Chapter 16

Clustering Biological Data

Chandan K. Reddy
Wayne State University
Detroit, MI
reddy@cs.wayne.edu

Mohammad Al Hasan
Indiana University - Purdue University
Indianapolis, IN
alhasan@cs.iupui.edu

Mohammed J. Zaki
Rensselaer Polytechnic Institute
Troy, NY
zaki@cs.rpi.edu

16.1 Introduction

With the advancement of recent technologies, a vast amount of biological data is being generated. As data banks increase their size, one of the current challenges in biology is to be able to infer some of the critical functions from such complex data. To analyze complex biological systems, researchers usually aim to identify some patterns that co-occur in the form of groups. Clustering analysis is an exploratory technique that discovers rich patterns from vast data, and hence, it has become an indispensable tool for various knowledge discovery tasks in the field of computational biology. Clustering is a powerful and widely used technique that organizes and elucidates the structure of biological data. Clustering data from a wide variety of biological experiments has proven to be immensely useful at deriving a variety of insights, such as the shared regulation or function of genes.

In analyzing this complex biological data, one can observe that the activities of genes are not independent of each other. It has been shown that genes with the same function (or genes involved in the same biological process) are likely to be co-expressed [56]. Hence, it is important to study groups of genes rather than to perform a single gene analysis. In other words, it is crucial to identify subsets of genes that are relevant to the biological problem under study. Analyzing such subsets of data yields crucial information about the biological processes and the cellular functions. Thus, *clustering the gene expression profiles* can provide insights into gene function, gene regulation, and cellular processes.

It has also been shown that proteins of known functions tend to cluster together [92]. The network distance is correlated with functional distance, and the proteins that are closer to one another tend to have similar biological functions [96]. Hence, *clustering the protein–protein interaction networks* is crucial in discovering the functions of proteins and thus understanding the inner workings of cells [84]. The most important building blocks of living organisms, such as DNA, RNA, mRNA, polypeptides, and proteins have linear structure and can be represented as sequences. *Clustering biological sequence data* aims to group together the biological sequences that are related. The identified clusters can help in providing a better understanding of the genome.

This chapter comprehensively reviews different kinds of biological data where clustering has provided promising and biologically meaningful results. More specifically, we will discuss the role of clustering for gene expression data, biological networks, and sequence data. For each type of data, the challenges and the most prominent clustering algorithms that have been successfully studied will be described. The rest of this chapter is organized as follows. Section 16.2 describes various types of clustering and the corresponding state-of-the-art clustering techniques for each category in the context of microarray data analysis. Section 16.3 provides details about several protein interaction network clustering algorithms. Section 16.4 describes the state-of-the-art biological sequence clustering algorithms. Software packages that implement most of the popular biological clustering algorithms are discussed in Section 16.5. Finally, Section 16.6 concludes our discussion.

16.2 Clustering Microarray Data

The recent advances in DNA microarray technology allow genome-wide expression profiling. It has revolutionized the analysis of genes and proteins and has made it possible to simultaneously measure the expression levels of tens of thousands of genes. The expression level of a gene is a measurement for the frequency with which the gene is expressed, and it can be used to estimate the current amount of the protein in a cell for which the gene codes [58]. The availability of such massive data has transformed the field of gene expression analysis [18].

Gene expression data clustering provides a powerful tool for studying functional relationships of genes in a biological process. Identifying correlated expression patterns of genes represents the basic challenge in this clustering problem. The underlying hypothesis here is based on a popular phenomenon known as *guilt-by-association* principle which states that genes with similar functions exhibit similar expression patterns (they are co-expressed together) [109, 25]. Hence, it becomes critical to study the relationships between the genes among various biological conditions. Clustering methods allow the biologists to capture the relationships between genes and identify the co-expressed genes in a given microarray study. Clustering also plays a critical role in other related biological applications. In addition to clustering the genes, there is also some research work on clustering the conditions to identify phenotype subtypes [44]. Clustering can also be used to extract regulatory motifs from the gene expression data [28].

More formally, the gene expression data is typically organized in a two-dimensional matrix format where *the rows correspond to genes and the columns correspond to some biological conditions (or samples)*. The columns usually represent various possible phenotypes such as normal cells, cancerous cells, drug treated cells, or time-series points. Also, the number of genes is significantly larger than the number of conditions. Clustering has been successfully employed as one of the key steps in high-throughput expression data analysis [24]. Several clustering techniques have been successfully applied to cluster the genes, conditions, and/or samples [56].

In this section, we will first describe some of the popular proximity measures used and then categorize the clustering methods proposed in the literature in the context of gene expression data analysis. We will then briefly describe the most representative methods that are widely used for analyzing gene expression datasets. Finally, we will provide a discussion about biologically validating the results of the clustering methods.

16.2.1 Proximity Measures

Before explaining more details on the clustering methods, we will define the proximity measures that are used to quantify the similarity (or distance) between two genes across all the conditions. The most popular measures used in the context of gene expression clustering are the following.

- *Euclidean distance*: Given two genes g_i and g_j, the distance between the two genes can be measured as

$$Euclidean(g_i, g_j) = \sqrt{\sum_{k=1}^{N} (g_{ik} - g_{jk})^2}$$

where N is the total number of samples (columns or features). g_{ik} represents the kth column of vector g_i. One of the problems with Euclidean distance measure is its inability to capture shifting and scaling patterns that commonly occur in gene expression data [2]. To avoid this problem, typically, these gene vector representations are Z-score normalized by subtracting the mean of the gene vector from individual column values and then dividing them by the variance of the original gene vector [20]. This will make the mean value of the resultant vector zero and the variance value one for each gene vector.

- *Pearson's correlation coefficient*: It measures the similarity between the shapes of the expression profiles of two genes as follows:

$$Pearson(g_i, g_j) = \frac{\sum_{k=1}^{N}(g_{ik} - \mu_{g_i})(g_{jk} - \mu_{g_j})}{\sqrt{\sum_{k=1}^{N}(g_{ik} - \mu_{g_i})^2}\sqrt{\sum_{k=1}^{N}(g_{jk} - \mu_{g_j})^2}}$$

where μ_{g_i} and μ_{g_j} represent the mean of the expression values for the genes g_i and g_j, respectively. This measure has been widely used in the analysis of gene expression data but it is not robust to outliers in the data.

- *Spearman correlation coefficient*: To make the similarity measure robust to the underlying distributions and outliers, Spearman correlation coefficient considers the rank ordering of the expression values. Rather than using the original expression values for each gene, the Spearman correlation coefficient uses the rank of each sample value for that particular gene [56]. It is defined to be the Pearson correlation coefficient between the ranked expression values. Since the original expression values are completely discarded, the results from this measure are almost always inferior to those obtained using Pearson's correlation coefficient in the context of standard gene expression clustering.

- *Mutual Information*: Mutual Information (MI) is an information-theoretic approach which uses a generalization of pairwise correlation coefficient to compare two gene expression profiles. *MI* can be used to measure the degree of independence between two genes [19]. The *MI* between a pair of genes g_i and g_j is computed as follows:

$$MI_{ij} = H_i + H_j - H_{ij}$$

where H denotes the entropy which is given as follows:

$$H_i = -\sum_{k=1}^{N} p(g_{ik}) log(p(g_{ik}))$$

It can be seen that the higher the entropy, the more randomly distributed are gene expression levels across the conditions. Also, *MI* becomes zero if the expression levels of genes i and j are statistically independent since their joint entropy $H_{ij} = H_i + H_j$. A higher value of *MI* indicates that the two genes are nonrandomly associated to each other. Even though *MI* has shown some promising results in the context of clustering [83], it is more widely used in the context of constructing gene co-expression networks.

16.2.2 Categorization of Algorithms

The existing clustering methods can be categorized into the following groups as shown below. Each of these categories will be explained more elaborately in this section.

1. **Standard (single-dimensional) clustering**: In this category, a standard clustering technique can be applied on the gene expression data to cluster the genes or to cluster the samples (or conditions). Such clustering methods can be used to capture the relationships between genes and identify the co-expressed genes based on the microarray data collected. It should be noted that these standard clustering techniques can be applied not only on the genes but also on biological conditions.

2. **Biclustering**: This is also referred to as co-clustering [71]. Methods in this category aim to discover local patterns in complex noisy gene expression data by simultaneously clustering both genes (rows) and conditions (columns). These methods are effective in identifying clusters of genes that are correlated only under a subset of conditions. [71].

3. **Triclustering:** The goal of triclustering is to find coherent clusters that are similar across three dimensions ($genes \times conditions \times time$) [123]. In the triclustering approach, the genes are clustered across a subset of conditions under a subset of time points.

4. **Time-Series clustering**: In some microarray studies, the gene expression values are collected over different time points that correspond to various experimental conditions [11]. One of the key characteristics of such time-series data is that it typically exhibits a strong autocorrelation between successive time points. In such scenarios, it is critical to capture the inherent relationships between genes over time in order to accurately perform clustering of the genes [6].

16.2.3 Standard Clustering Algorithms

In this section, we will briefly explain the most widely studied clustering methods for analyzing gene expression data.

16.2.3.1 Hierarchical Clustering

Hierarchical clustering algorithms (discussed in detail in Chapter 4) first create a dendrogram for the genes, where each node represents a gene cluster and is merged/split using a similarity measure. There are two categories of hierarchical clustering that are studied in the context of gene expression analysis.

- **Agglomerative clustering** *(bottom-up approach)*: This approach starts with each gene as an individual cluster, and at each step of the algorithm, the closest pair of clusters are merged until all the of the genes are grouped into one cluster. Eisen et al. [33] applied an agglomerative clustering algorithm called UPGMA (Unweighed Pair Group Method with Arithmetic Mean). Using this approach, each cell of the gene expression matrix is colored and the rows of the matrix are reordered based on the hierarchical dendrogram structure and a consistent node-ordering rule. An illustration of a simple dendrogram for gene expression data is shown in Figure 16.1.

- **Divisive clustering** *(top-down approach)*: This approach starts with a single cluster that contains all the genes; then repeatedly the clusters are split until each cluster contains one gene. Based on a popular deterministic annealing algorithm, authors in [3] proposed a divisive approach to obtain gene clusters. The algorithm first chooses two random initial centroids. An iterative Expectation-Maximization algorithm is then applied to probabilistically assign each gene to one of the clusters. The entire dataset is recursively split until each cluster contains only one gene.

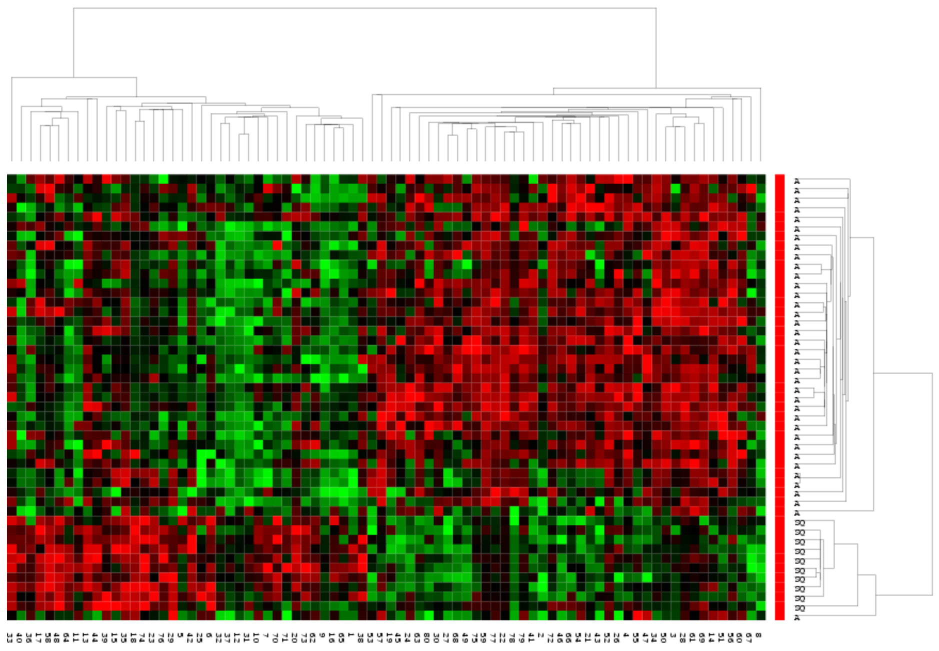


FIGURE 16.1 (See color insert): A simple dendrogram based on hierarchical clustering of rows and columns for gene expression data. The figure has been adapted from [39]. Here, rows correspond to the genes and columns correspond to the conditions. The color scale ranges from saturated green for log ratios -3.0 and below to saturated red for log ratios 3.0 and above.

16.2.3.2 Probabilistic Clustering

Since some of the genes are regulated by several biological pathways, it is important to obtain overlapping clusters; i.e., some genes might appear in multiple clusters. Probabilistic clustering provides an intuitive solution to this problem by implicitly modeling this overlapping nature through assigning probabilistic memberships. The most popular choice of probabilistic clustering of the data is by developing a model-based approach. *Model-based clustering* algorithms assume that the gene expression data is generated by a finite mixture of probability distributions [118]. The primary challenge here is to estimate the best probabilistic model that represents the patterns in the complex data. A popular Model-based CLUSTering algorithm, MCLUST, uses multivariate Gaussian distributions for clustering microarray data [118]. The basic idea here is that each cluster of genes is generated by an underlying Gaussian distribution. More details about clustering using Gaussian mixture models are given in Chapter 3. It should be noted that before applying any model-based approach, the raw gene expression data is first transformed or normalized. Several feature transformation methods have been shown to achieve good results when applying the model-based clustering. The other choice for probabilistic clustering is to apply a Fuzzy C-means (FCM) algorithm to cluster the gene expression data [26]. More details about this algorithm are available in Chapter 4.

16.2.3.3 Graph-Theoretic Clustering

In the graph-theoretical approach, a proximity graph is constructed where the nodes are the genes and the edges are the similarities between the nodes. After constructing the graph, the problem of clustering is transformed into finding minimum cut or maximal cliques in the proximity graph. CLuster Identification via Connectivity Kernels (CLICK) is a graph-theoretical algorithm that defines clusters as highly connected components in the proximity graph [95]. This algorithm does not make any prior assumptions on the number or the structure of the clusters.

Cluster Affinity Search Technique (CAST) is another graph-based clustering algorithm [14]. This algorithm alternates between adding high affinity elements to the current cluster and removing low affinity elements from this cluster. The affinity between gene i and cluster C is defined as the sum of the similarities between gene i and all genes in cluster C. The CAST algorithm then aims to find cluster assignments so that the affinity between genes and their clusters is maximized. It adds a gene to a cluster if the affinity is higher than some prespecified threshold. CAST repeats this operation over all genes and clusters, until all genes are assigned to at least one of the clusters. One of the advantages of this algorithm is that it does not require a predefined number of clusters and can efficiently handle outliers.

Another graph-theoretic algorithm that uses Minimum Spanning Tree (MST) for clustering gene expression data has been proposed in [114]. One of the key properties of using the MST representation is that each cluster of the expression data corresponds to one subtree of the MST, which will then transform a multidimensional clustering problem to a tree partitioning problem. The simple structure of an MST facilitates efficient implementations of rigorous clustering algorithms, and it does not depend on detailed geometric shape of a cluster. The implementation of this algorithm is available in a software package called EXpression data Clustering Analysis and VisualizATiOn Resource (EXCAVATOR).

16.2.3.4 Self-Organizing Maps

Self-Organizing Maps (SOMs) [64] is a clustering algorithm that is based on neural networks with a single layer. The clusters are identified by mapping all data points to the output neurons [103]. SOMs require the number of clusters and the grid layout of the neuron map as the user input. An unsupervised neural network algorithm called the Self-Organizing Tree Algorithm, (SOTA), was studied in [49]. SOTA is a top-to-bottom divisive hierarchical clustering method that is built using SOMs.

16.2.3.5 Other Clustering Methods

k-means clustering has been applied to the problem of clustering gene expression data [50]. In spite of its simplicity and efficiency, it is not well suited to the problem of gene expression clustering due to the following reasons:

- Gene expression data typically contains a lot of noise. The standard k-means algorithm is known to be sensitive to noise due to the objective function (root mean square) it optimizes.
- Since the number of gene clusters is unknown beforehand for gene expression data, there is a need to run this algorithm several times with different inputs. For large datasets, such an approach becomes impractical.

However, other variations of the k-means algorithm have been proposed to overcome some its drawbacks [104].

To improve the accuracy of clustering the tumor samples, resampling methods such as bagging, were proposed in [32]. In these ensemble methods, clustering is applied to bootstrap learning sets, and the resulting multiple partitions are combined. The main intuition of using bagging is to reduce variability in the partitioning results through averaging. Clustering gene expression data using Principal Components Analysis (PCA) was studied in [120]. However, it was shown that clustering with the principle components instead of the original variables often degrades the cluster quality. Therefore, PCA was not recommend before clustering except in specific cases [120].

16.2.4 Biclustering

Standard clustering techniques discussed in the previous section typically assume that closely related genes must have similar expression profiles across all the conditions [69]. However, this assumption does not hold in all of the biological experiments. From a practical point of view, not all the genes are involved in each biological pathway, and some of these pathways may be active under only a subset of the samples [75]. Hence, biclustering was proposed to overcome the limitations of the traditional clustering algorithms [71].

The objective of biclustering is to simultaneously cluster both rows and columns in a given matrix. Biclustering algorithms aim to discover local patterns that cannot be identified by the traditional one-way clustering algorithms. A bicluster can be defined as a subset of genes that are correlated under a subset of biological conditions (or samples). Biclustering has been used in several applications such as clustering microarray data [71], identifying protein interactions [68], and other data mining applications such as collaborative filtering [40] and text mining [18].

The concept of biclustering is illustrated using a simple example in Figure 16.2. In this example, the expression levels of three genes over 10 conditions are shown. Considering all of the ten samples, it is evident that there is no strong correlation between the three genes (Figure 16.2(a)). However, it can seen that there is a strong correlation between the three genes in a subset of the conditions, namely $\{2,3,5,8\}$ (Figure 16.2(b)). Hence, we will consider these three genes to be highly correlated though traditional proximity measures that consider all of the conditions will determine that these three genes are not correlated. Biclustering has emerged as a powerful tool to simultaneously cluster both dimensions of a data matrix by utilizing the relationship between the genes and the samples. It has been proven that the task of finding all the significant biclusters is an NP-hard problem [21].

There are *several challenges* that arise while searching for biclusters in gene expression data. A subset of genes can be correlated only in a small subset of conditions due to the *heterogeneity of the samples*. Such heterogeneity arises due to the complexities involved with different diseases, different patients, different timepoints, or different stages within a disease. In addition, since genes can be *positively or negatively correlated* [54], it is important to allow both types of correlations in the same bicluster. Moreover, there are *several types* of biclusters that can be biologically relevant [71]. A gene can be involved in more than one biological pathway; therefore, there is a need for a biclustering algorithm that *allows overlapping between the biclusters* [27, 75], i.e., the same gene

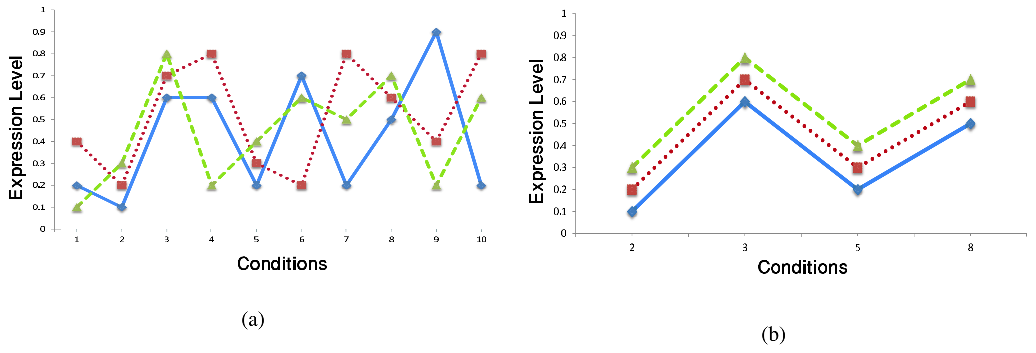


FIGURE 16.2: An illustration of biclustering. The expression levels of three genes over 10 different biological conditions are shown. (a) The genes are uncorrelated when all of the 10 conditions are considered. (b) The genes are strongly correlated in a subset of the conditions $\{2, 3, 5, 8\}$.

can be a member of more than one bicluster. Finally, a bicluster will have to capture the *positively and/or negatively co-expressed set of genes* since the genes in the same biological pathway can be positively and/or negatively correlated [54, 117, 76].

16.2.4.1 Types and Structures of Biclusters

We will now discuss the different types of biclusters that might appear in gene expression data [71]. Let μ be a typical value in the bicluster. α_i is the adjustment for row i and β_j is the adjustment for column j.

- Biclusters with *constant values*. All the elements in this type have the same value. $a_{ij} = \mu$ (Figure 16.3a).
- Biclusters with *constant values on rows*. $a_{ij} = \mu + \alpha_i$ (Figure 16.3b).
- Biclusters with *constant values on columns*. $a_{ij} = \mu + \beta_j$ (Figure 16.3c).
- Biclusters with *(additive) coherent values*. Each row and column is obtained by addition of the previous row and column by a constant value. $a_{ij} = \mu + \alpha_i + \beta_j$ (Figure 16.3d).
- Biclusters with *(multiplicative) coherent values*. Each row and column is obtained by multiplication of the previous row and column by a constant value. $a_{ij} = \mu \times \alpha_i \times \beta_j$ (Figure 16.3e).
- Biclusters with *coherent evolutions*. In this type, the coherence of the values is not considered. Only the direction of change of values is important (Figure 16.3(f)).

2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0

(a) Constant values.

1.0	1.0	1.0	1.0
2.0	2.0	2.0	2.0
3.0	3.0	3.0	3.0
4.0	4.0	4.0	4.0

(b) Constant rows.

1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0

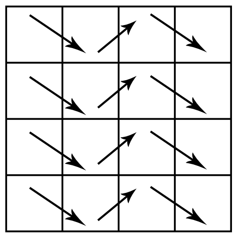
(c) Constant columns.

1.0	4.0	5.0	0.0
4.0	7.0	8.0	3.0
3.0	6.0	7.0	2.0
5.0	8.0	9.0	4.0

(d) Coherent values (additive).

1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0

(e) Coherent values (multiplicative).



(f) Coherent evolutions.

FIGURE 16.3: Examples of different types of biclusters.

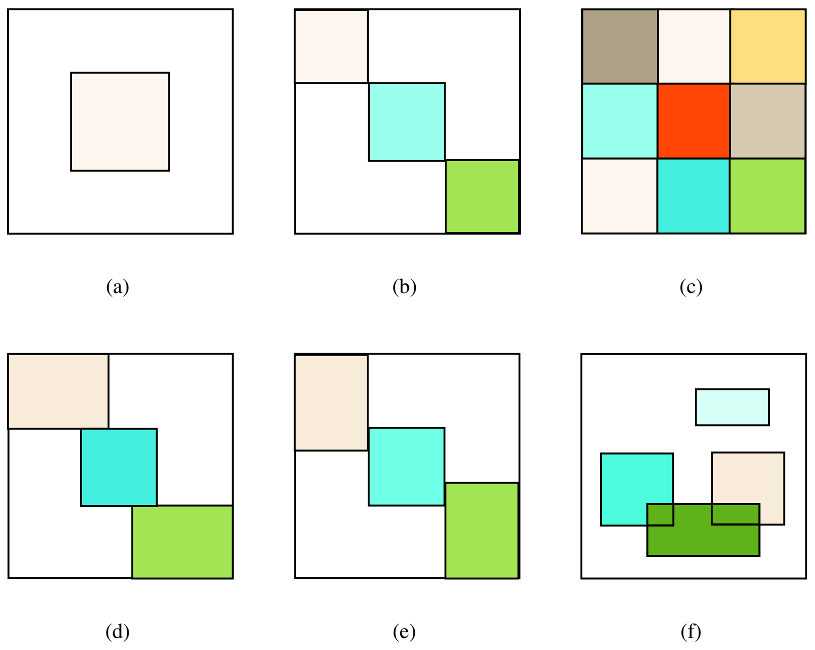


FIGURE 16.4 (See color insert): Examples of bicluster structures. (a) Single bicluster. (b) Exclusive row and column biclusters. (c) Checkerboard pattern biclusters. (d) Exclusive row biclusters. (e) Exclusive column biclusters. (f) Arbitrarily positioned overlapping biclusters.

In addition to the variations in the types of the biclusters, there are other important sources of variations, such as the variations in the size and the position of the biclusters in the gene expression data. Though the earlier biclustering algorithms used to find only a single bicluster at a time (Figure 16.4(a)), most of the recent approaches attempt to find several biclusters simultaneously. When there are several biclusters in the data, some of the standard bicluster structures are as follows.

- Exclusive row and column biclusters which form rectangular diagonal blocks after reordering rows and columns (Figure 16.4(b)).
- Checkerboard pattern biclusters that are completely nonoverlapping (Figure 16.4(c)).
- Exclusive row biclusters that might have overlapping columns (Figure 16.4(d)).
- Exclusive column biclusters that might have overlapping rows (Figure 16.4(e)).
- Arbitrarily positioned overlapping biclusters (Figure 16.4(f)).

16.2.4.2 Biclustering Algorithms

In the first biclustering algorithm proposed by Cheng and Church [21], the mean-squared residue (MSR) score was used as a measurement of the coherence between two genes. Given a gene expression submatrix X that has I genes and J conditions, the residue is computed as follows:

$$H(I,J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (x_{ij} - x_{Ij} - x_{iJ} + x_{IJ})^2 \qquad (16.1)$$

where $x_{iJ} = \frac{\sum_{j \in J} x_{ij}}{|J|}$ is the row mean, $x_{Ij} = \frac{\sum_{i \in I} x_{ij}}{|I|}$ is the column mean and $x_{IJ} = \frac{\sum_{i \in I, j \in J} x_{ij}}{|I| * |J|}$ is the overall mean of the matrix X. x_{ij} is a particular element (ith row and jth column) of the original matrix. A perfect bicluster will have MSR= 0. The MSR function has been used in many biclustering algorithms [21, 115, 27, 75].

This algorithm starts with the original data matrix; then, a set of row/column deletions and additions are applied to produce one bicluster, which will be replaced with random numbers. This procedure is repeated until a certain number of biclusters is obtained. The algorithm has two main limitations: (i) It finds only one bicluster at a time, and (ii) random interference (masking the discovered biclusters with random numbers) reduces the quality of the biclusters and obstructs the discovery of other biclusters. After this algorithm was proposed, a plethora of new heuristic algorithms that aim to extract biclusters from noisy gene expression data have been developed. We will mention only a few here; for a detailed discussion on several existing biclustering algorithms, we refer the readers to an excellent survey on this topic [71].

Coupled two-way clustering (CTWC) technique was proposed in [41]. In this technique, a subset of genes (conditions) are used to cluster the conditions (genes), while the Order-Preserving Submatrices (OPSMs) [13] algorithm finds local patterns in which the expression levels of all genes induce the same linear ordering of the experiments. However, the OPSM algorithm finds only one bicluster at a time and captures only positively correlated genes. Iterative Signature Algorithm (ISA) [51] is a statistical biclustering algorithm which defines a transcription module (bicluster) as a coregulated set of genes under a set of experimental conditions. ISA starts from a set of randomly selected genes (or conditions) that are iteratively refined until they are mutually consistent. At each iteration, a threshold is used to remove noise and to maintain coregulated genes and the associated coregulating conditions.

16.2.4.3 Recent Developments

Recently, there have been many emerging trends in the field of biclustering: (i) Identifying overlapping biclusters and handling both positive and negative correlations within a bicluster has gained some attention due to their biological importance [117]. Some of the recent algorithms [27, 75] allow for overlapping biclusters and find k row clusters and l column clusters simultaneously. (ii) *Differential biclustering* [76, 77, 37] aims to find gene sets that are correlated under a subset of conditions in one class of conditions but not in the other class. Identifying such class-specific biclusters can provide valuable knowledge for understanding the roles of genes in several diseases [76]. The classes could represent different tissue types (normal vs cancerous), different subject types (e.g., male vs female), different group types (African-American vs Caucasian American) [61], different stages of cancer (early stage vs developed stage) [76], or different time points [42]. (iii) *Query-based biclustering* algorithms [30, 122, 4] allow for identifying biclusters that are centered around a set of seed genes of interest. In these algorithms, new search strategies are developed to exploit the expression profiles of certain genes of interest that are used to guide the bicluster searching mechanism. These approaches are extremely handy when one wants to compare the expression profiles of a certain set of genes with that of the existing knowledge which can be obtained by querying for similar profile genes from a large-scale expression compendia.

16.2.5 Triclustering

The goal of triclustering is to find coherent subspace clusters that are similar across three dimensions. The motivation of this task comes from the biological domain where finding coherent clusters along the gene-sample-time (temporal) or gene-sample-region (spatial) dimensions is of great value. Although the problems that are suitable for triclustering had been addressed earlier by ad-hoc methods [55], Zhao and Zaki [123] proposed the first formal algorithm for triclustering. After that, a few more algorithms have also been proposed in recent years [98, 57].

Given G, the set of genes; S, the set of samples; and T, the set of time points; a tricluster C is a submatrix of the dataset $D = G \times S \times T$, where $C = X \times Y \times Z = \{c_{ijk}\}$, with $X \subseteq G, Y \subseteq S$, and $Z \subseteq T$ provided that certain conditions of homogeneity are satisfied. For example, a simple condition might be that all values $\{c_{ijk}\}$ are identical or approximately equal. If we are interested in finding common gene co-expression patterns across different samples and times, we can find clusters that have similar values in the G dimension, but possibly different values in the S and T dimensions. Other homogeneity conditions can also be defined, such as similar values in S dimension and order preserving submatrix. [71]. Let $\mathcal{B}$ be the set of all triclusters that satisfy the given homogeneity conditions, then $C \in \mathcal{B}$ is called a maximal tricluster iff there does not exist another cluster $C' \in \mathcal{B}$ such that $C \subset C'$. In most of the cases, we are interested in only the maximal triclusters.

Zhao and Zaki's method for triclustering is known as TRICLUSTER. It accepts the 3-dimensional dataset D; the minimum size thresholds, mx, my, and mz, that define the size of the clusters in three dimensions; and a maximum ratio threshold, ε, which represents the maximum allowed deviation among values in different cells of a cluster. It then constructs a range multigraph data structure for the data matrix at each of the timestamps. It uses the range multigraphs of a timestamp to obtain a set of robust biclusters (involving dimensions G and S) that are observed for that time value. As a final step, it merges similar biclusters across different timestamps to obtain maximal triclusters. For this step, it represents each of the biclusters that it has found in the earlier step as a node in a graph, and defines the node–node (bicluster–bicluster) relationships based of the similarity on the time dimension; then the maximal triclusters are simply the maximal cliques in this graph. As an optional step, it also merges and deletes clusters based on the degree of overlap among various dimensions. Interested readers can read more details on the TRICLUSTER algorithm from the original paper by the authors [123]. Key features of TRICLUSTER are that it is flexible and can accept various homogeneity criteria. Also, it is robust and generates only maximal triclusters. The downside of this method is that it requires a large number of parameters and domain knowledge is necessary to set the parameter values optimally.

16.2.6 Time-Series Gene Expression Data Clustering

In order to determine the complete set of genes that are expressed under a set of new conditions and to determine the interaction between these genes in these new conditions, it is important to measure a time course of expression experiments [11]. In such microarray studies, the gene expression values are collected over different time points which correspond to the experimental conditions. One of the key characteristics of such time-series data is that while static data from a sample population are assumed to be i.i.d. (independent and identically distributed), time series gene expression data typically exhibit a strong autocorrelation between successive timepoint values. In such scenarios, it is critical to capture the inherent relationships between genes over time in order to perform clustering [6]. One of the first models to cluster time series gene expression data was developed in [70]. This clustering algorithm was based on the mixed effects model using B-splines and was applied on the yeast cell cycle gene expression data. The estimated gene expression trajectory was also used to fill in the missing gene expression levels for any time point using the available data in the same cluster.

Hidden Markov Models (HMMs) have also been used to cluster time-series gene expression data [91]. The primary advantage of using HMMs is that they explicitly take into account the temporal nature of the expression patterns which will produce high quality clusters. Given gene expression data, the goal is to find a partition of the data into K HMMs which will maximize the likelihood of the data given the learned HMM model. For gene expression data, the emission probabilities are assumed to be Gaussians with fixed variance. The authors of [91] developed a new algorithm for clustering genes based on a mixture of HMMs. The parameters of this model are learned using an iterative Expectation-Maximization style algorithm. The two iterative steps in this algorithm are (i) genes are associated with the HMM that would have most likely generated their time courses and (ii) the parameters of each HMM are estimated using the genes assigned to it. This algorithm requires the number of time points to be much larger than the number of states. Though this algorithm is suitable for clustering long time-series data, it does not work well on short time-series data.

To tackle the challenges with short time-series data, the authors of [36] proposed an algorithm specifically designed for clustering short time-series expression data. The algorithm works by assigning genes to a predefined set of model profiles that capture the potential distinct patterns. After determining the significance of each of these profiles, the most significant ones are retained for further analysis and can be combined to form clusters. Using immune response data, the authors have shown that their algorithm can correctly detect the temporal profile of relevant functional categories. Using Gene Ontology-based evaluation, the algorithm outperformed both general clustering algorithms and algorithms designed specifically for clustering time-series gene expression data. STEM is a toolkit that is available based on this work for the analysis and clustering of short time series gene expression data [35]

In certain scenarios, a gene might not instantaneously be correlated with other genes at that time but a gene might regulate another gene after a certain time. In order to identify time-lagged coregulated gene clusters, [53] proposes a novel clustering algorithm for effectively extracting the time-lagged clusters. This algorithm first generates complete time-lagged information for gene clusters by processing several genes simultaneously. Instead of considering the lags for the entire sequence, it considers only small interesting parts (subsequences) of the genes that are coregulated while there is no distinct relationship between the remaining part. It identifies localized time-lagged co-regulations between genes and/or gene clusters. It builds a novel mechanism that aims to extract clusters (which are referred to as *q*-clusters) of (time-lagged) coregulated genes over a subset of consecutive conditions. Each such cluster essentially contains information of genes that have similar expression patterns over a set of consecutive conditions. More recently, authors in [113] have extended the concept of time-lagged clustering to three-dimensional clustering.

16.2.7 Cluster Validation

All the clustering algorithms that are described in the previous section will yield either groups of co-expressed genes or groups of samples with a common phenotype [22]. Reliability of the clusters, which measures the probability that the clusters are not formed by chance, is a commonly used metric for validating the results of these clustering algorithms. To compute the p-values of a cluster, typically the genes from a given cluster are mapped to the functional categories defined in annotated databases such as Martinsried Institute of Protein Sciences (MIPS) or Gene Ontology (GO) [87]. Typically, a hypergeometric distribution is used to calculate the probability of having at least k genes from a cluster of size n genes by chance in a biological process containing f genes from a total size of N genes as follows:

$$P = 1 - \sum_{i=0}^{k} \frac{\binom{f}{i}\binom{N-f}{n-i}}{\binom{N}{n}}$$

This test measures if a gene cluster is enriched with genes from a particular functional category to a greater extent than what would be expected by chance. The range of the p-values is from 0 to 1. Lower p-values indicate biological significance of the clusters.

Another popular metric for evaluating the goodness of the clusters is the Figure of Merit (FOM) which was originally proposed in [119] to estimate the predictive power of clustering algorithms. The FOM measure computes the mean deviation of the expression levels of genes in a particular condition relative to their corresponding cluster means. Thus, a small value of FOM indicates high predictive ability of the resulting clusters.

16.3 Clustering Biological Networks

Proteins control the functions of the cell [52]. Understanding these functions requires not only studying the proteins and but also studying their interactions [8]. Protein interactions are essential elements of all the biological processes [16]. Pairwise protein interactions have been identified and validated using recent technologies. These interactions have been obtained by different methods such as mass spectrometry, two-hybrid methods, and genetic studies. The whole network of protein–protein interactions for a given organism describes the interactome of that organism. The protein–protein interaction (PPI) network is represented as a graph in which the nodes represent the protein and the edges represent the interactions between the corresponding proteins [84].

It has been shown that proteins of known functions tend to cluster together [92]. The network distance is correlated with functional distance, and the proteins that are closer to one another tend to have similar biological function [96]. Therefore, studying the PPI networks is crucial in discovering the functions of proteins and thus understanding inner workings of cells [84]. Clustering the PPI network can be used to predict the unknown functional categories of proteins.

16.3.1 Characteristics of PPI Network Data

We will first describe some of the key characteristics of PPI networks [8]:

1. *Scale-free structures* [52]: PPI networks contain hub proteins which are typically involved with many interactions. In other words, most of the proteins in the network have few interactions, and only a few proteins will have a lot of interactions with other proteins. Applying existing clustering techniques on these networks would produce a few giant clusters (containing the hub nodes) and the remaining clusters would be very small. Hence, the clustering process should be adapted to produce better results in terms of the size of the clusters.

2. *Disassortativity:* In many forms of scale-free networks (such as social networks), highly connected nodes usually are strongly connected with each other. This property is known as *assortativity*. However, in protein interaction networks this is not the case. Hubs are not directly linked to each other thus causing the disassortativity in such networks though the average path lengths are relatively shorter compared to other networks.

3. *Multifunctionality:* The same protein can be involved in several biological processes [105]. Due to these overlapping structures, it becomes difficult to extract groups of proteins in such a way that a single protein is present in multiple groups.

16.3.2 Network Clustering Algorithms

Several network clustering algorithms have been proposed in the PPI literature. Some of the most popular ones will be discussed in this section.

16.3.2.1 Molecular Complex Detection

Molecular Complex Detection (MCODE) algorithm was one of the first computational methods that was proposed to detect protein complexes based on the protein connectivity values in the PPI networks. The MCODE algorithm aims to detect the densely connected nodes in complex networks [9]. The algorithm first assigns a weight to each node based on each protein's local network density using the highest k-core of the node neighborhood. A k-core is a graph of minimal degree k. As an alternative to the clustering coefficient, the MCODE algorithm defines the core-clustering coefficient of a node u as the density of the highest k-core of the immediate neighborhood of u. The final weight for any node is the product of the node core-clustering coefficient and the highest k-core level.

In the next step, the MCODE algorithm selects the highest weighted node and recursively moves outward from this seed node including the nodes whose weights are above a certain threshold. If a node is included, its neighbors are recursively checked in the same manner to see if they are part of the complex until no more nodes can be added to the complex, and this process is repeated for the next highest unseen weighted node. The experimental results from this algorithm indicate that the protein complexes obtained are generally small in number and each of the results is a much larger complex.

16.3.2.2 Markov Clustering

The Markov clustering (MCL) algorithm is one of the widely studied graph clustering algorithms [106]. Markov clustering has been applied to PPI networks such as yeast and human PPI networks, and the generated clusters accurately map to known protein complexes [65, 85]. MCL is an iterative algorithm that has two main alternating steps: expansion and inflation. Initially, the graph is translated into a stochastic Markov matrix so that the sum of the values in each of the columns is 1. This matrix represents the transition probabilities between all pairs of nodes. The resulting stochastic matrix is then clustered using the following steps [38, 106]. *(i) Expansion:* The expansion step spreads the flow out of a node to potentially new nodes and enhances flow in dense regions. This step leads to strengthening the strong edges and further weakening the weaker edges. *(ii) Inflation:* The inflation step aims to strengthen the intracluster flow and weaken the intercluster flow to obtain the natural clusters. The above two steps are repeated until convergence. The algorithm is said to be converged when we obtain a doubly idempotent matrix (a nonnegative column-homogenous matrix idempotent under matrix multiplication). Also, it has been shown that MCL is more robust to noise and identifies meaningful clusters [107].

16.3.2.3 Neighborhood Search Methods

The Restricted Neighborhood Search Clustering (RNSC) algorithm was proposed in [63] to cluster PPI data of Saccharomyces cerevisiae, Drosophila melanogaster, and Caenorhabditis elegans and predict protein complexes. This algorithm optimizes a cost-based local search algorithm based loosely on the tabu search metaheuristic [43]. Clustering of a graph begins with an initial random clustering and a cost function. Nodes are then randomly added or removed from clusters to find a partition that minimizes the cost function value. This cost function is based on the number of invalid connections (absence of intracluster connections or presence of intercluster connections) incident from a particular node. To achieve high accuracy in predicting the protein complexes, some post-processing is performed based on the functional homogeneity of the complex, density thresholding, and minimum size thresholding.

16.3.2.4 Clique Percolation Method

In order to extract the overlapping structures from complex networks, Palla et al. [79] proposed the Clique Percolation Method (CPM). CPM has the ability to generate the overlapping clustering by identifying the k-clique percolation communities. A k-clique corresponds to a complete subgraph of size k. A cluster is defined to be the maximal union of k-cliques that can be reached from others through a series of adjacent k-cliques. A software package named CFinder [1] implements this method and is currently being used even in other application domains such as social networks [78].

16.3.2.5 Ensemble Clustering

An ensemble clustering approach applies different clustering methods on the PPI network data and then combines the clustering results of all these methods (the base clustering algorithms) into a single comprehensive clustering result of the PPI networks [8]. The following three base clustering algorithms were used in this work.

(i) Repeated bisections [100]: This method starts with having the complete dataset as one cluster. Then the following steps is repeated until the desired number of clusters is obtained: (1) select a cluster to bisect into two clusters and (2) compute the similarity score for each cluster. This algorithm optimizes the $I2$ criterion defined as follows:

$$I2 = max \sum_{i=1}^{k} \sqrt{\sum_{v,u \in C_i} S(u,v)}$$

where k is the number of clusters, C_i is the set of objects in cluster i and $S(u,v)$ is the similarity between the two objects u and v.

(ii) Direct k-way partitioning [59]: This method works as follows: (1) select a set of k objects (seeds), and (2) compute the similarity between each object and the seed, and (3) assign each object to the most similar cluster. This procedure is repeated to optimize the $I2$ criterion.

(iii) Multilevel k-way partitioning [59]: This algorithm has three main steps: coarsening, initial partitioning, and refinement. In the coarsening step, k-way partitioning is used to cluster the graph into a set of smaller graphs. In the second step, the partitions are projected back onto the original graph by iterating over intermediate partitions. Finally, the refinement step reduces the edge-cut while conserving the balance constraints.

The clustering results of the above mentioned base clustering methods are combined by applying the following three phases:

1. *Cluster purification:* The similarity between the objects within each cluster is computed, and the weak clusters are removed such that each protein is a member of at least one third of the remaining clusters. The result of this step is represented using a binary cluster membership matrix where each column is a cluster produced by the base clustering algorithms and each row is a protein.

2. *Dimensionality reduction:* A dimensionality reduction method, such as PCA, can be used to reduce the number of dimensions in the cluster membership matrix. This will avoid the problem of the curse of dimensionality.

3. *Consensus clustering*: Two different consensus clustering algorithms are applied: (i) *the recursive bisection* algorithm where the best of the three base clustering algorithms is chosen and (ii) *the agglomerative hierarchical clustering* where the desired k-way clustering solution is computed using the agglomerative method.

Weighted consensus: This method considers the weight of the edges between proteins within each cluster [8]. In **soft consensus** clustering, the same protein can belong to more than one cluster. The cluster membership can be computed as a factor of the distance from the nodes in the cluster. The soft clustering solves the problem of multifunctional proteins. This ensemble method produced better results on yeast PPI networks compared to several other methods.

16.3.2.6 Other Clustering Methods

In [48], a clustering algorithm was developed that combines information from expression data and biological networks and computes a joint clustering of genes and nodes of the biological network. This method was validated using expression data of the yeast, and the results were explained in terms of the biochemical network and the gene expression data. In [82], a biclustering-based approach was proposed to cluster the PPI data and generate both overlapping and nonoverlapping clusters. This algorithm was applied on human and yeast networks.

16.3.3 Cluster Validation and Challenges

Using different clustering methods, various results are obtained from a given PPI network. Hence, it is important to compare and evaluate the performance of these clustering algorithms. Being an unsupervised approach, it is often quite difficult to evaluate and compare the performance of various clustering algorithms used in the context of PPI network analysis. Validating the clusters is primarily done by calculating the p-values for the clusters [110]. Statistically significant p-values indicate that the set of proteins in the same clusters are involved in the same biological functions. The GO database provides the annotation of known molecular functions and biological processes [7]. Similar to the evaluation of the gene clusters from the expression data, a hypergeometric distribution is used to calculate the probability of having at least k genes from a cluster of size n genes by chance in a biological process containing f genes from a total size of N genes. Lower p-values indicate biological significance of the clusters.

If there are known protein complexes, they can be used as a gold-standard to evaluate the performance of the clustering algorithm by comparing the predicted cluster to the known ones. Clustering the PPI data is still a challenging problem due to various reasons [110]. The false positive and false negative interactions in the protein network makes it difficult to evaluate the quality of the resulting clusters. Also, there is a need for clustering methods that allow overlapping between the clusters. Similar to other applications, identifying the optimal number of clusters in the PPI data is a challenging task. In addition, the large amount of available data makes it computationally expensive to analyze the PPI networks.

16.4 Biological Sequence Clustering

Sequence clustering is an essential task in biological data analysis, because the most important building blocks of living organisms, such as DNA, RNA, mRNA, polypeptides, and proteins, have a linear structure and can be represented as sequences. For DNA, RNA, and mRNA, the sequences are made of nucleic acids, also called bases and for polypeptides and proteins, the sequences are made of amino acids. Earlier studies on biological sequences were mostly limited to pairwise sequence alignment and multiple sequence alignment. However, in recent years, scientists are amassing an enormous amount of sequence data as a result of improvement on the high-throughput sequencing. Similarity search in such a large sequence database using alignment algorithms is extremely costly. So, alignment-based similarity search is performed on a small cluster of highly similar sequences. Sequence clustering plays a significant role in finding those small clusters.

16.4.1 Sequence Similarity Metrics

Many of the existing clustering methods that we have discussed in the earlier sections, such as bi-clustering, graph-theoretic clustering, and Markov clustering can be used for clustering sequences if a suitable distance (or similarity) metric is available. Formally speaking, a distance metric takes a pair of sequences and returns a real number which denotes the distance between the given sequences. Once all pairwise distances among a set of sequences are found, the similarity information can be encoded in a matrix or in a graph. In the case of a matrix, the rows and the columns correspond to the sequences, and for the graph, the nodes represent the sequences and an edge represents a pair of *similar* sequences for a chosen similarity threshold. Any traditional clustering algorithm can find clusters once a similarity matrix or a similarity graph is available.

For clustering biological sequences, finding a suitable distance metric is challenging due to the complexity associated with these sequences. For instance, a protein sequence may be composed of various functional domains, and two protein sequences may share only a few of those domains; in that case the overall similarity between these two proteins will be weak. However, if the matched functional domains are highly significant, these proteins should belong to the same cluster. Also, when clustering genome sequences, a similarity metric should consider only the coding part of the DNA and discard a significant part of the genome sequences, such as junk DNA and tandem repeats. Below, we discuss a collection of distance metrics that can be used for measuring the distance between a pair of sequences. It is important to note that though we use the term *metric*, many of the similarities measurement may not be a "metric" using its mathematical definition.

16.4.1.1 Alignment-Based Similarity

The most popular distance metric for sequence data is the Levenshtein distance, or edit distance [46]. It denotes the number of edits needed to transform one string into the other with the allowable edit operations being insertion, deletion, or substitution of a single character. For a pair of sequences, this distance can be computed by performing the global alignment between the sequences; for two sequences of length l_1 and l_2, the global alignment cost is $O(l_1 l_2)$. This is costly considering that biological sequences (particularly, DNA sequences) are typically long. So, Levenshtein distance is not an ideal metric for biological sequences for the task of sequence clustering. Another limitation of this distance metric is that it captures the optimal global alignment between two sequences, whereas for clustering biological sequences, local similarities between two sequences should be considered. Finally, the dependency of the edit distance metric on the length of the sequence makes it a poor metric for the cases where the length of the sequences in the dataset varies significantly.

To capture the local similarity between two sequences, Smith-Waterman's local alignment score [46] can be used. Instead of aligning the entire sequence the local alignment algorithm aligns similar regions between two sequences. The algorithm compares segments of all possible lengths and optimizes the similarity measure. Though it overcomes some of the problems of global alignment, it is as costly as the global alignment algorithm and is simply impractical for clustering thousands of biological sequences.

During the eighties and the nineties, two popular tools, known as FASTA [80] and BLAST (Basic Local Alignment Search Tool) [5], were developed to improve the scalability of similarity search from biological sequence databases. Both tools accept a query sequence and return a set of statistically significant similar sequences from a sequence database. The main benefit of these tools over an alignment-based method is that they are highly scalable, as they adopt smart heuristics for finding similar segments after sacrificing the strict optimality. Also, specifically BLAST has various versions (such as PSI-BLAST, PHI-BLAST, BLAST-tn) that are customized based on the kind of sequences and the kind of scoring matrix used. For a given set of sequences, FASTA and BLAST can also be used to find a set of similar sequences that are pairwise similar. For a pair of similar sequences, BLAST also returns bit score (also known as BLAST-score) which represents the similarity strength that can be used for clustering sequences.

16.4.1.2 Keyword-Based Similarity

To model the effect of local alignment explicitly, some sequence similarity metrics consider the q-gram-based method, where a sequence is simply considered as a bag of short segments of fixed length (say, q); thus, a sequence can be represented as a vector, in which each component corresponds to the frequency of one of the q-length segments. Then the similarity between two sequences is measured using any metric that measures the similarity between two vectors, such as dot product, Euclidean distance, Jaccard coefficient, or even *tf-idf* [89]. This approach is also known as a keyword-based method, as one can consider each sequence as a document and each q-gram as a keyword in the document. The biggest advantage of similarity computation using a keyword-based method is that it is fast. Another advantage is that this method represents a sequence using an $\mathbb{R}^n$ vector, which can accommodate some of the clustering algorithms (such as, k-means) that work only on vector-based data.

16.4.1.3 Kernel-Based Similarity

In recent years, kernel-based sequence similarity metrics also got popular. In [67, 66], the authors present several families of k-gram-based string kernels, such as restricted gappy kernels, substitution kernels, and wildcard kernels, all of which are based on feature spaces indexed by k-length subsequences (k-mers) from the string alphabet. Typically, kernels are used for supervised classification with support vector machines (SVM), however, they can also be used as a similarity metric for unsupervised clustering.

16.4.1.4 Model-Based Similarity

Probabilistic models, such as HMM are also used for finding similarity metrics. For a given set of sequences to be clustered, such a method trains one HMM for each of the sequences. Then, the similarity between two sequences can be obtained from the similarity (or distance) between the corresponding HMMs. In the past, few authors have proposed approaches for computing the distance between two HMMs [86]; early approaches were based on the Euclidean distance of the discrete observation probability, others on entropy, or on co-emission probability of two HMM models, or on the Bayes probability of error [10].

16.4.2 Sequence Clustering Algorithms

In an earlier section, we discussed that given a similarity metric, we can use any clustering method for clustering biological sequences. Nevertheless, there have been many clustering methods that are explicitly proposed for clustering biological sequences. We discuss them under the following groups.

16.4.2.1 Subsequence-Based Clustering

A subsequence-based clustering method mines a set of frequent subsequences from each of the sequences and uses them as features for clustering the sequences. The idea of such clustering is similar to the task of document clustering using the "bag-of-words" representation of a document. A traditional sequence mining method returns a large number of subsequences that are frequent, but all such subsequences are not good features for clustering. So, a good clustering method needs to choose a subset of these subsequences as features so that when projected on the feature-set the similarity between a pair of sequences is computed correctly. Different algorithms of sequence clustering vary in the way they choose the subsequence feature set. The advantages of subsequence-based clustering methods is that they are typically fast compared to other sequence clustering methods. However, this approach ignores the relative position of various subsequences, so they cannot model some of the sequential relations, such as ordering and sequential dependency.

One of the first among subsequence-based sequence clustering methods was proposed by Guralnik and Karypis [45]. They used traditional frequent sequence mining methods [121] to mine subsequences, but to control the number of subsequences, they imposed minimum and maximum length constraint on the mined subsequences; then they used a subset of the mined subsequences as the feature set for clustering. To select the feature set, they followed two approaches: global and local. The global approach prunes the feature space by selecting a set of independent subsequences, where dependency is defined as the overlap between the symbols of the sequences or as the overlap between their support list. On the other hand, the local approach finds independent features locally from each of the sequences, where two features are independent if they are supported by a nonoverlapping segment of the corresponding sequences. Once the feature set is defined, they used a *k*-means algorithm to cluster the sequences.

A more sophisticated variation of the subsequence-based method was proposed in [111]. In this paper, the authors introduce the notion of frequent summarization subsequence (FSS) and represent each sequence by a collection of those FSSs. Intuitively, an FSS is a keyword that can be viewed as a discriminating feature for clustering the input data sequences. However in this work, the authors use a *tf-idf* kind of weighting on each symbol to assign weight on each of these FSSs. They also present an effective method that directly mines all the FSSs from the sequence data. The final clustering method has two stages. The first stage generates microclusters, which are obtained by simply grouping the sequences with a shared FSS in a cluster. Typically, the number of microclusters is larger than the desired number of clusters, so a second stage is used to merge the microclusters using a hierarchical agglomerative clustering method, until the desired number of clusters is obtained.

16.4.2.2 Graph-Based Clustering

A graph-based sequence clustering method represents the sequences in a similarity graph, in which a vertex represents a sequence and an edge represents the similarity relation between the corresponding pair of sequences. In such a representation, a partition of the similarity graph represents a clustering of the input sequences. The crucial requirement in a graph-based sequence clustering method is to obtain the similarity graph in an efficient manner. A brute-force approach to obtain a similarity graph computes the similarity values between all $\binom{n}{2}$ pairs of sequences (here, n is the number of sequences) and then uses a user-defined threshold to add edges between sequences in a similarity graph. Clearly this is inefficient, as it requires to compute $O(n^2)$ similarity scores explicitly, so many graph-based sequence clustering algorithms use an efficient method for similarity graph construction. Once a similarity graph is obtained, one of the many available graph-clustering [90] methods can be used to obtain the desired clustering. In Algorithm 35, we present a pseudocode for a graph-based sequence clustering algorithm; based on the specific method for the similarity routine (Line 4) and the graph clustering (Line 9), various graph-based sequence clustering methods can be obtained.

The graph-based sequence clustering method that is shown in Algorithm 35 is sometimes not scalable as the computation cost grows quadratically with the number of sequences. Since the similarity computation of each pair of sequences is independent, a straightforward remedy to the lack of scalability is to use distributed or parallel computing to perform the tasks in Lines 2–8. There also exist algorithmic solutions; instead of finding the similarity between all the sequence-pairs explicitly, these solutions adopt methods that find similar segment-pairs across all the input sequences simultaneously. Then, they obtain the similarity between input sequences by scanning the frequency of highly significant similar segment-pairs. For instance, if two sequences share a large number of similar segment-pairs, the pair obtains a high similarity score, and hence the method adds an edge between those two sequences in the similarity graph. The advantage of such methods is that they avoid the all-pair similarity computation with has quadratic complexity and replace it with a method that has linear or sublinear complexity.

Line 9 of Algorithm 35 calls a graph-clustering algorithm. Many of the existing graph-clustering methods, such as spectral clustering [97, 29] or Markov clustering [31], have quadratic ($O(|V|^2)$) complexity and, hence, are not efficient for this task. Also, unlike the earlier task of similarity graph

Algorithm 35 Generic Graph-Based Sequence Clustering

```
Require: Sequence database ($\mathcal{S}$)
         Similarity threshold ($\sigma$)
         Cluster count ($k$)
 1: $G(V,E) = build\text{-}graph(V = \mathcal{S}, E = \emptyset)$
 2: for each edge $s_1 \in \mathcal{S}$ do
 3:     for each edge $s_2 \in \mathcal{S}$ do
 4:         if $sim(s_1,s_2) \geq \sigma$ and $s_1 \neq s_2$ then
 5:             $E = E \cup (s_1,s_2)$
 6:         end if
 7:     end for
 8: end for
 9: $\mathcal{C} = graph\text{-}clustering(G,k)$
10: return $\mathcal{C}$
```

construction, obtaining a parallel or distributed method for clustering graph is nontrivial. So, the majority of the methods choose a simple graph clustering algorithm. One such method is single-link clustering (SLC), which is an agglomerative clustering method that merges two of the existing clusters based on the largest similarity between a pair of sequences that are taken from those two clusters. The process continues until the desired number of clusters is obtained. The complexity of SLC is $O(|E|)$, which is much cheaper than $O(|V|^2)$ for sparse graphs. Since similarity graphs are very sparse, SLC method on such graph is highly efficient. However, the clustering quality of SLC can be poor; for instance, it can happen that the distance of an object from another object belonging to a different cluster can be smaller than the distance of the first object to another object belonging to the same cluster.

One of the earliest sequence clustering methods that uses a graph-based technique is GeneRAGE [34]. It performs an all-against-all sequence similarity search using BLAST; if the similarity between two proteins is higher than a given threshold, an edge is added between those two proteins in the similarity graph. Along this process, GeneRAGE also performs some preprocessing on the similarity graphs. For example, it identifies whether a protein is multidomain by considering the transitivity of similarity among other proteins that are similar to the said protein; if multidomain proteins are found, they are allowed to be part of multiple clusters. For the clustering task, GeneRAGE uses SLC. Another earlier clustering method, called d2_cluster [17] also uses SLC for clustering EST (expressed sequence tags) and full-length cDNA sequences. In a recent work [73], Miele et. al. propose a memory-efficient implementation of SLC by following the well-known Union-Rank data structure for disjoint sets.

Kawaji et al. [60] mention the limitations of single-link clustering and propose to use recursive graph partitioning to find clusters from the similarity graph. However, the balanced bipartition technique that they propose uses a one-change optimization scheme, which is very costly. In [81], the authors use spectral clustering with the multiway normalized cut criteria for clustering the similarity graph; however, as we mentioned earlier, spectral clustering-based methods are not scalable because they require finding eigenvectors and eigenvalues of similarity matrix—a computationally intensive task. Another clustering method called BAG [62] also uses partitioning of the similarity graph, but its partitioning method is targeted to find biconnected components of the similarity graph. For building the similarity graph, BAG uses the FASTA algorithm in an efficient manner; for every sequence $i : 1 \leq i \leq n$, it calls FASTA(s_i, S) and finds the pair of similar sequences with only $O(n)$ number of FASTA calls, instead of $O(n^2)$ such calls. Very recently, Voevodski et al. [108] proposed a method that finds the similarity graph with only $k(< n)$ number of BLAST calls, where the value of k is decided by following an active learning paradigm.

16.4.2.3 Probabilistic Models

The probabilistic approach is popular for sequence modeling. For instance, the earliest approaches for modeling protein families use profile-HMM, a hidden Markov model-based probabilistic approach. However, the main objective of profile-HMM is to perform multiple sequence alignment. Since then, several methods have been proposed to use HMM for sequence clustering. We discuss a few of them below.

Smyth's work [99] is one of the first that uses HMM for clustering biological sequences. His method has two steps: the first step devises a pairwise distance between observed sequences by computing a symmetrized similarity. This similarity is obtained by training an HMM for each sequence, so that the log-likelihood (LL) of each model, given each sequence, can be computed. This information is used to build an LL matrix which is then used to cluster the sequences into k groups, using a hierarchical algorithm. In the second step, one HMM is trained for each cluster; the resulting k models are then merged into a composite global HMM, where each HMM is used to design a disjoint part of this composite model. This initial estimate is then refined using the standard Baum-Welch procedure. As a result, a global HMM modeling of all the sequences is obtained. The number of clusters (k) is selected using a cross-validation method.

CLUSEQ [116] uses conditional probability distribution (CPD) for characterizing a cluster, i.e., for different clusters the CPD of the next symbol given a preceding segment is different, and hence the CPD can be used for clustering biological sequences. To store and retrieve the CPD of various segments, CLUSEQ uses a novel data structure, called a probabilistic suffix tree (PST); every node in PST stores a probability vector to store the probability distribution of the next symbol given the label of the node as the preceding segment. Usages of PST make CLUSEQ very efficient. In the following paragraphs, we describe CLUSEQ in more detail.

The CLUSEQ algorithm accepts a sequence database along with user-defined values for various threshold parameters. At the beginning, all sequences in the database are unclustered. Then, an iterative process is employed to continuously improve the quality of the clustering until no further improvement can be made. In each iteration, CLUSEQ starts with a set of new clusters; a few random sequences from the unclustered set are chosen to be clusters themselves. Then, it examines each sequence to evaluate its similarity to each of the clusters and updates the cluster membership of that sequence, if necessary. At the end of the iteration, CLUSEQ also merges multiple clusters, if their membership overlaps significantly. The key part of CLUSEQ is the step which computes the similarity of a sequence to a cluster. For this, CLUSEQ uses CPD of symbols given a segment. For efficient computation of CPD, the sequences in a cluster are stored in a probabilistic suffix tree. If S is a cluster and $\sigma = s_1 s_2 \cdots s_l$ is a sequence, its likelihood to be a member of the cluster S is given by $P_S(\sigma) = P_S(s_1) \times P_S(s_2|s_1) \times \ldots \times P_S(S_l|S_1 \ldots S_{l-1})$, where $P_S(s_i|s_1 \cdots s_{i-1})$ is the conditional probability that the symbol s_i is the next symbol right after the segment $s_1 \ldots s_{i-1}$ in the sequence cluster S. For each cluster, CLUSEQ maintains a PST, which stores the above conditional probabilities effectively. The strengths of CLUSEQ are that is is specifically adapted for biological sequence clustering and it is very efficient. The weakness is that is has several user-defined parameters that need to be selected appropriately for good clustering results. Also, due to the randomness, the clustering is nondeterministic, and its quality depends on the random choice of the initial sequences that constitute the cluster seeds.

In a recent work [112], the authors propose another CPD-based model, called DHCS, which overcomes some of the difficulties of the CLUSEQ; specifically, DHCS does not choose seed sequences randomly, rather it employs a two-tier Markov Model, where the first tier provides a good initialization for the CPD model in the second tier. The two-tier structure guarantees that the statistical models in the DHCS algorithm are constructed in a statistically significant way without resorting to pairwise comparison of sequences.

16.4.2.4 Suffix Tree and Suffix Array-Based Method

For fast processing of sequence similarity, efficient data structures that index small sequence-segments are also proposed. The main objective of using a data structure is to avoid the full pairwise similarity computation, which is infeasible for many clustering tasks because of its quadratic complexity. For example, databases of ESTs sometimes contain millions of ESTs and efficient data structure is critical for clustering those sequences.

In [72], the authors use suffix arrays to cluster ESTs; suffix arrays are particularly suitable because ETSs are short subsequences of cDNA sequences. To identify all matching blocks of length k, this method first adds all suffixes into a suffix array. Then it sorts the suffixes and groups the suffixes that share a k-length prefix. Then, for each pair of suffixes sharing at least one matching block, it finds the largest consistent matching blocks and the corresponding matching score. Finally, starting with the highest scoring sequence pair, it builds hierarchical clusters using single-link clustering. In [47], the authors propose a clustering tool called CLAGen, that uses a suffix tree for sequence clustering. CLAGen constructs a suffix tree from the input sequences; this construction is highly efficient with only linear time complexity. Then, CLAGen uses the suffix tree for searching and overlapping common subsequences so that it can obtain sequence pairs that are highly similar. From the pairwise similarity information, CLAGen finds sequence clusters. A nice feature of CLAGen is that it annotates the gene clusters by using information from the BLAST search. The CLUSEQ algorithm that we discussed earlier also uses a suffix tree, but it uses it to find the conditional probability of a symbol given a sequence in a cluster.

16.5 Software Packages

Implementations of the well-known clustering algorithms for microarray, protein interaction and sequence data are available online. We summarize some of the most popular ones that are currently being used.

The most popular tool for cluster analysis and visualization of microarray data is **Cluster** [33] which was developed by Michael Eisen. It contains some of the basic clustering algorithms such as k-means, hierarchical and self-organizing maps. **TreeView** is a software (associated with Cluster software) that provides a graphical user interface to browse through the clustering results. Both of these work only in the Windows environment and can be downloaded from `http://rana.lbl.gov/EisenSoftware.htm`.

Michiel de Hoon of the University of Tokyo has created a version of Cluster (called **Cluster 3.0**) [23] that implements the same algorithms for different platforms. Routines for hierarchical (pairwise simple, complete, average, and centroid linkage) clustering, k-means and k-medians clustering, and 2D self-organizing maps are included. This software can be downloaded from `http://bonsai.hgc.jp/~mdehoon/software/cluster/`. Extensions of these modules to Python and Perl languages are also available.

Java TreeView is a Java-based visualization software that can be used to view the original microarray data and also the corresponding clustering results [88]. This is an open source software which can be downloaded from `http://sourceforge.net/projects/jtreeview/`.

EXCAVATOR (EXpression data Clustering Analysis and VisualizATiOn Resource) was developed by the Protein Informatics Group of Oak Ridge National Laboratory. It uses concepts from graph theory (such as minimum spanning trees) to represent gene expression data. In addition to providing different distance measures for computing clusters, it contains many other additional features such as automatic selection of the number of clusters, background noise removal, and identifying gene expression profiles that are similar to a specified set of seed genes. This software is available at http://digbio.missouri.edu/software/Excavator/index.html. **EMMIX-GENE** (available at http://www.maths.uq.edu.au/~gjm/emmix-gene/) is a software for mixture model-based clustering for gene expression data.

EXPANDER (EXPression ANalyzer and DisplayER) is a gene expression analysis and visualization tool that contains several clustering methods such as *k*-means, hierarchical clustering, and self-organizing maps, and it enables visualizing the gene expression data and their clusters [94]. It is a Java-based tool which can be downloaded from http://acgt.cs.tau.ac.il/expander/. The **Pvclust** is an R package that can be used to assess the uncertainty in hierarchical cluster analysis. It calculates p-values for each cluster using bootstrap resampling technique [102]. It can be downloaded from http://www.is.titech.ac.jp/~shimo/prog/pvclust/. **Genesis** is a Java-based, platform-independet package for simultaneously analyzing and visualizing gene expression datasets. It contains several popular clustering algorithm implementations such as hierarchical clustering, self-organizing maps, and *k*-means clustering [101]. It can be downloaded from http://genome.tugraz.at/genesisclient/genesisclient_download.shtml. **wCLUTO** (available at http://glaros.dtc.umn.edu/gkhome/cluto/wcluto/overview) is a web-enabled data-clustering application that is designed for the clustering and data-analysis requirements of gene-expression analysis. wCLUTO is built on top of the CLUTO clustering library. Users can upload their datasets, select from a number of clustering methods, perform the analysis on the server, and visualize the final results. A **synthetic generator** for gene expression data is available at http://www.che.udel.edu/eXPatGen/. Such a simulator can generate hypothetical expression patterns that can be used to evaluate and compare different clustering methods.

A popular tool that implements many popular biclustering algorithms (such as CC, ISA, and OPSM) is the **Biclustering Analysis Toolbox (BicAT)** which is available at http://www.tik.ee.ethz.ch/sop/bicat/. BicAT is a Java-based software that provides a nice graphical user interface for analyzing gene expression data [12]. **TriCluster** [123] is the first triclustering algorithm for microarray expression clustering. Tricluster first mines all the biclusters across the gene-sample slices, and then it extends these into triclusters across time dimensions. This software can be downloaded from http://www.cs.rpi.edu/~zaki/software/TriCluster.tar.gz. **STEM** (Short Time-series Expression Miner) is a Java program for clustering, comparing, and visualizing short time-series gene-expression data from microarray experiments. It can be downloaded from http://www.cs.cmu.edu/~jernst/stem/.

clusterMaker [74] provides a unified platform for various traditional clustering and network clustering techniques. It is available at http://www.cgl.ucsf.edu/cytoscape/cluster/clusterMaker.html. All of the network partitioning cluster algorithms create collapsible "meta nodes" to allow interactive exploration. These clustering algorithms have been developed as a plugin to the Cytoscape software [93]. **Cytoscape** is an open source bioinformatics software platform for visualizing complex molecular interaction networks. It is available at http://www.cytoscape.org/.

CFinder is a free software for finding and visualizing overlapping dense groups of nodes in networks, based on the Clique Percolation Method (CPM) [1]. It is a fast program for locating and visualizing overlapping, densely interconnected groups of nodes in undirected graphs, and allowing the user to easily navigate between the original graph and the web of these groups. This software is available at http://www.cfinder.org/. The code for **Markov clustering** is available at http://www.micans.org/mcl/. It provides a collection of network analysis tools focused on analysis of very large networks, scaling up to millions of nodes and hundreds of millions of edges. The software for **MCODE** algorithm [9] is made available in the form of a Cytoscape plugin at http://baderlab.org/Software/MCODE. The **network analysis tools (NeAT)** [15] provides a user-friendly web access to a collection of modular tools for the analysis of networks (graphs) and clusters. It includes a set of tools that support basic graph operations and clustering algorithms. NeAT is designed to cope with large datasets and provides a flexible toolbox for analyzing biological networks stored in various databases or obtained from high-throughput experiments. This software is available at `http://rsat.ulb.ac.be/neat/`.

CD-HIT is a very widely used program for clustering protein and DNA sequence. It is available from `http://weizhong-lab.ucsd.edu/cd-hit/`. The website has a server that provides CD-HIT services online. Alternatively, the user can also download the program for off-line use. The CD-HIT package has various sequence clustering tools that are customized for different biological sequences. The standard BLAST package includes a program called BLASTclust that can be used to cluster either protein or DNA sequences. BLASTclust accepts a number of parameters that can be used to control the stringency of clustering including thresholds for score density, percent identity, and alignment length. Besides clustering, BLASTclust can also be used to create a nonredundant set of sequences from a source database. The following website `http://toolkit.tuebingen.mpg.de/blastclust` provides access to a blastClust server.

16.6 Discussion and Summary

This chapter provides a survey of different data clustering techniques that have been successfully applied in the context of biological data. We have discussed different types of biological data where data clustering has produced promising and biologically meaningful results. For clustering gene expression data, different categories of clustering, namely, biclustering, triclustering, and time-series clustering, have been discussed along with some of the standard clustering algorithms. Different properties of biological networks along with the most widely studied network clustering algorithms have also been discussed. Various biological sequence similarity measures and clustering algorithms have been studied. Along with the clustering algorithms, we also have discussed the biological validation of the clusters obtained.

As more and more biologists become familiar with the advancements in data clustering algorithms, we can envision a data-driven biological science instead of a hypothesis-driven science. In addition, we also hope that the new insights that these clustering algorithms provide can drive a new set of biologically constrained experiments that can capture better insights about cellular functions.

Bibliography

[1] B. Adamcsek, G. Palla, I. J. Farkas, I. Derenyi, and T. Vicsek. CFinder: Locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22(8):1021–1023, 2006.

[2] J. S. Aguilar-Ruiz. Shifting and scaling patterns from gene expression data. *Bioinformatics*, 21(20):3840–3845, October 2005.

[3] U. Alon, N. Barkai, D. A. Notterman, K. Gishdagger, S. Ybarradagger, D. Mackdagger, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences of the United States of America*, 96(12):6745–6750, June 1999.

[4] F. Alqadah, J. S. Bader, R. Anand, and C. K. Reddy. Query-based biclustering using formal concept analysis. In *Proceedings of SIAM International Conference on Data Mining*, pages 648–659, 2012.

[5] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[6] I. P. Androulakis, E. Yang, and R. R. Almon. Analysis of time-series gene expression data: Methods, challenges, and opportunities. *Annual Review of Biomedical Engineering*, 9:205–228, 2007.

[7] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene-Ontology: Tool for the unification of biology. The Gene Ontology Consortium. *Nature Genetics*, 25(1):25–29, 2000.

[8] S. Asur, D. Ucar, and S. Parthasarathy. An ensemble framework for clustering protein-protein interaction networks. *Bioinformatics*, 23(13):i29–i40, 2007.

[9] G. Bader and C. Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4(1):2, 2003.

[10] C. Bahlmann and H. Burkhardt. Measuring HMM similarity with the Bayes probability of error and its application to online handwriting recognition. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, ICDAR '01, pages 406–411, 2001.

[11] Z. Bar-Joseph. Analyzing time series gene expression data. *Bioinformatics*, 20(16):2493–2503, 2004.

[12] S. Barkow, S. Bleuler, A. Prelic, P. Zimmermann, and E. Zitzler. BicAT: A biclustering analysis toolbox. *Bioinformatics*, 22(10):1282–1283, 2006.

[13] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: The order-preserving submatrix problem. *Journal of Computational Biology*, 10(3–4):373–384, 2003.

[14] A. Ben-Dor and Z. Yakhini. Clustering gene expression patterns. In *Proceedings of the Third Annual International Conference on Computational Molecular Biology*, RECOMB '99, pages 33–42, New York, USA, 1999.

[15] S. Brohée, K. Faust, G. Lima-Mendez, O. Sand, R. Janky, G. Vanderstocken, Y. Deville, and J. van Helden. NeAT: A toolbox for the analysis of biological networks, clusters, classes and pathways. *Nucleic Acids Research*, 36(Web Server issue), July 2008.

[16] S. Brohee and J. van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*, 7(1):488, 2006.

[17] J. Burke, D. Davison, and W. Hide. d2_cluster: A validated method for clustering EST and full-length cDNAsequences. *Genome Research*, 9(11):1135–1142, November 1999.

[18] S. Busygin, O. Prokopyev, and P. M. Pardalos. Biclustering in data mining. *Computers and Operations Research*, 35(9):2964–2987, 2008.

[19] A.J. Butte and I.S. Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. In *Pacific Symposium on Biocomputing*, volume 5, pages 418–429, 2000.

[20] C. Cheadle, M. P. Vawter, W. J. Freed, and K. G. Becker. Analysis of microarray data using Z score transformation. *Journal of Molecular Diagnostics*, 5(2):73–81, May 2003.

[21] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 93–103, 2000.

[22] S. Datta and S. Datta. Comparisons and validation of statistical clustering techniques for microarray gene expression data. *Bioinformatics*, 19(4):459–466, 2003.

[23] M. J. L. de Hoon, S. Imoto, J. Nolan, and S. Miyano. Open source clustering software. *Bioinformatics*, 20(9):1453–1454, 2004.

[24] F. De Smet, J. Mathys, K. Marchal, G. Thijs, B. De Moor, and Y. Moreau. Adaptive quality-based clustering of gene expression profiles. *Bioinformatics*, 18(5):735–746, 2002.

[25] R. De Smet and K. Marchal. Advantages and limitations of current network inference methods. *Nature Reviews Microbiology*, 8(10):717–729, October 2010.

[26] D. Dembele and P. Kastner. Fuzzy c-means method for clustering micrroarray data. *Bioinformatics*, 19(8):973–980, 2003.

[27] M. Deodhar, G. Gupta, J. Ghosh, H. Cho, and I. Dhillon. A scalable framework for discovering coherent co-clusters in noisy data. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 241–248, 2009.

[28] P. D'haeseleer, S. Liang, and R. Somogyi. Genetic network inference: From co-expression clustering to reverse engineering. *Bioinformatics*, 16(8):707–726, 2000.

[29] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel *k*-means: Spectral clustering and normalized cuts. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 551–556, 2004.

[30] T. Dhollander, Q. Sheng, K. Lemmens, B. De Moor, K. Marchal, and Y. Moreau. Query-driven module discovery in microarray data. *Bioinformatics*, 23(19):2573–2580, 2007.

[31] S. Dongen. A cluster algorithm for graphs. Technical report, CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands, 2000.

[32] S. Dudoit and J. Fridlyand. Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, 19(9):1090–1099, 2003.

[33] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, December 1998.

[34] A. J. Enright and C. A. Ouzounis. GenerAGE: A robust algorithm for sequence clustering and domain detection. *Bioinformatics*, 16(5):451–457, 2000.

[35] J. Ernst and Z. Bar-Joseph. STEM: a tool for the analysis of short time series gene expression data. *BMC Bioinformatics*, 7(1):191, 2006.

[36] J. Ernst, G.J. Nau, and Z. Bar-Joseph. Clustering short time series gene expression data. *Bioinformatics*, 21(Suppl 1):i159–i168, 2005.

[37] G. Fang, R. Kuang, G. Pandey, M. Steinbach, C. L. Myers, and V. Kumar. Subspace differential coexpression analysis: Problem definition and a general approach. *Pacific Symposium on Biocomputing*, pages 145–156, 2010.

[38] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

[39] M. E. Garber, O. G. Troyanskaya, K. Schluens, S. Petersen, Z. Thaesler, Pacyna M. Gengelbach, M. van de Rijn, G. D. Rosen, C. M. Perou, R. I. Whyte, and others. Diversity of gene expression in adenocarcinoma of the lung. *Proceedings of the National Academy of Sciences*, 98(24):13784–13789, 2001.

[40] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, pages 625–628, Washington, DC, USA, 2005.

[41] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proceedings of National Academy of Sciences of the United States of America*, 97:12079–12084, 2000.

[42] R. Gill, S. Datta, and S. Datta. A statistical framework for differential network analysis from microarray data. *BMC Bioinformatics*, 11(1):95, 2010.

[43] F. Glover. Tabu search—Part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[44] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, October 1999.

[45] V. Guralnik and G. Karypis. A scalable algorithm for clustering sequential data. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 179–186, 2001.

[46] D. Gusfield. *Algorithms on strings, trees, and sequences*. Cambridge University Press, 1997.

[47] S. Han, S. G. Lee, K. H. Kim, C. J. Choi, Y. H. Kim, and K. S. Hwang. CLAGen: A tool for clustering and annotating gene sequences using a suffix tree algorithm. *BioSystems*, 84(3):175–182, June 2006.

[48] D. Hanisch, A. Zien, R. Zimmer, and T. Lengauer. Co-clustering of biological networks and gene expression data. *Bioinformatics*, 18(suppl 1):S145–S154, 2002.

[49] J. Herrero, A. Valencia, and J. Dopazo. A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics*, 17(2):126–136, 2001.

[50] L. J. Heyer, S. Kruglyak, and S. Yooseph. Exploring expression data: Identification and analysis of coexpressed genes. *Genome Research*, 9(11):1106–1115, November 1999.

[51] J. Ihmels, S. Bergmann, and N. Barkai. Defining transcription modules using large-scale gene expression data. *Bioinformatics*, 20(13):1993–2003, 2004.

[52] H. Jeong, S. P. Mason, A. L. Barabasi, and Z. N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, 2001.

[53] L. Ji and K. Tan. Identifying time-lagged gene clusters using gene expression data. *Bioinformatics*, 21(4):509–516, 2005.

[54] L. Ji and K. Tan. Mining gene expression data for positive and negative co-regulated gene clusters. *Bioinformatics*, 20(16):2711–2718, 2004.

[55] D. Jiang, J. Pei, M. Ramanathan, C. Tang, and A. Zhang. Mining coherent gene clusters from gene-sample-time microarray data. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 430–439, 2004.

[56] D. Jiang, C. Tang, and A. Zhang. Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386, November 2004.

[57] H. Jiang, S. Zhou, J. Guan, and Y. Zheng. gTRICLUSTER: A more general and effective 3d clustering algorithm for gene-sample-time microarray data. In *Proceedings of the 2006 International Conference on Data Mining for Biomedical Applications*, BioDM'06, pages 48–59, 2006.

[58] K. Kailing, H. Kriegel, and P. Kroger. Density-connected subspace clustering for high-dimensional data. In *SDM*, pages 256–257, 2004.

[59] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, Vol. 48, 96–129, 1998.

[60] H. Kawaji, Y. Yamaguchi, H. Matsuda, and A. Hashimoto. A graph-based clustering method for a large set of sequences using a graph partitioning algorithm. *Genome Informatics*, 12:93–102, 2001.

[61] G. C. Kennedy, H. Matsuzaki, S. Dong, W. M. Liu, J. Huang, G. Liu, X. Su, M. Cao, W. Chen, J. Zhang, W. Liu, G. Yang, X. Di, T. Ryder, Z. He, U. Surti, M. S. Phillips, Boyce M. T. Jacino, S. P. Fodor, and K. W. Jones. Large-scale genotyping of complex DNA. *Nature Biotechnology*, 21(10):1233–1237, 2003.

[62] S. Kim and J. Lee. BAG: A graph theoretic sequence clustering algorithm. *International Journal of Data Mining and Bioinformatics*, 1(2):178–200, 2006.

[63] A. D. King, N. Przulj, and I. Jurisica. Protein complex prediction via cost-based clustering. *Bioinformatics*, 20(17):3013–3020, 2004.

[64] T. Kohonen. *Self-organization and associative memory*. 3rd edition. Springer-Verlag, New York, NY, USA, 1989.

[65] N. J. Krogan, G. Cagney, H. Yu, G. Zhong, X. Guo, A. Ignatchenko, J. Li, S. Pu, N. Datta, et al. Global landscape of protein complexes in the yeast Saccharomyces cerevisiae. *Nature*, 440(7084):637–643, March 2006.

[66] C. Leslie and R. Kuang. Fast string kernels using inexact matching for protein sequences. *Journal of Machine Learning Research*, 5:1435–1455, 2004.

[67] C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.

[68] J. Li, K. Sim, G. Liu, and L. Wong. Maximal quasi-bicliques with balanced noise tolerance: Concepts and co-clustering applications. In *Proceedings of the SIAM International Conference on Data Mining SDM'08*, pages 72–83, April 2008.

[69] J. Liu, Z. Li, X. Hu, and Y. Chen. Biclustering of microarray data with mospo based on crowding distance. *BMC Bioinformatics*, 10(Suppl 4):S9, 2009.

[70] Y. Luan and H. Li. Clustering of time-course gene expression data using a mixed-effects model with B-splines. *Bioinformatics*, 19(4):474–482, 2003.

[71] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[72] K. Malde, E. Coward, and I. Jonassen. Fast sequence clustering using a suffix array algorithm. *Bioinformatics*, 19(10):1221–1226, 2003.

[73] V. Miele, S. Penel, and L. Duret. Ultra-fast sequence clustering from similarity networks with silix. *BMC Bioinformatics*, 12(1):116, 2011.

[74] J. H. Morris, L. Apeltsin, A. M. Newman, J. Baumbach, T. Wittkop, G. Su, G. D. Bader, and T. E. Ferrin. clusterMaker: A multi-algorithm clustering plugin for Cytoscape. *BMC Bioinformatics*, 12(1):436+, November 2011.

[75] O. Odibat and C. K. Reddy. A generalized framework for mining arbitrarily positioned overlapping co-clusters. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 343–354, 2011.

[76] O. Odibat, C. K. Reddy, and C. N. Giroux. Differential biclustering for gene expression analysis. In *Proceedings of the ACM Conference on Bioinformatics and Computational Biology (BCB)*, pages 275–284, 2010.

[77] Y. Okada and T. Inoue. Identification of differentially expressed gene modules between two-class DNA microarray data. *Bioinformation*, 4(4):134–137, 2009.

[78] G. Palla, A. L. Barabási, and T. Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, 2007.

[79] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[80] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences*, 85(8):2444–2448, 1988.

[81] W. Pentney and M. Meila. Spectral clustering of biological sequence data. In *Proceedings of the 20th National Conference on Artificial Intelligence—Volume 2*, AAAI'05, pages 845–850, 2005.

[82] C. Pizzuti and S. E. Rombo. A coclustering approach for mining large protein-protein interaction networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9:717–730, 2012.

[83] I. Priness, O. Maimon, and I. Ben-Gal. Evaluation of gene-expression clustering via mutual information distance measure. *BMC Bioinformatics*, 8(1):111, 2007.

[84] N. Przulj, D. A. Wigle, and I. Jurisica. Functional topology in a network of protein interactions. *Bioinformatics*, 20(3):340–348, 2004.

[85] S. Pu, J. Vlasblom, A. Emili, J. Greenblatt, and S. J. Wodak. Identifying functional modules in the physical interactome of Saccharomyces cerevisiae. *Proteomics*, 7(6):944–960, March 2007.

[86] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition, In Alex Waibel and Kai-Fulee (Eds.) *Readings in Speech Recognition*, pages 267–296. Morgan Kaufmann, San Francisco, CA, 1990.

[87] M. D. Robinson, J. Grigull, N. Mohammad, and T. R. Hughes. FunSpec: A web-based cluster interpreter for yeast. *BMC Bioinformatics*, 3:35+, 2002.

[88] A. J. Saldanha. Java Treeview—Extensible visualization of microarray data. *Bioinformatics*, 20(17):3246–3248, November 2004.

[89] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5): 513–523, 1988.

[90] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[91] A. Schliep, A. Schonhuth, and C. Steinhoff. Using hidden Markov models to analyze gene expression time course data. *Bioinformatics*, 19(Suppl 1):i255–i263, 2003.

[92] B. Schwikowski, P. Uetz, and S. Fields. A network of protein-protein interactions in yeast. *Nature Biotechnology*, 18(12):1257–1261, December 2000.

[93] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, November 2003.

[94] R. Sharan, A. Maron-Katz, and R. Shamir. CLICK and EXPANDER: A system for clustering and visualizing gene expression data. *Bioinformatics*, 19(14):1787–1799, 2003.

[95] R. Sharan and R. Shamir. Click: A clustering algorithm with applications to gene expression analysis. In *ISMB*, pages 307–316, 2000.

[96] R. Sharan, I. Ulitsky, and R. Shamir. Network-based prediction of protein function. *Molecular Systems Biology*, 3(1), March 2007.

[97] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[98] K. Sim, Z. Aung, and V. Gopalkrishnan. Discovering correlated subspace clusters in 3d continuous-valued data. In *IEEE 10th International Conference on Data Mining (ICDM), 2010*, pages 471–480. IEEE, 2010.

[99] P. Smyth. Clustering sequences with hidden Markov models. In *Advances in Neural Information Processing Systems*, volume 9, pages 648–654, MIT Press, Cambridge, MA, 1997.

[100] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. *KDD Workshop on Text Mining*, Vol. 400, pages 109–111, 2000.

[101] A. Sturn, J. Quackenbush, and Z. Trajanoski. Genesis: Cluster analysis of microarray data. *Bioinformatics*, 18(1):207–208, 2002.

[102] R. Suzuki and H. Shimodaira. Pvclust: An R package for assessing the uncertainty in hierarchical clustering. *Bioinformatics*, 22(12):1540–1542, 2006.

[103] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander, and T. R. Golub. Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proceedings of the National Academy of Sciences of the United States of America*, 96(6):2907–2912, March 1999.

[104] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. Systematic determination of genetic network architecture. *Nature Genetics*, 22(3):281–285, July 1999.

[105] D. Ucar, S. Asur, U. Catalyurek, and S. Parthasarathy. Improving functional modularity in protein-protein interactions graphs using hub-induced subgraphs. *Lecture Notes in Computer Science*, 4213:371, 2006.

[106] S. Van Dongen. Graph clustering by flow simulation. PhD Thesis. University of Utrecht, 2000.

[107] J. Vlasblom and S. Wodak. Markov clustering versus affinity propagation for the partitioning of protein interaction graphs. *BMC Bioinformatics*, 10(1):99, 2009.

[108] K. Voevodski, M. Balcan, H. Röglin, S. Teng, and Y. Xia. Active clustering of biological sequences. *Journal of Machine Learning Research*, 13:203–225, 2012.

[109] B. H. Voy, J. A. Scharff, A. D. Perkins, A. M. Saxton, B. Borate, E. J. Chesler, L. K. Branstetter, and M. A. Langston. Extracting gene networks for low-dose radiation using graph theoretical algorithms. *PLoS Computational Biology*, 2(7):e89, 2006.

[110] J. Wang, M. Li, Y. Deng, and Y. Pan. Recent advances in clustering methods for protein interaction networks. *BMC Genomics*, 11(Suppl 3):S10, 2010.

[111] J. Wang, Y. Zhang, L. Zhou, G. Karypis, and C. C. Aggarwal. Contour: An efficient algorithm for discovering discriminating subsequences. *Data Mining and Knowledge Discovery*, 18(1):1–29, 2009.

[112] T. Xiong, S. Wang, Q. Jiang, and J. Z. Huang. A new Markov model for clustering categorical sequences. In *Proceedings of the IEEE 11th International Conference on Data Mining*, ICDM '11, pages 854–863, 2011.

[113] X. Xu, Y. Lu, K. Tan, and A. K. H. Tung. Finding time-lagged 3d clusters. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, ICDE '09, pages 445–456. IEEE Computer Society, 2009.

[114] Y. Xu, V. Olman, and D. Xu. Clustering gene expression data using a graph-theoretic approach: An application of minimum spanning trees. *Bioinformatics*, 18(4):536–545, 2002.

[115] J. Yang, H. Wang, W. Wang, and P. Yu. Enhanced biclustering on expression data. In *Proc. of 3rd IEEE Symposium on BioInformatics and BioEngineering*, pages 321–327, 2003.

[116] J. Yang and W. Wang. CLUSEQ: Efficient and effective sequence clustering. In *Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE)*, pages 101–112, 2003.

[117] W. Yang, D. Dai, and H. Yan. Finding correlated biclusters from gene expression data. *IEEE Transactions on Knowledge and Data Engineering*, 23(4):568–584, April 2011.

[118] K. Y. Yeung, C. Fraley, A. Murua, A. E. Raftery, and W. L. Ruzzo. Model-based clustering and data transformations for gene expression data. *Bioinformatics*, 17(10):977–987, 2001.

[119] K. Y. Yeung, D. R. Haynor, and W. L. Ruzzo. Validating clustering for gene expression data. *Bioinformatics*, 17(4):309–318, 2001.

[120] K. Y. Yeung and W. L. Ruzzo. Principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763–774, 2001.

[121] M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1):31–60, 2001.

[122] H. Zhao, L. Cloots, T. Van den Bulcke, Y. Wu, R. De Smet, V. Storms, P. Meysman, K. Engelen, and K. Marchal. Query-based biclustering of gene expression data using probabilistic relational models. *BMC Bioinformatics*, 12(Suppl 1):S37, 2011.

[123] L. Zhao and M. J. Zaki. TRICLUSTER: An effective algorithm for mining coherent clusters in 3d microarray data. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 694–705. ACM Press, New York, 2005.