information Processing in Cells and Tissues*, Plenum, New York, pp. 203–212.
73. Eisen, M. (2000) Cluster 2.11 and treeview 1.50 http://rana.lbl.gov/Eisen Software.htm.
74. Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E. S., and Golub, T. R. (1999) Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *Proc. Natl. Acad. Sci. USA* **96,** 2907–2912.
75. Kohonen, T. *Self-Organizing Maps*, Springer-Verlag, Berlin, 1995.
76. Ben-Dor, A. and Yakhini, Z. (1999) Clustering gene expression patterns, in *Proceedings of the 3rd Annual International Conference on Research in Computational Molecular Biology*, ACM Press, New York, NY, pp. 33–42.
77. Shamir, R. and Sharan, R. (2000) Click: a clustering algorithm for gene expression analysis, in *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology* (Altman, R., et al., eds.), AAAI Press, Menlo Park, CA, pp. 307–316.
78. Zhao, Y. and Karypis, G. (2002) Comparison of agglomerative and partitional document clustering algorithms, in *SIAM (2002) Workshop on Clustering High-Dimensional Data and Its Applications*. Arlington, VA. Also available as technical report #02-014, University of Minnesota.
79. Karypis, G. and Kumar, V. (1998) hMeTiS 1.5: a hypergraph partitioning package. Technical report, Department of Computer Science, University of Minnesota. Available at www-users.cs.umn.edu/~karypis/metis
80. Karypis, G. and Kumar, V. (1998) MeTiS 4.0: unstructured graph partitioning and sparse matrix ordering system. Technical report, department of Computer Science, University of Minnesota. Available at www-users.cs.umn.edu/~karypis/metis