A roadmap of clustering algorithms: finding a match for a biomedical application

Bill Andreopoulos, Aijun An, Xiaogang Wang and Michael Schroeder

Submitted: 28th August 2008; Received (in revised form): 12th December 2008

Abstract

Clustering is ubiquitously applied in bioinformatics with hierarchical clustering and *k*-means partitioning being the most popular methods. Numerous improvements of these two clustering methods have been introduced, as well as completely different approaches such as grid-based, density-based and model-based clustering. For improved bioinformatics analysis of data, it is important to match clusterings to the requirements of a biomedical application. In this article, we present a set of desirable clustering features that are used as evaluation criteria for clustering algorithms. We review 40 different clustering algorithms of all approaches and datatypes. We compare algorithms on the basis of desirable clustering features, and outline algorithms' benefits and drawbacks as a basis for matching them to biomedical applications.

Keywords: *clustering; protein; gene expression; interactome; network; bioinformatics*

INTRODUCTION

Clustering in biomedicine is traced back to Greek antiquity when Aristotle attempted to classify living organisms. Clustering algorithms are developed for datasets that are too large and complex for manual analysis [1]. Not all existing clustering algorithms have acquired prominence in bioinformatics and many remain in obscurity [2, 3]. Clustering partitions objects into clusters, such that objects with similar characteristics are clustered together and dissimilar objects are in different clusters [4–6]. Clustering is a form of *unsupervised learning*. This means that no prior knowledge exists on any object classifications. Unsupervised learning differs from *supervised learning*, such as support vector machines (SVMs) or decision trees, where prior knowledge of several objects' classifications is used for training the classification algorithm.

Clustering in bioinformatics involves two groups of users, both of which need to understand what algorithmic features a biological application requires. One user group includes biologists with experience on the underlying biological problem, who apply existing clustering algorithms to solve the problem. The challenge is to choose a suitable algorithm from the toolbox, since each algorithm will produce different results. For instance, in clustering gene expression data a biologist wishes to mix numerical expression levels with discrete Gene Ontology (GO) categorization. Another user group includes computer scientists who develop novel bioinformatics algorithms. This group assumes current algorithms

Corresponding author. Bill Andreopoulos. E-mail: williama@biotec.tu-dresden.de

Bill Andreopoulos is a Post-Doctoral researcher at the Biotechnological Centre, Technische Universitaet Dresden, Germany. He completed his PhD at the Department of Computer Science and Engineering, York University, Toronto, Canada (2006), and his MSc at the Department of Computer Science of University of Toronto (2001). His interests include clustering, data mining, bioinformatics.
Aijun An is an Associate Professor at the Department of Computer Science and Engineering, York University, Toronto, Canada. She received her PhD in Computer Science from the University of Regina.
Xiaogang Wang is an Associate Professor at the Department of Mathematics and Statistics, York University, Toronto, Canada. He received his PhD in Statistics from the University of British Columbia.
Michael Schroeder is Professor of Bioinformatics at the Biotechnological Centre and Department of Computer Science, Technische Universitaet Dresden, Germany. He received his PhD from the University of Hanover and Lisbon (1997).

are insufficient for the underlying biological problem, and that progress requires improved methods. For instance, clustering large sequence databanks requires an algorithm that is fast and not affected by redundant sequences. There is significant overlap between these two user groups, since applications often stimulate algorithmic development.

Biomedical applications have different requirements and clustering algorithms have various features. To evaluate a clustering algorithm's suitability for a problem, we use a general set of desirable features [7, 8].

Scalability: The runtime and memory requirements should not explode on large (high-dimensional) datasets [9]. For example, clustering large sequence datasets is computationally intensive [10]. Krause *et al.* used 25 Linux machines to cluster nearly one million non-redundant sequences in 3 months [11]. Li *et al.* used a Linux workstation with dual 3.0 GHz Xeon processors and 4 GB RAM to cluster 3.2 million proteins at 90% sequence identity level in <8 h [12]. In such a setting scalability of clustering algorithms is essential. For much smaller gene expression datasets, most clustering algorithms perform satisfactorily on a desktop computer and thus scalability is not an impotant criterion in this setting.

Robustness: Ability to detect outliers that are distant from the rest of the samples. Outliers may indicate objects that belong to a different population. For example, cancer genes (oncogenes) may be activated, over- or down-expressed, only in a small number of samples [13].

Order insensitivity: A clustering algorithm should not be sensitive to the ordering of the input objects. For example, reordering the proteins in an interactome dataset should not result in different clusters [14]. Order insensitivity is important for every application, as it is key to ensure reproducability of results.

Minimum user-specified input: Parameters, such as the number of clusters, will affect the result [1, 4]. For example, in gene expression data specifying the number of clusters will result in different groups of co-regulated genes [2, 3, 15].

Mixed datatypes: Objects may have numerical descriptive attributes, such as a set of genes expressed at different levels over time; and discrete (categorical) descriptive attributes, such as genes with GO annotations [16–18], or interactomes where a protein is described by its connections to other proteins [19].

Arbitrary-shaped clusters: A clustering algorithm should find arbitrary shaped clusters [20]. Figure 1 shows coinduced genes that present expression

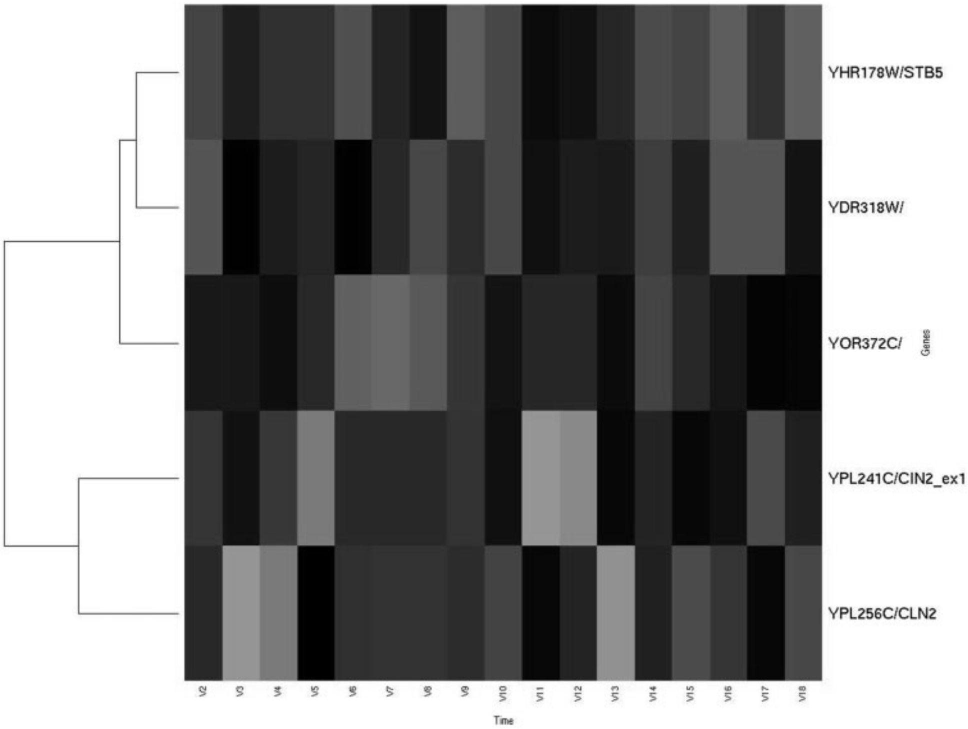

Figure 1: Each row represents a yeast gene's expression pattern over 17 time points [21]. Coinduced genes present local similarities and time-shifted relationships.

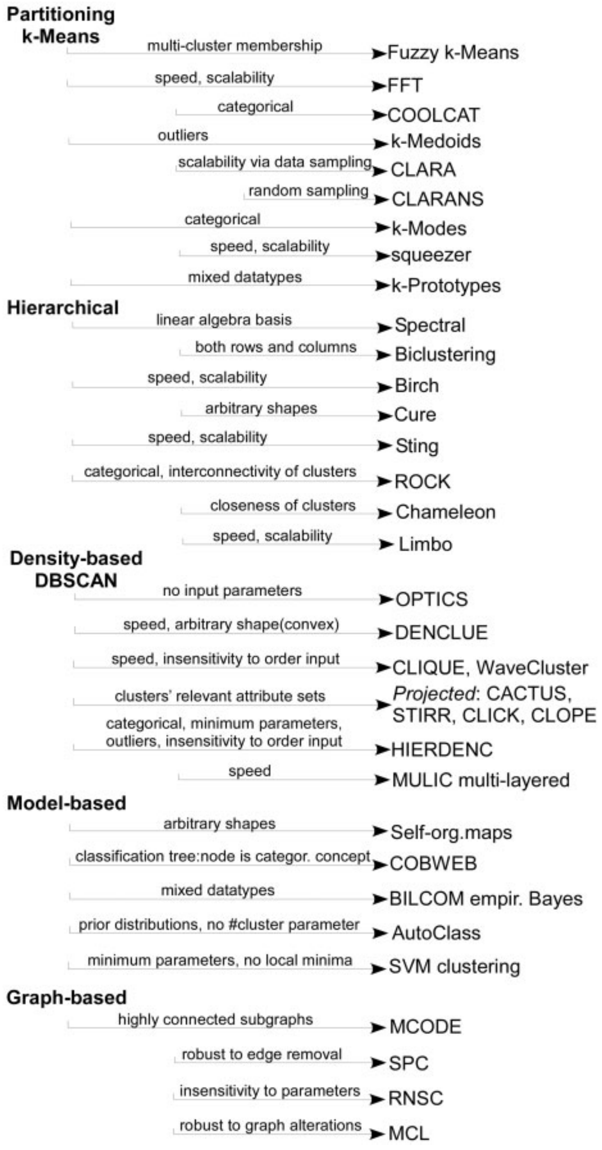

Figure 2: Classification of clustering algorithms.

patterns with local similarities and time-shifted relationships [21–23]. Some algorithms can identify such clusters, while others find global similarity in genes' expression patterns. Cluster shapes are meaningful only for numerical data.

Point proportion admissibility: Duplicating objects and re-clustering should not change the result [20]. For example, Krause *et al*. removed nearly 200 000 redundant sequences, which were at least 80% identical over at least 80% of their length to another sequence, and clustered the remaining ~1 000 000 sequences [11]. In contrast, in gene expression datasets there are usually no identical expression patterns in a series.

Evaluation of clustering quality is application-dependent, with choices for quality measures including: precision/recall to a gold standard [24]; entropy of the clusters [25]; ontology annotation enrichment [17, 26]; reproducibility [27]; Hubert-Arabie Indices, the number of pairs of objects that are correctly in the same or different clusters divided by all pairs of objects [28–30].

The objectives of this article are: *a.* to survey important clustering applications in biomedicine, *b.* to explain benefits and drawbacks of existing clustering algorithms, *c.* to provide guidelines on selecting a clustering algorithm for an application.

This article is organized as follows. First, we outline different biomedical applications' requirements. Next, we provide a classification of clustering algorithms based on features that they inherit from one another, as shown in Figure 2; refinement algorithms inherit and improve upon root algorithms' features. Then, we compare the features of 40 clustering algorithms, divided by approach and datatype, and explain their utility for biomedical applications.

BIOMEDICAL APPLICATIONS

A suitable clustering algorithm depends on the application and datatype [8]. Next, we outline applications to gene expression, interactomes and sequences.

Gene expression

A *gene expression* dataset contains measurements of increasing or decreasing *expression levels* of a set of genes [2, 3, 7, 31]. A number of gene expression measurements are usually taken, across *time* points, *tissue* samples, or *patients*. It is represented as a matrix of numerical values: gene versus time, gene versus tissue, gene versus patient. The clusters contain similar objects with respect to a metric, representing genes of similar functionality, or tissues of similar profiles, or patients of similar clinical outcomes [32–37].

There may be thousands of genes expressed only at low levels. Desirable clustering features include *minimum user input*, since small changes in parameters will lead to different clusterings [38]. Another desirable feature is ability to cluster *mixed datatypes* [22]; especially for disease data, it is desirable to complement the expression data with discrete data such as patient sex, age group, etc. [39–41]. Arbitrary-shaped clusters are desirable since gene expression patterns are locally rather than globally similar and potentially time-shifted, as Figure 1 shows [21, 22, 42, 43, 143]. Moreover, *robustness to noise and outliers* is desirable. Outliers appear in tumor expression, where cancer is divided into subtypes, and oncogenes are activated in few subtypes of a cancer group; without outlier detection, subtypes may be misclassified in clusters [13].

Microarrays allow to compare expression in tumors and healthy tissues for several patients. With the amount of tumor expression data increasing rapidly, cancer is a common clustering application [44]. Tumor clustering typically places tumor samples into classes with different clinical behavior and overall survival [2, 45, 46]. The ultimate aim is targeting specific treatments to distinct tumor types [47]. The best treatment can differ for each tumor, e.g. prostate cancers may follow various clinical courses after treatment [48, 49]. Tumor classification involves: (i) Clustering tumor samples into groups of similar behavior [3]. It is difficult to determine whether clusters are meaningful, reflecting true structure in the data or random aggregation [50, 51]. The noise problem is aggravated by outlier oncogenes, or differentially expressed genes, which are activated for a small number of tumor samples [52]. (ii) Classifying a new tumor sample correctly. Gruetzmann *et al*. identified 568 differentially expressed genes that were consistently up- or down-regulated in pancreatic cancer, and could represent good candidates for novel diagnostic and therapeutic approaches to pancreatic cancer [53]. Birnie *et al*. used clustering to describe an expression signature of 581 genes whose levels are significantly different in prostate cancer stem cells, and found tissue-specific signalling pathways [54].

Networks

Protein–protein interaction networks (*interactomes*) consist of nodes representing biomolecules (proteins), and edges representing interactions [55–59]. An interactome can be represented as a square matrix, where a nonzero point means two proteins interact, and zero otherwise. Clustering interactomes often aims to predict complexes, where a complex is a group of proteins interacting at the same time and cellular location [60–64]. Desirable features include *robustness to noise and outliers*, since interactomes have many errors and clustering is sensitive to network alterations [65]. *User-specified parameters* should be minimal, since it is hard for users to specify correct values for parameters, such as the number of complexes. Integrating *mixed data types* such as GO annotations helps with predicting complexes, since three connected proteins do not imply all three interactions occur at the same time and location [16, 19].

Sequences can be clustered into protein families [66]. Protein sequence family relationships are typically represented as hierarchical [67, 68]. The sequences can be of genomic, 'transcriptomic' (ESTs) or protein origin [69]. The similarity is often based on BLAST sequence alignment [70, 11]. Determining a representative structure for each family is the aim of many initiatives [71]. With the tremendous growth of sequence databases, clustering must be scalable and fast. Because of the large size of sequence datasets, essential features include *order insensitivity*. *Point proportion admissibility* is also essential, since sequences in public databases are often redundant [72, 73]. Unlike gene expression or interactomes, these features are essential in dealing with sequences. The requirements of this domain pose a challenge for hierarchical grouping methods.

Regulatory networks model transcription factors activating or inhibiting the rates of gene transcription into mRNA. Genes in a cell interact with one another indirectly through their mRNA and protein products.

Synthetic mutant lethality searches for interactions between two genes, by examining if simultaneous mutations in both genes lead to death or another cell growth defect [74, 75]. This is explained by genetic redundancy; both genes perform the same function but each gene carries it out in a different way. Synthetic mutant lethality data is represented as a square matrix, where a point is set to one if the two genes are lethal in combination and zero otherwise.

Clustering roadmap

Clustering partitions N objects into k clusters. Object o has m attributes, $\{o_1, \ldots, o_m\}$ (usually $N \gg m$). Attribute o_i, $i = 1 \ldots m$, has a domain D_i of a datatype, such as numerical or discrete. Figure 2 depicts inheritance relationships between clustering algorithms. Root approaches are separated into algorithms with different features: partitioning (*k*-means), hierarchical, grid-based, density-based, model-based [8]. Refinement algorithms improve upon a root approach, inheriting the approach's features, while possibly introducing drawbacks. Table 1 compares algorithms' features (Section 1) and shows whether they are recommended for

Table 1: Clustering algorithm comparison

Algorithm	Complexity	Robust to outliers	Order independence	User input	Mixed datatypes	Arbitrary-shaped cluster	Point prop. admiss	Availability	Since	Gene expression	Interactomes	Sequences	Applied to biology
Partitioning (*k*-means)													
k-Means [76]	$O(tkN)$	No	No	1, 10	No	No	No	Matlab, Weka, R	1998				Yes
Farth. First Trav. [77]	$O(Nk)$	No	No	1	Yes	No	No	Weka	2001				No
k-Medoids (PAM) [78]	$O(tkN)$	Yes	No	1	No	No	Yes	Matlab, R	1987				Yes
CLARA [79]	$O(ks^2 + k(N-k))$	Yes	Yes	1	No	Yes	Yes	Code	1990	rec			No
CLARANS [80]	$O(N^2)$	Yes	Yes	1	No	Yes	Yes	Code	1994	rec			No
Fuzzy *k*-means [43, 81]	$O(tkN)$	No	No	1	No	Yes	No	Matlab, Weka	2002				Yes
k-Modes [82]	$O(tkN)$	No	No	1	No	–	No	Matlab, Weka, R	1998				Yes
Fuzzy *k*-modes [83]	$O(tkN)$	No	No	1	No	–	No	Matlab, Weka	1999				Yes
Squeezer [84]	$O(kN)$	No	No	13	Yes	No	No	Code	2006				No
k-Prototypes [85]	$O(tkN)$	No	No	1	Yes	No	No	Matlab, Weka	1997				No
COOLCAT [86]	$O(N^2)$	No	No	1	No	No	No	–	2002				No
CLICK (gene expr.) [36]	'Fast'	–	Yes	–	No	No	No	C	2000				Yes
Hierarchical													
Agglomerative single, average, complete-linkage [145, 147]	$O(N^2)$ single, $O(N^2 logN)$ average & complete	No	Yes	5, 15	Yes	Yes	No	R hclust	1999				Yes
Eisen gene expr. [15, 87]	$O(N^2)$ single, $O(N^2 logN)$ average & complete	No	Yes	5	Yes	Yes	No	Java appl.	1998				Yes
Spectral [88, 89]	$O(N)$ (roughly)	No	Yes	5	Yes	No	No	Matlab, R, BicAT	2001				Yes
BIRCH [90]	$O(N)$	Yes	Yes	–	No	No	Yes	C++	1996	rec	rec	rec	No
CURE [91]	$O(N)$	Yes	Yes	–	No	Yes	Yes	Code	1998	rec	rec	rec	No
ROCK [92]	$O(kN^2)$	No	Yes	1, 13	Yes	–	No	C	2000				No
Chameleon [93]	$O(N^2)$	Yes	Yes	13	No	Yes	Yes	Code, web service	1999				Yes
LIMBO [94]	$O(NlogN)$	Yes	Yes	14	No	–	No	C++	2004				No
hMETIS [95]	'Fast'	No	Yes	5, 10	No	No	No	Code, web service	1997				No
Power graphs [96]	$O(Nd^2)$	Yes	Yes	5, 10, 12, 13	Yes	–	Yes	Cytoscape	2008		rec		Yes
Density-based													
HIERDENC [97]	$O(N)$	Yes	Yes	–	Yes	–	No	Code	2007	rec	rec	rec	Yes
MULIC [14, 97]	$O(N^2)$	Yes	No	–	Yes	–	No	Code, web service, Cytoscape	2006		rec		Yes
DBSCAN [98]	$O(NlogN)$	Yes	Yes	3, 7	No	Yes	No	Weka	1998				Yes
OPTICS [99]	$O(NlogN)$	Yes	Yes	3, 7	No	Yes	No	Weka	1999				Yes
DENCLUE [100]	$O(N^2)$	Yes	No	7	No	Yes	Yes	Weka	1998				No

(continued)

Table 1: Continued

Algorithm	Complexity	Robust to outliers	Order independence	User input	Mixed datatypes	Arbitrary-shaped cluster	Point prop. admiss	Availability	Since	Gene expression	Interactomes	Sequences	Applied to biology
CACTUS [101]	’Scalable’	No	Yes	1, 4	No	No	No	Code	1999				No
STIRR [102]	’Scalable’	No	No	12	No	No	No	–	1998				No
CLICK (categ.) [103]	‘Scalable’	No	Yes	–	No	–	No	Code	2005				No
CLOPE [104]	$O(kdN)$	No	Yes	–	No	No	No	–	2002				No
WaveCluster [105]	$O(N)$	Yes	Yes	8, 9	No	Yes	No	Code	1998				No
STING [106]	$O(N)$	Yes	Yes	–	No	No	No	Code	1997	rec	rec		No
CLIQUE [107]	$O(N)$	Yes	Yes	3,8	Yes	Yes	Yes	Code	1998			rec	No
Model-based													
SOMs (NeuralNet) [23]	$O(N^2)$	No	No	1, 2, 5	No	Yes	No	Matlab, Python, WebSOM	1999				Yes
COBWEB [108]	$O(Nd^2)$	Yes	No	–	No	–	No	Weka	1987		rec		No
BILCOM [109]	$O(N^2)$	Yes	No	5	Yes	–	No	Code	2006		rec		Yes
AutoClass (ExpMax) [110]	$O(kd^2Nt)$	Yes	Yes	–	Yes	Yes	No	Weka, C++, R mclust	1995	rec	rec		Yes
SVM clustering [111]	$O(N^{1.8})$	No	No	–	Yes	Yes	Yes	Matlab, C++, Java, SVMlight	2007				Yes
Graph-based													
MCODE [19]	$O(Nd^3)$	No	Yes	6	No	–	No	Cytoscape	2003				Yes
RNSC [112]	$O(N^2)$	No	Yes	1	No	–	No	Code upon request	2004				Yes
SPC [65, 70]	$O(N^2)$	Yes	Yes	1	No	Yes	No	Code upon request	1996				Yes
MCL [113]	$O(N^3)$	Yes	Yes	11	No	–	No	Web service	2002				Yes

Variables: *N* = *num objects, d* = *dimensionality, k* = *clusters, t* = *iterations, s* = *sample size*. The reported complexities are according to the authors. Since these are the worst case complexities, an algorithm may achieve faster runtimes on most datasets. User-specified parameters include: 1. number of clusters, 2. training data as background knowledge, 3. minimum number of objects or density threshold, 4. average number of dimensions, 5. threshold/cutoff for cluster discrimination, 6. degree (connectivity), 7. radius/maximum distance, 8. grid size, 9. wavelet transform, 10. stop criterion, 11. matrix inflation, 12. node/edge weights, 13. minimum similarity, 14. bound for memory/info loss, 15. dissimilarity measure.

an application. The $O(\cdot)$ notation describes how the size of the dataset affects an algorithm's runtimes; higher values are slower. In the next sections, we discuss these algorithms.

PARTITIONING CLUSTERING

In this approach, objects are partitioned and may change clusters based on dissimilarity [7]. Partitioning methods are useful for bioinformatics applications where a fixed number of clusters is desired, such as small gene expression datasets [38]. A drawback is that the user typically specifies the number of clusters as an input parameter.

Numerical

k-*Means*

In *k*-means, the user specifies the number of clusters *k* [76]. Clusters have a representative, which is the *mean vector*, for finding the closest cluster to an object, which minimizes mean-object dissimilarity with a metric such as Euclidean distance. *k*-Means iteratively assigns objects to the closest of *k* clusters, and the means get updated. Mean μ_C for cluster C with n_C objects contains the mean values of its members, and is defined for attribute i: $\sum_{j=1}^{n_C} X_{ij}/n_C$. The iteration continues until the number of objects changing clusters is below a user-specified threshold. In [82] two methods for selecting the initial means are discussed.

k-Means deals with numerical attribute values (NAs), but it is also applicable to binary datasets. *k*-Means has complexity $O(tkN)$, where t is the number of iterations. Typically k, $t \ll N$, so that *k*-means is essentially linear in the number of objects N. *k*-Means may terminate at a local rather than global optimum. *k*-Means is unsuitable for noise, since outliers may distort the representative cluster means. *k*-Means is unsuitable for discovering arbitrary shaped clusters (Figure 1). The result depends on the initial cluster means.

The *k*-means algorithm was recently parallelized for fast laboratory use [114]. Next, we discuss improvements upon *k*-means addressing its shortcomings: speed, user parameters, sensitivity to outliers and initial means, arbitrary shaped clusters.

Farthest First Traversal k-center (FFT) algorithm

The FFT algorithm improves the *k*-means complexity to $O(Nk)$. It deals with *k*-means' sensitivity to initial cluster means, by ensuring means represent the dataset variance. Like *k*-means, FFT is not suitable for noisy datasets, since the means might be outliers. K cluster centers are set. The first center is chosen randomly. Each remaining center is determined by greedily choosing the object farthest from the set of already chosen centers, where the furthest object, x, from a set, S, is defined as, $\max_x\{\min\{\text{distance}(x, j), j \in S\}\}$ [77]. Then, each remaining object is assigned to the cluster center with minimum distance.

k-Medoids or PAM: Partitioning Around Medoids

k-Medoids deals with *k*-means' problem of outliers, by setting a cluster's mean to the object that is nearest to the 'center' of the cluster [78]. *k*-Medoids involves reducing the distance between all objects in a cluster and the central object.

CLARA: Clustering Large Applications

k-Medoids does not scale well to large datasets. CLARA is an extension of *k*-medoids with a focus on scalability. CLARA selects a representative sample of the entire dataset. Medoids are then chosen from this sample, similar to *k*-medoids. If the sampling is done properly, the medoids chosen from the sample are similar to the ones that would have been chosen from the whole dataset. CLARA's effectiveness depends on the sample size. CLARA's complexity is $O(ks^2 + k(N - k))$ where s is the sample size [79].

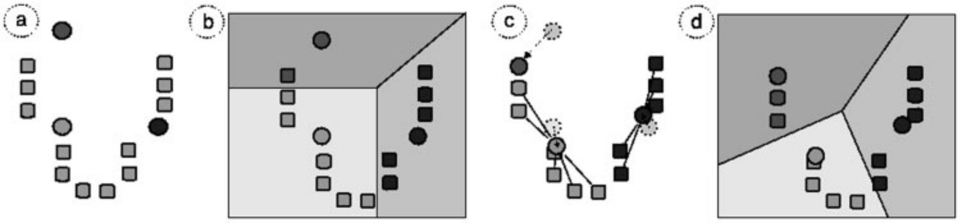

Figure 3: *k*-Means clustering. (a) Initial randomized means. (b) Objects are associated with the nearest mean. (c) Means are moved to the center of their respective clusters. (d) Last two steps are repeated until convergence.

CLARANS: Clustering Large Applications Based Upon Randomized Search

CLARANS improves upon CLARA by allowing the sample to change throughout clustering. Unlike CLARA, CLARANS does not limit itself to a fixed sample, but draws a sample with some randomness at each iteration. CLARANS supports outlier detection. CLARANS is more effective than *k*-medoids and CLARA. Its complexity is $O(N^2)$ [80].

Fuzzy k-means

The *k*-means clusters are 'hard', since an object either is or is not a member of a cluster. Fuzzy *k*-means produces 'soft' or 'fuzzy' clusters, where an object has a degree of membership in each cluster [81, 43]. Fuzzy *k*-means was applied to gene expression to study overlapping gene sets [115, 43].

Discrete

k-Modes

k-Modes is a discrete adaptation of *k*-means, with similar runtime, benefits and drawbacks [82]. The *mode* of cluster c is a vector $\mu_c = \{\mu_{c1}, \ldots, \mu_{cm}\}$, where μ_{ci} is the most frequent value in c for the ith attribute. The Hamming distance is used for finding an object's nearest cluster mode.

Fuzzy k-modes

Fuzzy *k*-modes extends fuzzy *k*-means to discrete data [83]. Objects' assignments to clusters involve degrees of membership.

Squeezer

Squeezer is a one-pass algorithm that improves upon the iteration-bound speed of *k*-modes [84]. Squeezer reads objects one-by-one. The first tuple forms a cluster alone. Next objects are either put into an existing cluster, or rejected by all to form a new cluster. Squeezer may not produce the most accurate clusterings. Squeezer is efficient with a complexity of $O(kN)$.

COOLCAT

COOLCAT deals with *k*-modes' sensitivity to the initial cluster modes. COOLCAT is sensitive to the order of object selection. Clusters are created by reducing their entropy [86]. COOLCAT finds a set of k maximally dissimilar objects to create initial clusters. All remaining objects are placed in one of the clusters, such that the increase in entropy is minimized [4].

Mixed Discrete and Numerical

k-Prototypes

An extension of *k*-modes called *k*-prototypes handles mixed datatypes [85]. *k*-Prototypes uses a distance metric that weighs the contribution of the numerical versus discrete attributes. *k*-Prototypes iterates until few objects change clusters.

HIERARCHICAL CLUSTERING

Hierarchical clustering algorithms partition the objects into a tree of nodes, where each node represents a cluster [116, 117]. Each node in a tree has zero or more child nodes, which are below it in the tree; by convention, trees grow down, not up as they do in nature. A node that has a child is called the child's parent node. A node has at most one parent. Hierarchical methods include:

Agglomerative: Initially, many small clusters are formed, which are merged based on their similarity. Finally, one cluster contains all objects.

Divisive: Initially, all objects form one cluster, which is decomposed into smaller clusters. Finally, each object is in a cluster individually.

Linkage is the criterion by which the clustering algorithm determines distance between two clusters:

Single linkage: the distance between two clusters is their minimum distance: the distance between their two closest objects. Single linkage may cause the chaining problem, which forces clusters together due to single objects being close to each other.

Complete linkage: the distance between two clusters is their maximum distance. Complete linkage is useful if objects are far in high-dimensional space. Complete linkage is unsuitable for highly noisy datasets, since outliers are given more weight in the cluster decision.

Average linkage: takes the mean distance between all pairs of objects of two clusters. Average linkage is more computationally expensive than the methods above. Average linkage is the most popular of the three, avoiding the chaining problem and without giving special weight to outliers.

Hierarchical methods are popular in bioinformatics since clusters can be navigated at various levels of granularity [15, 87, 118, 119]. Eisen *et al.* used *average linkage* for clustering genes by expression pattern similarity, as assessed by Euclidean distance. For N genes, an $N \times N$ matrix is computed containing gene pair similarities, and then scanned to find the most similar genes. The clustering is illustrated by appending a tree reflecting gene relations, with branch lengths reflecting gene similarity. Hierarchical methods are useful for representing protein sequence family relationships [67, 68, 119]. Regulatory and interactome network clustering methods often involve finding cliques [120, 121]. Visualization of hierarchical clusters of cliques is called 'power graphs' [96].

Hierarchical methods are often slow. Errors in merging clusters cannot be undone and will affect the result. If large clusters are merged then interesting local cluster structure may be lost. Next, we discuss hierarchical clustering of numerical and then discrete data.

Numerical

BIRCH

BIRCH is suitable for large databases, improving the slow runtimes of other hierarchical algorithms. BIRCH is scalable with $O(N)$ complexity. BIRCH is based on clustering features (CF), and an in-memory data structure, called CF-tree. A CF-tree is a multilevel summary of the data distribution. A nonleaf node in a CF tree contains summaries of the CFs of its children, preserving the structure of the data. Disadvantages include a difficulty in finding arbitrary shaped clusters, since it uses the notion of radius. BIRCH received the SIGMOD 10-Year Test-of-Time award [90].

CURE

CURE improves upon BIRCH by ability to discover clusters of arbitrary shapes. CURE is also more robust with respect to outliers [91]. These benefits are achieved by using several representative objects for a cluster. The representatives for each cluster are 'shrunk' or moved toward the cluster center by a user-specified shrinking factor. At each iteration, the two clusters with the closest pair of representative objects are merged. CURE is scalable to large datasets with a complexity of $O(N)$, since CURE requires one scan of the dataset. A drawback is the user-specified parameter values, the number of clusters and the shrinking factor.

Spectral clustering

Spectral clustering originated in graph partitioning [122, 123, 88, 148]. Spectral algorithms use the second largest eigenvalue of the Laplacian of the graph adjacency (pairwise similarity) matrix, to decide where to partition the matrix. The resulting clusters are re-partititoned to generate a cluster hierarchy [89]. Spectral clustering may be ineffective for producing more than two clusters. Efficient linear algebra software facilitate spectral clustering of large datasets.

Biclustering of gene expression data allows simultaneous clustering of the rows and columns of a matrix, where rows correspond to genes and columns to conditions [33, 124–127]. Biclustering also allows overlap between clusters [37, 128, 129].

Discrete

ROCK

ROCK is an agglomerative algorithm [92]. ROCK assumes a similarity measure between objects and defines a 'link' between two objects whose similarity exceeds a threshold. Initially, each object is assigned to a separate cluster. Then, clusters are merged repeatedly according to their closeness: the sum of the number of 'links' between all pairs of objects between two clusters. ROCK has cubic complexity in N, and is unsuitable for large datasets [4, 103].

Chameleon

Chameleon improves upon some drawbacks of CURE and ROCK. Chameleon considers the internal interconnectivity and closeness of the objects both between and within two clusters to be merged. Chameleon applies to all datatypes, if a similarity metric is specified [93]. Its complexity is $O(N^2)$.

LIMBO

LIMBO improves on the scalability of other hierarchical clustering algorithms. LIMBO builds on the Information Bottleneck (IB) framework for quantifying the relevant information preserved when clustering [94]. LIMBO uses the IB framework to define a distance measure. LIMBO handles large datasets, using a memory bound summary for the data.

GRID-BASED CLUSTERING

While hierarchical and partitioning clustering are common in bioinformatics, there exist other approaches such as grid-based methods, which start by forming a grid structure of cells from the input objects. Each object is classified in a cell of the grid. The clustering is performed on the resulting grid structure.

Numerical

STING

STING combines grid-based and hierarchical clustering for numerical datasets [106]. STING has complexity of $O(N)$. STING partitions the dataset into rectangular cells. A cell at a high level is partitioned to form a number of cells at the next lower level. Hierarchical levels of cells correspond to different levels of resolution. Each cell of each level stores count, mean, standard deviation, minimum and maximum and the type of distribution that the attribute value in the cell follows, such as normal, uniform etc. A drawback is that cell boundaries are not diagonal, such that objects could be separated by multiple attribute values.

DENSITY-BASED CLUSTERING

Density-based approaches use a local density criterion; clusters are subspaces in which the objects are dense and are separated by subspaces of low density. Density-based methods are useful in bioinformatics for finding the densest subspaces in interactome networks, typically involving cliques [19, 65]. Advantages of many density-based algorithms include time efficiency and ability to find clusters of arbitrary shapes. Some density-based algorithms take user-specified input parameters, but not the number of clusters k that changes the clustering. Some cannot identify clusters of varying densities. With some density-based algorithms the central subspace of a cluster cannot be distinguished from the rest of the cluster based on a higher density [4, 103, 130]. Some density-based approaches are also grid-based, since a histogram is constructed by partitioning the dataset into a number of non-overlapping regions.

Numerical

DBSCAN

DBSCAN regards clusters as dense regions of objects in space that are separated by regions of low density. For each object of a cluster, the neighborhood of a given radius (ε) has to contain at least a minimum number of objects ($MinPts$), where ε and $MinPts$ are input parameters. Every object not contained in any cluster is considered noise [98]. Its complexity is $O(NlogN)$ if a spatial index is used; otherwise, it is $O(N^2)$. Its main advantage is that it can discover clusters of arbitrary shapes. DBSCAN is resistant to noise and provides a means of filtering for noise if desired. Its main drawback is the user-specified parameter values. DBSCAN is not suitable for high-dimensional data; as dimensionality increases, so does the relative distance between objects making it harder to perform density analysis. Because of its parameters, DBSCAN does not respond well to varying densities, such as sequence databases with subspaces of protein families; this leads to OPTICS.

OPTICS

Both DBSCAN and OPTICS require parameters to be specified by the user that will affect the result. However, OPTICS considers that different clusters could require different values. OPTICS covers a spectrum of all different $\varepsilon' \leq \varepsilon$ OPTICS has the same complexity as DBSCAN, $O(NlogN)$ if a spatial index is used. DBSCAN and OPTICS have difficulty identifying clusters within clusters [4, 130]. OPTICS finds an ordering of the data that is consistent with DBSCAN [99]. For sequence clustering, OPTICS was extended into SEQOPTICS to support users choosing parameters [10].

DENCLUE

The main advantage of DENCLUE is ability to find arbitrary shaped clusters. DENCLUE differs from other density-based approaches in that it pins density to a point in the attribute space instead of an object [100]. Its complexity is $O(N)$. The drawback is that it has a large number of input parameters. The influence of each object within its neighborhood is modeled using an influence function. The density function is the sum of the influence functions of all objects. Clusters are determined by identifying local maxima of the overall density function.

WaveCluster

WaveCluster uses a wavelet to transform the original data and find dense regions in the transformed space [105]. A wavelet transform is useful because it can suppress weaker information, and thus being effective for noise removal. This results in two main benefits: with less information one can speed up the process, and it can detect clusters at varying levels of accuracy. Its complexity is $O(N)$. However, it is only applicable to low-dimensional datasets. Its input parameters are the number of grid cells for each dimension, the wavelet transform, and the number of wavelet applications.

CLIQUE

CLIQUE improves on the scalability of other methods as its complexity is *O*(*N*). CLIQUE is useful for clustering high-dimensional data and is insensitive to object ordering. User-specified parameters are the grid size and a global density threshold for clusters. CLIQUE partitions space into non-overlapping rectangular units and identifies the dense units; a unit is dense if the fraction of total objects contained in it exceeds the user-specified value [107]. CLIQUE considers only hyper-rectangular clusters and projections parallel to the axes [4, 130].

Discrete

HIERDENC: Hierarchical Density-based Clustering

A challenge involved in applying density-based clustering to discrete datasets is that the 'cube' of attribute values has no ordering defined. The HIERDENC algorithm for *hierarchical density-based clustering of discrete data* offers a probabilistic basis for designing faster density-based discrete clustering algorithms. The characteristics of HIERDENC include insensitivity to the order of object input, and ability to handle outliers [97].

MULIC: Multiple Layer Incremental Clustering

MULIC is a faster simplification of HIERDENC that focuses on the multi-layered structure of special datasets, such as interactomes [14, 97, 131, 132]. MULIC produces layered clusters, which has several differences from traditional hierarchical clustering. First, MULIC requires no user-specified parameters. MULIC clusters have a clear separation, not requiring a cut-off to get the clusters as in hierarchical clustering. MULIC does not merge clusters during clustering, not losing interesting local cluster structure; instead, any cluster mergings that may be desirable are done after objects' clustering has finished. MULIC results in layered (or nested) clusters of biomolecules, with each cluster corresponding to a collapsed edge (biclique). MULIC clusters had higher recall of known complexes than other methods, and can be visualized in the power graphs cytoscape tool (http://www.proteinclustering.com).

Projected (subspace) clustering

Projected clustering is motivated by high-dimensional datasets, where clusters exist only in specific attribute subsets [133]. Clusters are subspaces of high-dimensional datasets, determined by the subset of attributes most relevant to each cluster. The values at the relevant attributes are distributed around some specific values in the cluster, while objects of other clusters are less likely to have such values. The drawback is that clustering depends on user parameters for determining the relevant attributes of each cluster; such parameters are the number of clusters or the average number of dimensions for each cluster. Projected clustering may distinguish the center of a cluster based on higher density or the relevant attributes [107].

CACTUS

CACTUS uses a minimum size for the relevant attribute sets, and assumes that a cluster is identified by a unique set of attribute values that seldom occur in other clusters [101]. This assumption may be unnatural for clustering many real world datasets. CACTUS may return too many clusters [103]. CACTUS has difficulty finding clusters within clusters [4, 130].

STIRR

STIRR looks for relationships between all attribute values in a cluster [102]. Two sets of attribute values, one with positive and another with negative weights, define two clusters. STIRR is sensitive to object ordering and lacks a definite convergence. The notion of weights is non-intuitive and several parameters are user-specified. The final detected clusters are often incomplete [103].

CLICK

CLICK creates a graph representation; vertices are discrete values and an edge is a co-occurrence of values in an object. A cluster is a *k*-partite maximal clique such that most pairs of vertices are connected by an edge. CLICK may return too many clusters or too many outliers [103].

CLOPE

CLOPE uses a heuristic of increasing the height-to-width ratio of the cluster histogram [104]. CLOPE is fast and scalable to high-dimensional datasets. The accuracy of CLOPE's results may suffer.

MODEL-BASED CLUSTERING

Model-based clustering assumes that objects match a model, which is often a statistical distribution. Then, the process aims to cluster objects such that they match the distribution. The model may be user-specified as a parameter and the model may change during the process. In bioinformatics, model-based clustering methods integrate background knowledge into gene expression, interactomes, and sequences [134, 135]. Building models is an oversimplification; user assumptions may be false and then results will be inaccurate. Another disadvantage of model-based clustering (especially neural networks) is slow processing time on large datasets.

Numerical

Self-Organizing Maps

Self-Organizing Maps involve neural networks, which resemble processing that occurs in the brain. NNs and SOM clustering involve several layers of units that pass information between one another in a weighed manner. Several units compete for an object; the unit that is closest to the object becomes the winning unit. SOMs assume that the winning units will eventually learn the correct cluster structure. NNs can model nonlinear relationships between attributes, and can handle dependent attributes. However, NNs have a number of drawbacks. NNs can handle binary data, but discrete attributes are hard to handle. Classifying a result into multiple clusters is done by setting arbitrary value thresholds for discriminating clusters. NNs do not present an easily understandable model, being more of a 'black box' that delivers results without explaining how the results were derived.

Discrete

COBWEB

COBWEB is a *conceptual* clustering method. COBWEB creates a hierarchical clustering in the form of a classification tree. COBWEB integrates observations incrementally into an existing classification tree by classifying the observation along a path of best matching nodes. A benefit of COBWEB is that it can adjust the number of clusters in a partition, without the user specifying this input parameter. A drawback is that it may assume correlated attributes are independent.

A classification tree differs from a decision tree. In a COBWEB classification tree each node refers to a concept, and contains the probability of the concept and the probabilities of the attribute-value pairs, which apply to the objects classified under that node. This is unlike decision trees, which label branches rather than nodes and use logical rather than probabilistic descriptions. Sibling nodes at a classification tree level form a partition [108].

Mixed Discrete and Numerical

BILCOM Empirical Bayesian

Model-based methods for gene expression clustering, such as Bi-level clustering of Mixed Discrete and Numerical Biomedical Data (BILCOM) [109], often adopt an empirical Bayesian approach, with GO annotations as the prior. Model-based clustering can find arbitrary shaped gene expression clusters (Figure 1) by including background knowledge as GO annotations [16, 18, 47, 136–139, 149].

For protein sequence clustering, Brown *et al.* used mixture densities to estimate amino-acid preferences within known subfamily clusters, achieving scalability and accuracy [134].

AutoClass

AutoClass is a clustering algorithm for mixed datatypes, which uses a Bayesian method for determining the optimal classes based on prior distributions [110]. Advantages of AutoClass include that Bayesian theory is theoretically well-founded and empirically well-tested. AutoClass investigates different numbers of clusters, which are not user-specified. The output is a mixture of several likely answers. Drawbacks include that users have to specify the model spaces to be searched in and wrong models may produce wrong results. AutoClass can be slow.

AutoClass finds the most likely classifications of objects in clusters, given a prior distribution for each attribute, symbolizing prior beliefs of the user. It changes the classifications of objects in clusters and changes the means and variances of the attributes' distributions in each cluster, until they stabilize. As an example, let the evidence on an object be $X = \{\text{age} = 28,\ \text{blood-type} = \text{A},\ \text{weight} = 73\,\text{kg}\}$; *blood-type* is a discrete attribute that is modeled with a *Bernoulli* distribution, while *age* and *weight* are continuous attributes modeled with a *normal (Gaussian)* distribution. AutoClass could classify X in a cluster, based on attribute distributions in each of the clusters.

SVM Clustering

SVMs provide a method for supervised learning (Section 1). SVMs were also adapted for clustering. SVM-based clustering does not use prior knowledge of object classifications [111]. Initially, every object in the dataset is randomly labelled and a binary SVM classifier trained. Then the lowest confidence classifications, those objects with confidence factor values beyond some threshold, repeatedly have labels switched to the other class label. The SVM is re-trained after each re-labeling on the lowest confidence objects. The repetition of the above process improves the classification accuracy and limits local minima traps.

GRAPH-BASED CLUSTERING

Graph-based clustering methods have been applied to interactomes for complex prediction (Section 2) and to sequence networks. These methods are sensitive to user-specified parameter values, and often slow; exceptions are SEQOPTICS and MULIC, presented previously.

Molecular Complex Detection

MCODE is designed to detect subnetworks with many edges in an interactome [19]. MCODE is sensitive to network alterations, edge removal and addition, and threshold parameters. MCODE complex prediction had high correctness, but few complexes were retrieved.

Super Paramagnetic Clustering

SPC is similar to entropy-based COOLCAT; when temperature or entropy increases, the system becomes less stable and the clusters become smaller [70]. SPC is robust to edge removal, but sensitive to edge addition. SPC gave weak complex prediction results [65].

Restricted Neighborhood Search Clustering

RNSC is similar to ROCK and Chameleon, considering the number of edges within and between clusters [112]. RNSC is relatively robust to parameter values and edge addition, but sensitive to edge removal. It gives many mini-clusters of small sizes [65].

Markov Clustering

MCL for interactomes is similar to projected clustering, which often improves results on high-dimensional datasets. It simulates a flow, finding clusters as high-flow regions separated by no-flow boundaries [113]. MCL is robust to network alterations, both edge removal and addition. An overall comparison showed MCL's superiority for finding complexes [65].

Other sequence clustering

TribeMCL clusters sequences into families using BLAST similarity searches [113]. SPC was also applied to sequences, improving sequence clustering over TribeMCL [70]. CD-HIT removes redundant sequences, satisfying the point proportion admissibility requirement [12]. ProClust improves results, using transitivity to conclude homology between $A \leftrightarrow C$ based on homology between $A \leftrightarrow B$ and $B \leftrightarrow C$ [141]. The BAG algorithm uses graph theoretic properties to guide cluster splitting and reduce errors [142].

DISCUSSIONS AND FUTURE APPLICATIONS

Requirements and desirable features of biomedical clustering applications were defined. Though not all existing clustering algorithms have been applied to biomedical problems yet, one can use Table 1 and Figure 2 to find a method that matches an application.

For gene expression clustering, *k*-means, hierarchical clustering, and SOMs have been applied. As discussed in the Introduction, desired features are minimum user input, finding arbitrary shaped clusters, robustness to outliers, and mixed datatypes. Table 1 shows that algorithms CLARA, CLARANS, BIRCH, CURE, HIERDENC, STING, AutoClass are good matches. Suppose a biologist uses *k*-means for clustering gene expression data, but outliers are a problem [140]. Table 1 and Figure 2 show that s/he could consider *k*-medoids, CLARA, CLARANS as alternatives, which improve upon *k*-means supporting outlier detection. CLARANS further improves upon *k*-medoids by supporting arbitrary shaped clusters. Fuzzy *k*-means supports overlapping clusters of co-regulated genes. If hierarchical clustering is desired for visualization, BIRCH is suitable for outlier detection. CURE additionally supports arbitrary shaped clusters, but requires the number of clusters as input. The projected density-based approaches are known to improve quality on high-dimensional data, such as 1000 samples of gene expression, but take user-specified parameters.

For interactomes, previous algorithms gave reasonable complex prediction results [146]. As discussed in the Introduction, one desires minimum user-specified parameters, integration of background knowledge, such as GO annotations, and robustness to noise. As Table 1 shows, AutoClass uses background knowledge distributions but requires parameters. SVM clustering requires no user-specified parameters. COBWEB finds the probabilities of concepts in clusters, requiring no parameters. In noisy high-throughput interactomes, weighed edges reflect confidence of correctness; a version of MULIC clustering called MULICsoft considers such weights [132].

For sequences, hierarchical methods are suitable for navigation and visualization. As discussed in the Introduction, scalability and point-proportion admissibility are essential. As Table 1 shows, BIRCH, CURE and LIMBO are hierarchical, scalable and satisfy point-proportion admissibility. As future work, indexing methods are developed for huge emerging sequence databanks [144].

Finally, we plan to do a large-scale study to compare the performance of clustering algorithms on various bioinformatics tasks. This study will evaluate whether less common algorithms perform better than the common ones in practice. We will also address the problem of objectively comparing the algorithms' performance.

Key Points

- Desired clustering features for different biomedical applications include: speed, minimal parameters, robustness to noise and outliers, redundancy handling, and object order independence.
- Many clustering algorithms have not been applied yet to biomedicine, but their features match applications' requirements.
- Clustering algorithms are separated into approaches: partitioning, hierarchical, density-based, grid-based, model-based, graph-based, as well as numerical, discrete and mixed datatypes.
- Within an approach, inheritance relationships between clustering algorithms specify common features and improvements they make upon one another.
- Table 1 and Figure 2 summarize benefits and drawbacks of 40 algorithms that are applicable to biomedical data, and help with finding a suitable method for an application.

Acknowledgements

The authors thank Frank Dressel, Rainer Winnenburg and Matthias Reimann for helpful comments and feedback, and Christof Winter for using the R heatmap. The authors thank Weston.pace for the *k*-means example image.

FUNDING

EU Sealife project; Dresden-Exists; the Ontario Graduate Scholarship; the Natural Sciences and Engineering Research Council of Canada.

References

1. Berkhin P. *Survey of Clustering Data Mining Techniques*. Accrue Software, Inc. Technical Report. San Jose, CA. 2002.
2. Alizadeh A, Eisen M, Davis R, *et al*. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature* 2000;**403**:503–11.
3. Alon U, Barkai N, Notterman D, *et al*. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc Natl Acad Sci* 1999;**96**:6745–50.
4. Grambeier J, Rudolph A. Techniques of cluster algorithms in data mining. *Data Mining Knowl Discov* 2002;**6**:303–60.
5. Kleinberg J, Papadimitriou C, Raghavan P. Segmentation problems. In: *Proceedings of the ACM STOC'98*, pp. 473–82, 1998.
6. Xu R. Survey of clustering algorithms. *IEEE Trans. Neural Networks* 2005;**16**.
7. D'haeseleer P. How does gene expression clustering work? *Nature Biotechnol* 2005;**23**:1499–501.
8. Han J, Kamber M. *Data Mining: Concepts and Techniques*, 2nd edition, Morgan Kaufmann, 2006.
9. Ding C, He X, Zha H, *et al*. Adaptive dimension reduction for clustering high dimensional data. In: *Proceedings of ICDM 2002*, pp. 107–14, 2002.
10. Chen Y, Reilly K, Sprague A. SEQOPTICS: a protein sequence clustering system. *BMC Bioinformatics* 2006; **7**(Suppl. 4):S10.
11. Krause A, Stoye J, Vingron M. Large scale hierarchical clustering of protein sequences. *BMC Bioinformatics* 2005; **6**:15.
12. Li W, Godzik A. CD-HIT: A fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics* 2006;**22**:1658–9.
13. Wu B. Cancer outlier differential gene expression detection. *Biostatistics* 2007;**8**:566–75.
14. Andreopoulos A, An A, Wang X, *et al*. Clustering by common friends finds locally significant proteins mediating modules. *Bioinformatics* 2007;**23**:1124–31.
15. Eisen M, Spellman P, Brown P, *et al*. Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci* 1998;**95**:14863–8.
16. Adryan B, Schuh R. Gene ontology-based clustering of gene expression data. *Bioinformatics* 2004;**20**:2851–2.
17. Gibbons F, Roth F. Judging the quality of gene expression-based clustering methods using gene annotation. *Genome Res* 2002;**12**:1574–81.
18. Pasquier C, Girardot F, Jevardat de Fombelle K, *et al*. THEA: Ontology driven analysis of microarray data. *Bioinformatics* 2004;**20**:2636–43.
19. Bader G, Hogue C. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics* 2003;**4**. Available at ftp://ftp.blueprint.org/pub/BIND/README
20. Gordon AD. *Classification*. Second Edition. London: Chapman and Hall CRC, 1999.
21. Balasubramaniyan R, Huellermeier E, Weskamp N, *et al*. Clustering of gene expression data using a local shape-based similarity measure. *Bioinformatics* 2005;**21**:1069–77.

22. Kim D, Lee K, Lee D. Detecting clusters of different geometrical shapes in microarray gene expression data. *Bioinformatics* 2005;**21**:1927–34.

23. Tamayo P, Slonim D, Mesirov J, *et al*. Interpreting patterns of gene expression with self-organizing maps (SOMs): Methods and applications to hematopoietic differentiation. *Proc Natl Acad Sci USA* 1999;**96**:2907–12.

24. Gat-Viks I, Sharan R, Shamir R. Scoring clustering solutions by their biological relevance. *Bioinformatics* 2003; **19**:2381–9.

25. Priness I, Maimon O, Ben-Gal I. Evaluation of gene-expression clustering by mutual information distance measures. *BMC Bioinformatics* 2007;**8**:111.

26. Yeung K, Haynor D, Ruzzo W. Validating clustering for gene expression data. *Bioinformatics* 2001;**17**:309–318.

27. McShane L, Radmacher M, Freidlin B, *et al*. Methods for assessing reproducibility of clustering patterns observed in analyses of microarray data. *Bioinformatics* 2002;**18**:1462–9.

28. Hubert L, Arabie P. Comparing partitions. *J Class* 1985; 193–218.

29. Datta S, Datta S. Comparisons and validation of statistical clustering techniques for microarray gene expression data. *Bioinformatics* 2003;**19**:459–66.

30. Datta S, Datta S. Methods for evaluating clustering algorithms for gene expression data using a reference set of functional classes. *BMC Bioinformatics* 2006;**7**:397.

31. Michaels G, Carr D, Fuhrman S, *et al*. Cluster analysis and data visualization of large-scale gene expression data. *Pac Symp Biocomput* 1998;**3**:42–53.

32. Ben-Dor A, Bruhn L, Friedman N, *et al*. Tissue classification with gene expression profiles. *J Comput Biol* 2000;**7**: 559–83.

33. Kluger Y, Basri R, Chang J, *et al*. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Res* 2003;**13**:703–16.

34. Ross-Macdonald P. Functional analysis of the yeast genome. *Funct. Integr. Genomics* 2000;**1**:99–113.

35. Shamir R, Sharan R. Algorithmic approaches to clustering gene expression data. In: Jiang T, Smith T, Xu Y, Zhang MQ (eds). *Current Topics in Computational Molecular Biology.* MIT Press, pp. 269–300, 2002.

36. Sharan R, Maron-Katz A, Shamir R. CLICK and EXPANDER: A system for clustering and visualizing gene expression data. *Bioinformatics* 2003;**19**:1787–99.

37. Tanay A, Sharan R, Shamir R. Discovering statistically significant biclusters in gene expression data. *Bioinformatics* 2002;**18**:S136–44.

38. Chopra P, Kang J, Yang J, *et al*. Microarray data mining using landmark gene-guided clustering. *BMC Bioinformatics* 2008;**9**:92.

39. Smith C, Finger J, Hayamizu T, *et al*. The mouse gene expression database (GXD). *Nucleic Acids Res* 2007;**35**: D618–23.

40. Spellman P, Sherlock G, Zhang M, *et al*. Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol Biol Cell* 1998;**9**:3273–97.

41. Tomancak P, Berman B, Beaton A, *et al*. Global analysis of patterns of gene expression during Drosophila embryogenesis. *Genome Biol* 2007;**8**:R145.

42. Gasch A, Spellman P, Kao C, *et al*. Genomic expression programs in the response of yeast cells to environmental changes. *Mol Biol Cell* 2000;**11**:4241–57.

43. Gasch A, Eisen M. Exploring the conditional coregulation of yeast gene expression through fuzzy *k*-means clustering. *Genome Biol* 2002;**3**:22.

44. Slonim D. From patterns to pathways: gene expression data analysis comes of age. *Nat Genet Suppl* 2002;**32**: 502–8.

45. Golub T, Slonim D, Tamayo P, *et al*. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* 1999;**286**:531–7.

46. Shi T, Seligson D, Belldegrun A, *et al*. Tumor classification by tissue microarray profiling: random forest clustering applied to renal cell carcinoma. *Mod Pathol* 2005;**18**:547–57.

47. Brown M, Grundy W, Lin D, *et al*. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc Natl Acad Sci* 2000;**97**:262–7.

48. Perou C, Sorlie T, Eisen M, *et al*. Molecular portraits of human breast tumours. *Nature* 2000;**406**:747–52.

49. Ross D, Scherf U, Eisen M. Systematic variation in gene expression patterns in human cancer cell lines. *Nat Genet* 2000;**24**:227–35.

50. Bolshakova N, Azuaje F, Cunningham P. An integrated tool for microarray data clustering and cluster validity assessment. *Bioinformatics* 2005;**21**:451–5.

51. Valentini G. Clusterv: A tool for assessing the reliability of clusters discovered in DNA microarray data. *Bioinformatics* 2006;**22**:369–70.

52. Tomlins S, Rhodes D, Yu J, *et al*. The role of SPINK1 in ETS rearrangement-negative prostate cancers. *Cancer Cell* 2008;**13**:519–28.

53. Gruetzmann R, Boriss H, Ammerpohl O, *et al*. Meta-analysis of microarray data on pancreatic cancer defines a set of commonly dysregulated genes. *Oncogene* 2005;**24**:5079–88.

54. Birnie R, Bryce S, Roome C, *et al*. Gene expression profiling of human prostate cancer stem cells reveals a pro-inflammatory phenotype and the importance of extracellular matrix interactions. *Genome Biol* 2008;**9**:R83.

55. Chua H, Sung W, Wong L. Exploiting indirect neighbours and topological weight to predict protein function from protein-protein interactions. *Bioinformatics* 2006;**22**: 1623–30.

56. Ito T, Chiba T, Ozawa R, *et al*. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proc Natl Acad Sci USA* 2001;**98**:4569–74.

57. Ito T, Tashiro K, Muta S, *et al*. Toward a protein-protein interaction map of the budding yeast: A comprehensive system to examine two-hybrid interactions in all possible combinations between the yeast proteins. *Proc Natl Acad Sci USA* 2000;**97**:1143–47.

58. Schwikowski B, Uetz P, Fields S. A network of protein-protein interactions in yeast. *Nat Biotechnol* 2000; **18**:1257–61.

59. Uetz P, Giot L, Cagney G, *et al*. A comprehensive analysis of protein-protein interactions in saccharomyces cerevisiae. *Nature* 2000;**403**:623–7.

60. Arnau V, Mars S, Marin I. Iterative cluster analysis of protein interaction data. *Bioinformatics* 2005;**21**:364–78.

61. Asur S, Ucar D, Parthasarathy S. An ensemble framework for clustering protein-protein interaction networks. *Bioinformatics* 2007;**23**:i29–40.

62. Brun C, Herrmann C, Guenoche A. Clustering proteins from interaction networks for the prediction of cellular functions. *BMC Bioinformatics* 2004;**5**:95.

63. Greene D, Cagney G, Krogan N, *et al*. Ensemble non-negative matrix factorization methods for clustering protein-protein interactions. *Bioinformatics* 2008;**24**: 1722–8.

64. Hanisch D, Zien A, Zimmer R, *et al*. Co-clustering of biological networks and gene expression data. *Bioinformatics* 2002;**18**(Suppl. 1):S145–54.

65. Brohue S, Van Helden J. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics* 2006;**7**:488.

66. Kawaji H, Takenaka Y, Matsuda H. Graph-based clustering for finding distant relationships in a large set of protein sequences. *Bioinformatics* 2004;**20**:243–52.

67. Bilu Y, Linial M. The advantage of functional prediction based on clustering of yeast genes and its correlation with non-sequence based classifications. *J Comput Biol* 2002;**9**: 193–210.

68. Kaplan N, Friedlich M, Fromer M, *et al*. A functional hierarchical organization of the protein sequence space. *BMC Bioinformatics* 2004;**5**:196.

69. Hazelhurst S, Hide W, Liptak Z, *et al*. An overview of the wcd EST Clustering Tool. *Bioinformatics* (Advance Access May 2008).

70. Tetko I, Facius A, Ruepp A, *et al*. Super paramagnetic clustering of protein sequences. *BMC Bioinformatics* 2005; **6**:82.

71. Torarinsson E, Havgaard J, Gorodkin J. Multiple structural alignment and clustering of RNA sequences. *Bioinformatics* 2007;**23**:926–32.

72. Cameron M, Bernstein Y, Williams H. Clustered sequence representation for fast homology search. *J Comput Biol* 2007; **14**:594–614.

73. Yang J, Wang W. CLUSEQ: Efficient and effective sequence clustering. In: *Proceedings of the 19th ICDE'03*, pp. 101–12, 2003.

74. Tong A, Lesage G, Bader G, *et al*. Global mapping of the yeast genetic interaction network. *Science* 2004;**303**:808–13.

75. Fields S, Song O. A novel genetic system to detect protein-protein interactions. *Nature* 1989;**340**:245–6.

76. MacQueen J. Some methods for classification and analysis of multivariate observations. Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1:281–297, 1967.

77. Hochbaum D, Shmoys D. A best possible heuristic for the *k*-center problem. *Math. Operat. Res.* 1985;**10**:180–4.

78. Kaufmann L, Rousseeuw P. Clustering by means of medoids. In: Dodge Y, (ed.). *Statistical Data Analysis Based on the L 1 Norm and Related Methods*. Elsevier Science, pp. 405–16, 1987.

79. Kaufmann L, Rousseeuw P. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.

80. Ng R, Han J. CLARANS: A method for clustering objects for spatial data mining. *IEEE Trans Knowl Data Eng* 2002;**14**.

81. Dembele D, Kastner P. Fuzzy C-means method for clustering microarray data. *Bioinformatics* 2003;**19**:973–80.

82. Huang Z. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining Knowl Disc* 1998;**2**:283–304.

83. Huang Z, Ng M. A fuzzy k-modes algorithm for clustering categorical data. *IEEE Trans Fuzzy Syst* 1999;**7**: 446–52.

84. He Z, Xu X, Deng S, *et al*. Squeezer: An efficient algorithm for clustering categorical data. *J Comput Sci Technol* 2002;**17**: 611–24.

85. Huang Z. Clustering large data sets with mixed numeric and categorical values. *Knowledge Discovery and Data Mining: Techniques and Applications*. World Scientific, 1997.

86. Barbara D, Li Y, Couto J. COOLCAT: an entropy-based algorithm for categorical clustering. In: *Proceedings of the CIKM'02*, pp. 582–9, 2002.

87. Eisen M, Brown P. DNA arrays for analysis of gene expression. *Methods Enzymol* 1999;**303**:179–205.

88. Ng A, Jordan M, Weiss Y. On spectral clustering: Analysis and an algorithm. In: *Advances in Neural Information Processing Systems*, Vol. 14: Proceedings of the 2001.

89. White S, Smyth P. A spectral clustering approach to finding communities in graphs. In: *Proceedings of the SDM'05*, 2005.

90. Zhang T, Ramakrishnan R, Livny M. BIRCH: An efficient data clustering method for very large databases. In: *Proceedings of the ACM SIGMOD'96*, Montreal, Canada, 1996.

91. Guha S, Rajeev R, Kyuseok S. CURE: An efficient clustering algorithm for large databases. In: *Proceedings of the ACM SIGMOD'98*. Seattle, USA, pp. 73–84, 1998.

92. Guha S, Rastogi R, Shim K. ROCK: A robust clustering algorithm for categorical attributes. *Information Syst* 2000;**25**: 345–66.

93. Karypis G, Han E, Kumar V. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. *IEEE Comput Special Issue Data Anal Mining* 1999;**32**:68–75.

94. Andritsos P, Tsaparas P, Miller R, *et al*. LIMBO: Scalable clustering of categorical data. In: *Proceedings of the 9th International Conference on Extending Database Technology EDBT'04*, Heraklion, Greece. March 14–18, 2004.

95. Karypis G, Aggarwal R, Kumar V, *et al*. Multilevel hypergraph partitioning: Application in VLSI domain. In: *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 526–9, 1997.

96. Royer L, Reimann M, Andreopoulos B, *et al*. Unravelling protein networks with power graph analysis. *PLoS Comput Biol* 2008;**4**. Available at http://www.biotec.tu-dresden.de/schroeder/group/powergraphs

97. Andreopoulos B, An A, Wang X. Hierarchical density-based clustering of categorical data and a simplification. In: *Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2007)*, Springer LNCS 4426/2007, pp. 11–22, Nanjing, China, May 22–25, 2007.

98. Sander J, Ester M, Kriegel H, *et al*. Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications. *Data Mining Knowl Disc* 1998;**2**:169–94.

99. Ankerst M, Breunig M, Kriegel H, *et al*. OPTICS: Ordering points to identify the clustering structure. In: *Proceedings of the ACM SIGMOD'99 Int. Conf. on Management of Data*. Philadelphia, USA, pp. 49–60, 1999.

100. Hinneburg A, Keim D. An efficient approach to clustering in large multimedia databases with noise. In: *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining KDD'98*, pp. 58–65, 1998.

101. Ganti V, Gehrke J, Ramakrishnan R. CACTUS-clustering categorical data using summaries. In: *Proceedings of the KDD'99*. San Diego, CA, USA, pp. 73–83, 1999.

102. Gibson D, Kleiberg J, Raghavan P. Clustering categorical data: An approach based on dynamical systems. In: *Proceedings of the 24th International Conference on Very Large Databases (VLDB'98)*, New York City, USA. pp. 311–23. August 24–27, 1998.

103. Zaki M, Peters M. CLICKS: Mining subspace clusters in categorical data via K-partite maximal cliques. In: *Proceedings of the 21st International Conference on Data Engineering ICDE'05*, Tokyo, Japan, pp. 355–6, 2005.

104. Yang Y, Guan S, You J. CLOPE: A fast and effective clustering algorithm for transactional data. In: *Proceedings of KDD'02*, pp. 682–7, 2002.

105. Sheikholeslami G, Chatterjee S, Zhang A. WaveCluster: A multi-resolution clustering approach for very large spatial databases. In: *Proceedings of the 24th International Conference on Very Large Data Bases*,, 1998.

106. Wang W, Yang J, Muntz R. STING: A statistical information grid approach to spatial data mining. In: *Proceedings of the 23rd International Conference on Very Large Data Bases VLDB'97*, pp. 186–195, 1997.

107. Agrawal R, Gehrke J, Gunopulos D, *et al*. Automatic subspace clustering of high dimensional data for data mining applications (1998). In: *Proceedings of the ACM-SIGMOD'98 Int. Conf. Management of Data*. Seattle, Washington, June 1998, pp. 94–105.

108. Fisher D. Knowledge acquisition via incremental conceptual clustering. *Machine Learning* 1987;**2**:139–72.

109. Andreopoulos B, An A, Wang X. Bi-Level Clustering of mixed categorical and numerical biomedical data. *Int J Data Mining Bioinformatics* 2006;**1**:19–56.

110. Stutz J, Cheeseman P. Bayesian classification (autoclass): Theory and results. *Advances in Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, CA, pp. 153–80, 1995.

111. Winters-Hilt S, Merat S. SVM clustering. *BMC Bioinformatics* 2007;**8**:S18.

112. King A, Przulj N, Jurisica I. Protein complex prediction via cost-based clustering. *Bioinformatics* 2004;**20**: 340–8.

113. Enright A, Dongen S, Ouzounis C. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res* 2002;**30**:1575–84.

114. Kraj P, Sharma A, Garge N, *et al*. ParaKMeans: Implementation of a parallelized K-means algorithm suitable for general laboratory use. *BMC Bioinformatics* 2008;**9**:200.

115. Belacel N, Cuperlovic-Culf M, Laflamme M, *et al*. Fuzzy J-means and VNS methods for clustering genes from microarray data. *Bioinformatics* 2004;**20**:1690–701.

116. Langfelder P, Zhang B, Horvath S. Defining clusters from a hierarchical cluster tree: The Dynamic Tree Cut package for R. *Bioinformatics* 2008;**24**:719–20.

117. Mojena R. Hierarchical grouping methods and stopped rules: An evaluation. *Comput J* 1977;**20**:359–63.

118. Baehrecke E, Dang N, Babaria K, *et al*. Visualization and analysis of microarray and gene ontology data with treemaps. *BMC Bioinformatics* 2004;**5**. Available at http://www.cs.umd.edu/hcil/treemap-history/

119. Loewenstein Y, Elon P, Fromer M, *et al*. Efficient algorithms for accurate hierarchical clustering of huge datasets: Tackling the entire protein space. *Bioinformatics* 2008;**24**:i41–9.

120. Grochow J, Kellis M. Network motif discovery using subgraph enumeration and symmetry-breaking. *Res Comput Mol Biol* 2007;92–106.

121. Zhang Y, Xuan J, Benildo G de los Reyes, *et al*. Network motif-based identification of transcription factor-target gene relationships by integrating multi-source biological data. *BMC Bioinformatics* 2008;9.

122. Azar Y, Fiat A, Karlin A, *et al*. Spectral analysis of data. In: *Proceedings of STOC'01*, Crete, Greece, July 6–8, 2001.

123. Kannan R, Vampala S, Vetta A. On clusterings: Good, bad and spectral. In: *Proceedings of the Foundations of Computer Science'00*, pp. 367–378, 2000.

124. Barkow S, Bleuler S, Prelic A, *et al*. BicAT: A biclustering analysis toolbox. *Bioinformatics* 2006;**22**:1282–3.

125. Dhillon I. Co-clustering documents and words using bipartite spectral graph partitioning. In: *Proceedings of the ACM KDD*, 2001.

126. Madeira S, Oliveira A. Biclustering algorithms for biological data analysis: A survey. *IEEE Trans Comput Biol Bioinformatics* 2004;**1**:24–45.

127. Pensa R, Robardet C, Boulicaut J. A bi-clustering framework for categorical data. In: *Proceedings of the KDD'05*, Vol. 3721, pp. 643–50, 2005.

128. Cheng Y, Church G. Biclustering of expression data. In: *Proceedings of the ISMB*, pp. 93–103, 2000.

129. Reiss D, Baliga N, Bonneau R. Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks. *BMC Bioinformatics* 2006;**2**: 280–302.

130. Gionis A, Hinneburg A, Papadimitriou S, *et al*. Dimension induced clustering. In: *Proceedings of the KDD'05*, 2005.

131. Andreopoulos B, An A, Wang X. Finding molecular complexes through multiple layer clustering of protein interaction networks. *Int J Bioinformatics Res Appl* 2007;**3**: 65–85.

132. Andreopoulos B, An A, Tzerpos V, *et al*. Clustering large software systems at multiple layers. *Elsevier Information Softw Technol* 2007;**49**:244–54.

133. Aggarwal C, Han J, Wang J, *et al*. A framework for projected clustering of high dimensional data streams. In: *Proceedings of the 30th VLDB'04*, Toronto, Canada, 2004.

134. Brown D. Efficient functional clustering of protein sequences using the Dirichlet process. *Bioinformatics* 2008; **24**:1765–71.

135. Qi Y, Balem F, Faloutsos C, *et al*. Protein complex identification by supervised graph local clustering. In: *Proceedings of the ISMB 2008*, Toronto, Canada.

136. Huang D, Pan W. Incorporating biological knowledge into distance-based clustering analysis of microarray gene expression data. *Bioinformatics* 2006;**22**:1259–68.

137. Joo Y, Booth J, Namkoong Y, *et al*. Model-based Bayesian clustering (MBBC). *Bioinformatics* 2008;**24**:874–5.

138. Joshi A, Van de Peer Y, Michoel T. Analysis of a Gibbs sampler method for model-based clustering of gene expression data. *Bioinformatics* 2008;**24**:176–83.

139. Schachtner R, Lutter D, Knollmueller P, *et al*. Knowledge-based gene expression classification via matrix factorization. *Bioinformatics* (Advance Access June 2008).

140. Lian H. MOST: detecting cancer differential gene expression. *Biostatistics* 2008;**9**:411–8.

141. Pipenbacher P, Schliep A, Schneckener S, *et al*. ProClust: Improved clustering of protein sequences with an extended graph-based approach. *Bioinformatics* 2002;**18**:182–91.

142. Kim S, Lee J. BAG: a graph theoretic sequence clustering algorithm. *International Journal of Data Mining and Bioinformatics* 2006;**1**:178–200.

143. Wang H, Wang W, Yang J, *et al*. Clustering by pattern similarity in large datasets. In: *Proceedings of the SIGMOD'02 Conference*, 2002.

144. Morgulis A, Coulouris G, Raytselis Y, *et al*. Database indexing for production MegaBLAST searches. *Bioinformatics* 2008;**24**:1757–64.

145. Hartigan JA. *Clustering Algorithms*. New York: Wiley, 1975.

146. Dittrich M, Klau G, Rosenwald A, *et al*. Identifying functional modules in protein-protein interaction networks: an integrated exact approach. *Bioinformatics* 2008;**24**:i223–i231.

147. Murtagh F. Multidimensional clustering algorithms. In: *COMPSTAT Lectures 4*, Physica-Verlag, Wuerzburg, 1985.

148. Higham D, Kalna G, Kibble M. Spectral clustering and its use in bioinformatics. *Journal of Computational and Applied Mathematics* 2007;**204**:25–37.

149. Bellazzi R, Zupan B. Towards knowledge-based gene expression data mining. *J Biomed Inform, Intelligent Data Analysis in Biomedicine* 2007;**40**:787–802.