Combinatorial Optimization Techniques for Network-Based Data Mining

Oleg Shirokikh, Vladimir Stozhkov and Vladimir Boginski

Contents

O. Shirokikh • V. Stozhkov
Industrial and Systems Engineering Department, University of Florida, Gainesville, FL, USA
e-mail: olegshirokikh@ufl.edu; vstozhkov@ufl.edu

V. Boginski
Industrial and Systems Engineering Department, University of Florida, Shalimar, FL, USA
e-mail: boginski@reef.ufl.edu

P.M. Pardalos et al. (eds.), *Handbook of Combinatorial Optimization*,
DOI 10.1007/978-1-4419-7997-1_6, © Springer Science+Business Media New York 2013

Abstract

The matters of discussion are combinatorial optimization aspects, concepts, and applications arising in the broad area of network-based data mining. The approach of representing real-world datasets as large-scale networks (graphs) has become increasingly popular during recent years. The purpose of this chapter is to briefly review the graph-theoretic and combinatorial optimization concepts that are important in the context of data mining, as well as to discuss the interpretation of these concepts from mathematical modeling perspective.

1 Introduction

The process of studying real-life complex systems often deals with large datasets arising in diverse applications including telecommunications, biotechnology, medicine, finance, physics, ecology, and geographical information systems [2, 21]. Understanding the structural properties of a certain dataset is in many cases the task of the crucial importance. To extract useful information from data, one often needs to apply advanced mathematical modeling techniques to summarize, visualize, and process the information contained in a dataset. *Data mining* problems deal with multiple aspects of information retrieval from datasets, and a variety of techniques has been developed to address different types of tasks arising in this area. Data mining is a part of *knowledge discovery in databases (KDD)* [81]. This is the process of converting data into useful information, which includes several main steps summarized in Fig. 1. The purpose of preprocessing is to transform input data to a proper format for further analysis. An example of postprocessing is visualization, which allows to research and explore data and data mining results from different viewpoints. One can also apply statistical measures and hypothesis testing techniques at the postprocessing step to potentially eliminate inconsistent data mining results. However, data mining itself is a crucial step in the KDD process. The purpose of this chapter is to review possible methods of utilizing combinatorial optimization and network-based tools to extract information from real-world datasets.

During the past several years, *optimization-based approaches* have received increased attention in the context of data mining problems [27]. Moreover, along with classical data mining techniques (e.g., support vector machines, artificial neural networks, decision trees, K-means/K-median clustering, and fuzzy C-means clustering), many recent studies deal with *network (graph)-based representations* of large real-world datasets and utilize graph-theoretic and network optimization concepts to extract useful and nontrivial information from these datasets. In this approach, a dataset is represented as a *graph* (*network*) with certain attributes associated with its vertices and edges. As it will be discussed later, combinatorial optimization problems often arise in this analysis, and solving these problems is an integral part of network-based data mining.

Studying the structure of a graph representing a dataset is often important for understanding the internal properties of the application it represents, as well as for

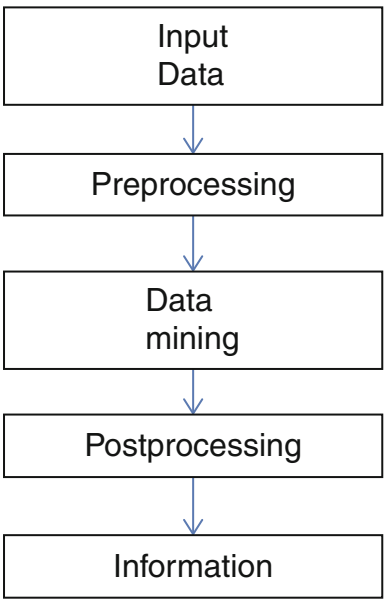

Fig. 1 Knowledge discovery in databases

improving storage organization and information retrieval. One can visualize a graph as a set of dots and links connecting them, which often makes this representation convenient and easily understandable.

The main concepts of graph theory were founded several centuries ago, and many network optimization algorithms have been developed since then. However, the trend of applying graph models to represent various real-life massive datasets has started relatively recently. Graph theory and related operations research techniques are quickly becoming a field with a strong practical impact. The expansion of graph-theoretical approaches in various applications gave birth to the terms "graph practice" and "graph engineering" [47]. Graph (network)-based data mining techniques have gained significant popularity in the recent years. Various aspects of this emerging field are addressed in [32, 84]. The term "network science" referring to a broad variety of network-based modeling approaches has also been widely used recently. In [10] potential relevance between "network science" and operations research is discussed. Although potential applications of operations research in this broad field can be diverse, network-based data mining is among the areas where optimization-based techniques can provide valuable contributions.

Network-based models allow one to extract information from real-world datasets using various standard concepts from graph theory. In many cases, one can investigate specific properties of a dataset by detecting special formations in the corresponding graph, for instance, *connected components*, *spanning trees*, *cliques*, and *independent sets*. In particular, connected components, cliques, and independent sets can be used to address the important *clustering* problems arising in many data mining applications. This problem essentially represents partitioning the set of elements of a certain dataset into a number of subsets (clusters) of objects according

to some *similarity* (or *dissimilarity*) criterion. These concepts are associated with a number of network optimization problems that will be discussed below.

Another aspect of investigating network models of real-world datasets is studying the global organization and structural properties of these networks, which are reflected by *degree distributions* of the constructed graphs. The degree distribution represents the large-scale pattern of connections in the graph, which reflects the global properties of the dataset. One of the important results discovered during the last several years is the observation that many graphs representing the datasets from diverse areas (Internet, telecommunications, biology, sociology) obey the *power-law* model [5]. The fact that graphs representing completely different datasets have a similar well-defined power-law structure has been widely reflected in the literature [4, 8, 16, 17, 21, 31, 47]. It indicates that global organization and evolution of datasets arising in various spheres of life follow similar laws and patterns. This fact served as a motivation to introduce a concept of "self-organized networks." The information about the degree distribution of a graph that represents a certain dataset can be significant in the data mining context, since it may allow one to analytically predict the size of certain types of clusters in this graph (e.g., connected components, cliques, and other structures). In particular, the size of connected components (clusters) in very large power-law graphs has been studied in [5].

Further in this chapter, the main issues and concepts arising in network-based data mining will be addressed: specifically, the process of representing a dataset as a graph using appropriate similarity (proximity) measures, which serves as a basis for applying graph-theoretic and network optimization techniques. These concepts and techniques are discussed later in this chapter. Finally, there will be given several examples of popular recent application areas of network-based data mining.

2 Similarity Measures Used in Graph-Based Data Mining

As indicated above, the essence of network-based data mining is representing a dataset under consideration as a network (graph) where vertices and edges are constructed according to certain principles that may depend on the structure and the origin of this dataset. The vertices of the graph usually represent data records (objects), whereas the edges that connect the vertices are constructed using an appropriate *similarity criterion*. That is, two data objects (vertices) are connected if they are "similar enough" to each other. However, there are a lot of different ways to quantify the degree of similarity. In this section, possible types of *similarity measures* (also referred to as *proximity measures*) will be discussed. More precisely, those are proximity measures between data objects that can be used for constructing the corresponding graph representations of datasets.

Usually, the *similarity* between two objects is a numerical measure of the degree to which the two objects are alike. It can often be scaled between 0 (no similarity) and 1 (complete similarity). The *dissimilarity* between two objects is a numerical measure of the degree to which the two objects are different. Also, the term *distance* is used to measure the degree of similarity/dissimilarity. Dissimilarities sometimes occupy the interval $[0, 1]$, but it is usual for them to belong to the range $[0, \infty)$.

Transformations are used to convert similarity to dissimilarity and vice versa. First of all, proximity measures should be converted to the interval $[0, 1]$. For instance, if the grade has a range from 1 to 5, it makes sense to use the formula $\hat{s} = (s - 1)/4$ to replace the initial proximity measure s by standard $\hat{s}$ that belongs to the interval $[0, 1]$. Generally, the transformations of similarities to the interval $[0, 1]$ are given by the formula

$$\hat{s} = \frac{s - \min s}{\max s - \min s}.$$

Analogously, dissimilarity measures with a finite range can be mapped to the interval $[0, 1]$ by transformation

$$\hat{d} = \frac{d - \min d}{\max d - \min d}.$$

In other cases, when values of the original proximity measure fill the interval $[0, \infty)$, a nonlinear conversion is necessary. As an example, the formula $\hat{d} = \frac{d}{1+d}$ can be utilized. The dissimilarities $0, \frac{1}{3}, \frac{2}{3}, 1, 5, 10, 100, 1{,}000$ will be converted to the new dissimilarities 0, 0.25, 0.4, 0.5, 0.833, 0.909, 0.99, 0.999. Transforming dissimilarities into similarities and vice versa is rather straightforward too. If the similarity falls in the interval $[0, 1]$, then the dissimilarity can be defined by the formula $d = 1 - s$. For $d \in [0, \infty)$, various nonlinear transformations can be employed. Consistent examples are $s = 1/(d + 1)$ and $s = e^{-d}$.

In the standard setup of many data mining problems, one deals with a dataset of N data records that can be represented by vectors $x^i = (x_1^i, x_2^i, \ldots, x_n^i)$, $i = 1, \ldots, N$, where every vector x^i has n components that are referred to as the *features*, or *attributes*, of the data record. The similarity/dissimilarity measure between any two data records would be defined by the corresponding values of their attributes. The following similarity/dissimilarity measures are commonly used in practice.

- *Distance metrics*: Generally it can be expressed in the form of Minkowski distance metric

$$d(x, y) = \left(\sum_{k=1}^{n} |x_k - y_k|^r \right)^{1/r},$$

  where r is a parameter that defines different types of metrics. The most commonly used distance metrics are L_1 norm for $r = 1$ (*Manhattan distance*, *city block distance*), L_2 norm for $r = 2$ (*Euclidean distance*), and L_∞ norm for $r = \infty$ (*supremum distance*) that signifies

$$d(x, y)_{L_\infty} = \max_{i=1,\ldots,n} |x_k - y_k|.$$

  A common particular instance of Manhattan distance is the *Hamming distance*, which is the number of bits that are different between two objects that have only binary attributes (i.e., between two binary vectors). In Fig. 2, the difference

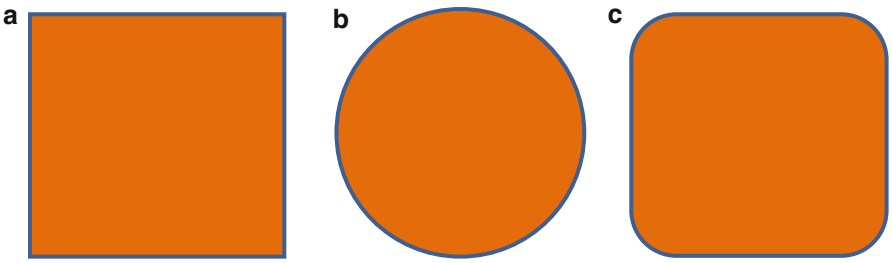

Fig. 2 Circumferences in different L_p norms. (**a**) Standard circle in L_1 norm. (**b**) Standard circle in L_2 norm. (**c**) Standard circle in L_4 norm

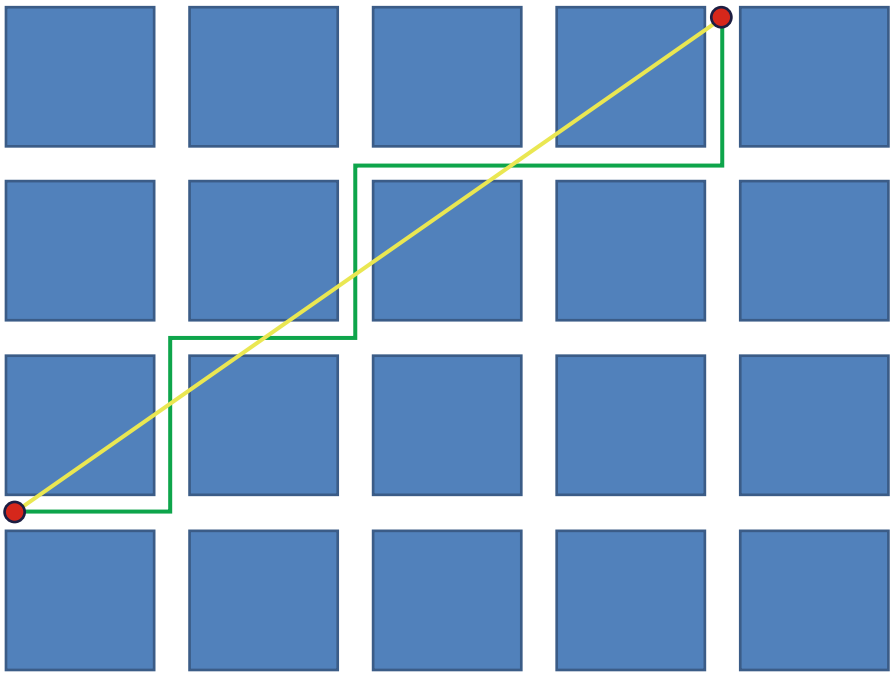

Fig. 3 Comparison of city block distance and Euclidean distance

between L_1, L_2, and L_4 can be observed graphically on the example of the standard circle presented in all these metrics. The norms L_1 and L_2 are compared visually in Fig. 3.

Supremum ($L_{\max}$ or L_∞ norm) distance is the maximum difference between any attribute of two objects. More rigorously, L_∞ is defined by the limit

$$d(x, y) = \lim_{r \to \infty} \left(\sum_{k=1}^{n} |x_k - y_k|^r \right)^{1/r}.$$

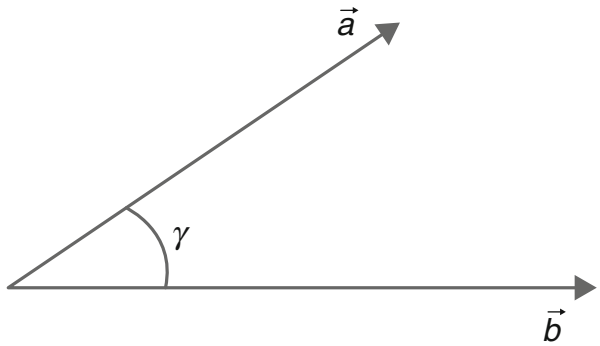

Fig. 4 Geometric representation of cosine similarity

To add more flexibility in the situations when the values of some attributes have substantially different importance levels or orders of magnitude, the formulas of proximity can be modified by weighting the contribution of each attribute. In the case of the distance metrics, the Minkowski distance can be modified as follows:

$$d(x, y) = \left(\sum_{k=1}^{n} \omega_k |x_k - y_k|^r \right)^{1/r},$$

where ω_k is the relative weight of the kth component in the considered vector space.

- *Cosine similarity*: This measure is often used to characterize text document similarity (e.g., documents represented as vectors with attributes denoting the frequency of each word in the considered document) and can be expressed as

$$\cos(a, b) = \frac{a \cdot b}{||a||||b||},$$

where $a \cdot b$ is a scalar product of vectors a and b. Even though documents have thousands of attributes, each document is sparse because there is a few nonzero attributes. Thus, similarity should not depend on the number of shared zero values. Therefore, a proximity measure for documents must ignore 0–0 matches like *Jaccard measure*. The cosine similarity is one of the useful measures in this case (Fig. 4).
- *Correlation*: The standard correlation measure is commonly used to characterize the similarity between different types of data, for instance, time series data, where the attributes represent the values of a certain parameter for different time moments. A well-known example that the correlation measure can be effectively used is the analysis of stock market data, where high correlation would mean high degree of similarity between a given pair of stocks.

The *general correlation coefficient* is given by the formula

$$\overline{r} = \frac{\sum_{i<j} c_{ij}(x)c_{ij}(y)}{\sqrt{\sum_{i<j} c_{ij}^2(x) \cdot \sum_{i<j} c_{ij}^2(y)}},$$

where x and y are two samples of comparable data and c_{ij} is some function measuring relative distance between x_i and x_j (ith and jth coordinates of the sample vector x). Three particular examples of the general correlation coefficient are given below.

Pearson's correlation coefficient is the most widely used correlation measure. It is defined by the formula

$$\text{corr}(x, y) = \frac{\text{covariance}(x, y)}{\text{standard deviation}(x) * \text{standard deviation}(y)},$$

where from statistics theory it is known that

$$\text{covariance}(x, y) = \sum_{k=1}^{n} (x_k - \overline{x})(y_k - \overline{y}),$$

$$\text{standard deviation}(x) = \sqrt{\frac{1}{n-1} \sum_{k=1}^{n} (x_k - \overline{x})^2},$$

$$\text{standard deviation}(y) = \sqrt{\frac{1}{n-1} \sum_{k=1}^{n} (y_k - \overline{y})^2}.$$

Using standard notations, $\overline{x}$ and $\overline{y}$ are means of two samples x and y, respectively. Notice that Pearson's correlation coefficient could be easily derived from the general correlation coefficient if it is assigned that $c_{ij} = x_j - x_i$. Also, Spearman's rank correlation coefficient [79] and Kendall's *tau* (τ) rank correlation coefficient [55] are frequently utilized for peculiar statistical tests and hypotheses.

If it is assigned that $c_{ij} = R_j - R_i$ where R_k is the rank of x_k in the given sample vector $x = (x_1, \ldots, x_n)$ for each $k = 1, \ldots, n$, then the formula below for *Spearman's rank correlation coefficient* can be derived:

$$\rho_S = \frac{\sum_{i=1}^{n} (R_i - \overline{R})(S_i - \overline{S})}{\sqrt{\sum_{i=1}^{n} (R_i - \overline{R})^2 \sum_{i=1}^{n} (S_i - \overline{S})^2}},$$

where R_k and S_k are the ranks of x_k and y_k in the samples x and y, correspondingly. Also, it should be pointed out that always $\overline{R} = \overline{S} = (n+1)/2$.

Having assigned $c_{ij} = \text{sgn}(x_j - x_i) = \text{sgn}(R_j - R_i)$, the formula for *Kendall's tau (τ) rank correlation coefficient* is obtained:

$$\tau = \sum_{i<j} \text{sgn}(R_i - \overline{R}) \cdot \text{sgn}(S_i - \overline{S}),$$

where R_k and S_k are ranks of the given samples as in the previous notations.

Regarding binary data, other definitions of similarity (proximity) measures are also used in practice [81]. For instance, simple matching coefficient, Jaccard coefficient, and extended Jaccard coefficient can be efficiently used to measure the similarity between data objects of aforementioned kind. Let x and y be two objects that consist of n binary attributes. Comparing of binary data x and y induces the following qualities:

f_{00} = the number of attributes where $x_i = 0$ and $y_i = 0$
f_{01} = the number of attributes where $x_i = 0$ and $y_i = 1$
f_{10} = the number of attributes where $x_i = 1$ and $y_i = 0$
f_{11} = the number of attributes where $x_i = 1$ and $y_i = 1$.

Simple matching coefficient (SMC) is commonly used similarity measure for binary data. It does not pay attention to elimination of 0–0 matches from consideration. SMC reflects both presences and absences equally. For instance, it can be utilized to find respondents who had answered questions on a test similarly.

$$SMC = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}}$$

Jaccard coefficient can be employed for sparse data. The idea is to compare nonzero elements belonging to two samples. Consequently, it makes little sense to use SMC coefficient in such a case because samples will be always similar because of the large number of zeros. In this case, 0–0 matches must not be counted. That is why under such conditions, Jaccard coefficient is more reasonable than simple matching coefficient.

$$J = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$$

Extended Jaccard coefficient is an expansion of Jaccard coefficient to real data. In the binary case, it is equal to a regular Jaccard coefficient. The extended Jaccard coefficient can be used for document data and also known as Tanimoto coefficient.

$$EJ(x, y) = \frac{x \cdot y}{||x||^2 + ||y||^2 - x \cdot y}$$

A significant issue related to distance measures is how to handle the situation when attributes do not have the same range of values. Often there is some correlation between some of attributes. In this case, a generalization of Euclidean distance referred to as *Mahalanobis distance* can be applied to measure similarity.

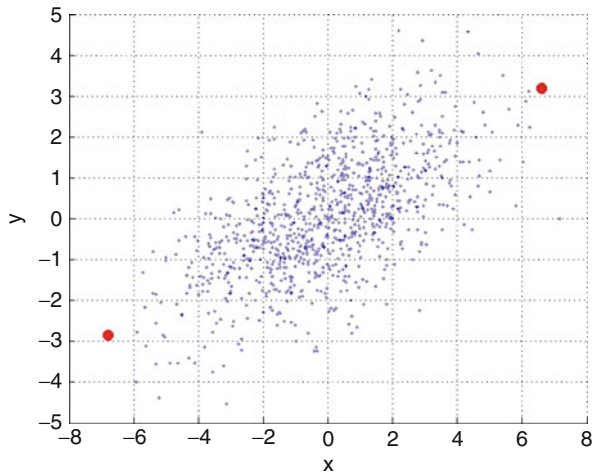

Fig. 5 Mahalanobis distance versus Euclidean distance [81]. The Mahalanobis distance between these two points are represented by *large dots* is 6. The corresponding Euclidean distance is 14.7. The correlation between axes causing distortion of data is equal to 0.6

The Mahalanobis distance between two objects x and y is defined as follows:

$$\text{mahalanobis}(x, y) = (x - y)\Sigma^{-1}(x - y)^T,$$

where Σ is the covariance matrix of the given data that reflects dependence between some attributes. The good example [81] is provided in Fig. 5. In practice, computing the Mahalanobis distance is worth doing it for correlated points in spite of its expensive cost.

Moreover, besides defining the similarity between data objects, in some cases, it is also needed to quantify the *dissimilarity*. The aforementioned proximity measures can be also used to define the pairwise dissimilarity. Therefore, all further discussion is applicable to clustering applications based on either similarity or dissimilarity. Without loss of generality, the term "similarity" will mostly be used throughout this chapter.

Determining an appropriate similarity measure is often a nontrivial task. However, if a suitable similarity measure is determined for a given dataset, this gives one the information about pairwise proximities for all pairs of data objects. Based on this information, one can easily construct the $N \times N$ proximity matrix and then use the proximity matrix to create a graph with N vertices and $N(N-1)/2$ weighted edges, where the weights $w_{ij} = d(x^i, x^j)$ are equal to the corresponding proximity measures between data objects x^i and x^j.

Note that this simple procedure would produce a complete weighted graph (with all possible edges), which can be challenging to analyze from both theoretical and computational perspectives. To reduce the edge density of the obtained graph, one can use an appropriate *sparsification* procedure that can substantially reduce the number of edges in the graph while still preserving valuable information about the dataset. The following simple principles can be used to achieve this:

- *k-nearest neighbor approach*: For every vertex i, keep only the edges that connect it to its k-nearest neighbors, that is, preserve k edges (i, j) with the smallest values of w_{ij} (as determined by an appropriate proximity measure) and delete the remaining edges.
- *Threshold approach*: For every edge (i, j), delete it if the corresponding similarity measure w_{ij} is below (or above) a specified threshold.

Applying one or both of these approaches can, in many cases, eliminate most of the edges in the constructed graph and make sure that the remaining edges preserve the most meaningful similarity relationships between the data objects in the considered dataset. It should be noted that the similarity measures between each pair of vertices can be further updated to incorporate the idea of *shared nearest neighbors*. That is, if two vertices i and j are both connected to certain number of other vertices, the weight of the edge (i, j) is increased by the number of these shared neighbors. This idea can be used for constructing the *shared nearest neighbor (SNN) similarity graph*. In particular, the concept of SNN graph has been used in the Jarvis-Patrick clustering algorithm, which essentially performs clustering on the considered dataset by identifying connected components in the corresponding SNN graph [49].

Finalizing this section, it should be emphasized that in many real-world situations, one of the most crucial problems is to choose a suitable proximity measure which is the best fit for the considered type of data. For many cases with dense continuous data, metric distance measures such as Euclidean distance are appropriate and often used. On the contrary, for sparse data, measures that disregard 0–0 matches are frequently utilized. The cosine similarity, Jaccard, and Extended Jaccard measures are typically used for such data. For comparing time series, Euclidean distance or various correlation coefficients depending on the given data should be employed. If the time series contain similar quantities, then Euclidean distance can be used. On the contrary, if the time series represent different quantities (for instance, quantities of various substances in human organism in different biomedical studies), then one should use the one of aforementioned definitions of correlation coefficients.

In general, after the graph representing a dataset has been constructed and sparsified, one can use a variety of graph-theoretic concepts and techniques to extract useful information about the considered dataset and divide the dataset into a number of meaningful clusters according to the specified similarity criterion. The next section discusses relevant concepts, definitions, and optimization problems arising in this analysis.

3 Basic Concepts from Graph Theory and Their Data Mining Interpretation

To facilitate further discussion, several formal basic definitions and notations from graph theory will be presented, and the interpretation of the introduced concepts from the perspective of data mining and information retrieval will be discussed.

Let $G = (V, E)$ be an undirected graph with the set of N vertices V and the set of edges $E = \{(i, j) : i, j \in V\}$. Directed graphs, where the head and tail of each edge are specified, are considered in some applications. The concept of a *multigraph* is also sometimes used. A multigraph is a graph where multiple edges connecting a given pair of vertices may exist. A directed multigraph is a directed graph in which loops and multiple edges between any two vertices are permitted. Examples of different types of graphs are given in Fig. 6.

One of the important characteristics of a graph is its *edge density*: the ratio of the number of edges in the graph to the maximum possible number of edges. A *dense graph* is a graph in which the number of edges is close to the maximum possible number of edges. A *sparse graph* is a graph in which the number of edges is substantially smaller than maximum possible number of edges. In undirected graphs, edge density formally defined as

$$D = \frac{2|E|}{|V|(|V| - 1)}, \tag{1}$$

where $|E|$ denotes the number of edges and $|V|$ denotes the number of vertices in graph G. Clearly, the maximum number of edges is $\frac{1}{2}|V|(|V| - 1)$, so the highest possible edge density is 1 (for complete graphs) and the lowest edge density is 0. As indicated in the previous section, one often needs to eliminate a substantial number of edges from a graph representing a real-life dataset, so that the edge density is reduced and only the most meaningful connections are preserved.

Often, a real-world dataset can be represented as a graph with the particular structure, and the examined properties of such graphs can be used for data mining. Some examples of well-known graph structures are:

- *Complete graph*: a graph, which contains all possible edges, that is, any two vertices are adjacent.
- *Bipartite graph*: a set of graph vertices decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent.
- *Complete bipartite graph*: a bipartite graph such that every pair of graph vertices in the two sets are adjacent.
- *Linear (path) graph*: a graph, which vertices can be listed in order, $v_0, v_1, \ldots, v_n$, so that the edges are $(v_{i-1}, v_i) \in E$ for each $i = 1, 2, \ldots, n$. If a linear graph occurs as a subgraph of another graph, it is a *path* in that graph.
- *Cycle graph*: a graph containing a single cycle through all nodes.
- *Planar graph*: a graph whose vertices and edges can be drawn in a plane such that no two of the edges intersect.

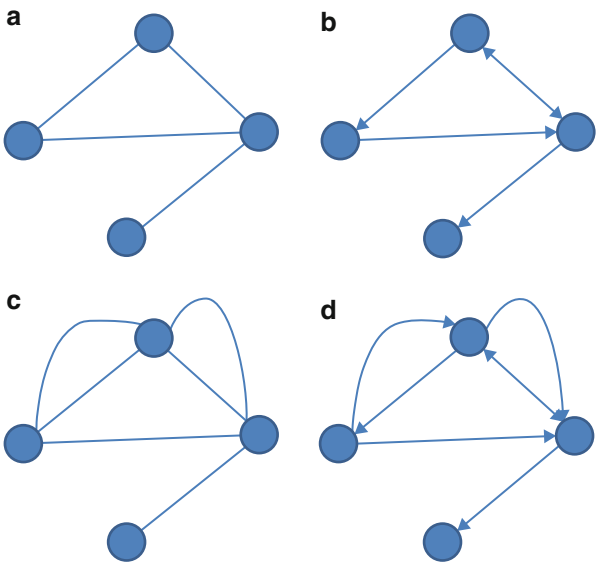

Fig. 6 Examples of different concepts of graphs. (**a**) Undirected graph. (**b**) Directed graph. (**c**) Multigraph. (**d**) Directed multigraph

- *Tree*: a connected graph with no cycles.
- *Forest*: a graph with no cycles, that is, the disjoint union of one or more trees.

3.1 Connected Components and Degree Distributions

The graph $G = (V, E)$ is *connected* if there is a path from any vertex to any vertex in the set V. If the graph is disconnected, it can be decomposed into several connected subgraphs, which are referred to as the *connected components* of G. A graph is called *K-vertex-connected* or *K-edge-connected* if there is no set of $K - 1$ vertices (edges, respectively) that disconnects the graph. A *K-vertex-connected* graph is often simply called K-connected. In many clustering techniques, distinct connected components are interpreted as distinct *clusters* in the corresponding dataset. It should be noted that although using connected components for clustering is rather common in a variety of applications, there are alternative graph-theoretic concepts that can be utilized in clustering problems. These concepts will be discussed in further sections.

The *degree* of a vertex is the number of edges emanating from it. For every integer k, one can calculate the number of vertices $n(k)$ with a degree equal to k, and then get the estimate of the probability that a vertex has the degree k as $P(k) = n(k)/n$, where n is the total number of vertices. The function $P(k)$ is referred to as the *degree distribution* of the graph. In the case of a directed graph, the concept of degree distribution is generalized: one can distinguish the distribution of *in-degrees*

and *out-degrees*, which deal with the number of edges ending at and starting from a vertex, respectively.

Degree distribution is an important characteristic of a dataset represented by a graph. It reflects the overall pattern of connections in the graph, which in many cases reflects the global properties of the dataset this graph represents. As mentioned above, many real-world graphs representing the datasets coming from diverse areas (Internet, telecommunications, finance, biology, chemistry, sociology) have degree distributions that follow the *power-law* model, which states that the probability that a vertex of a graph has a degree k (i.e., there are k edges emanating from it) is $P(k) \propto k^{-\gamma}$. Equivalently, one can represent it as $\log P \propto -\gamma \log k$, which demonstrates that this distribution forms a straight line in the logarithmic scale, and the slope of this line equals the value of the parameter γ. Figures 7 and 8 illustrate the power-law distribution in a regular and logarithmic scale.

An important characteristic of the power-law model is its *scale-free* property. This property implies that the power-law structure of a certain network should not depend on the size of the network. Clearly, real-world networks dynamically grow over time; therefore, the growth process of these networks should obey certain rules in order to satisfy the scale-free property. The necessary properties of the evolution of the real-world networks are *growth* and *preferential attachment* [17]. The first property implies the obvious fact that the size of these networks grows continuously (i.e., new vertices are added to a network, which means that new elements are added to the corresponding dataset). The second property represents the idea that new vertices are more likely to be connected to old vertices with high degrees. It is intuitively clear that these principles characterize the evolution of many real-world complex networks and massive datasets they represent.

From another perspective, some properties of graphs that follow the power-law model (and the corresponding datasets) can be predicted theoretically. Interesting properties of power-law graphs were studied using the theoretical *power-law random graph model* [5,31]. Among these results, one can mention the existence of a giant connected component (i.e., a giant cluster that contains $\Theta(N)$ vertices) in a power-law graph with $\gamma < \gamma_0 \approx 3.47875$ and the fact that all connected components (clusters) are small ($o(N)$ vertices) otherwise.[1] The emergence of a giant connected component (or a giant connected cluster) at the point $\gamma_0 \approx 3.47875$ is often referred to as a *phase transition*.

The size of connected components of the graph may provide useful information about the structure of the corresponding dataset, since, as indicated above, the connected components would normally represent clusters of "similar" objects. In many applications, decomposing the graph into a set of connected components can provide a reasonable solution to the clustering problem. From the computational perspective, identifying all connected components in a graph is a *polynomially*

[1] These results are valid *asymptotically almost surely (a.a.s.)*, which means that the probability that a given property takes place tends to 1 as the number of vertices N goes to infinity.

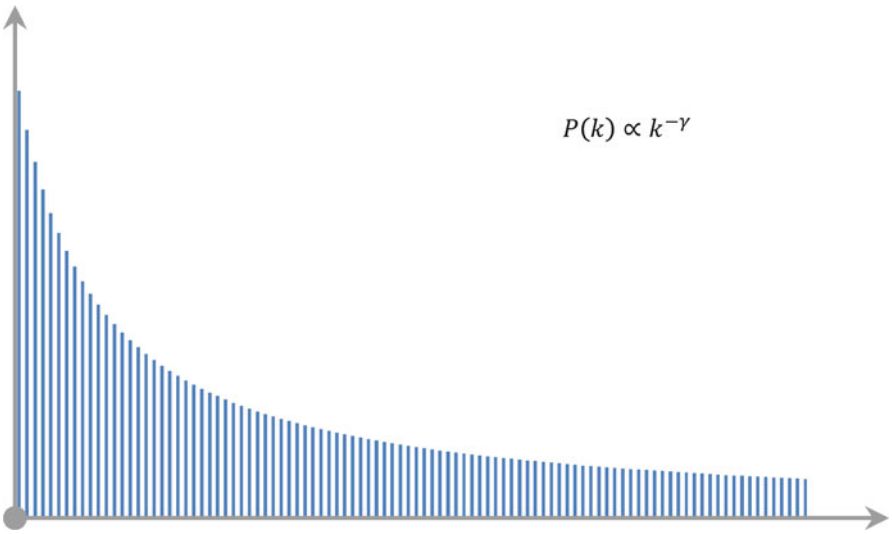

Fig. 7 A power-law distribution in the regular scale. The *horizontal axis* represents node degrees, and the *vertical axis* represents the corresponding probabilities (frequencies)

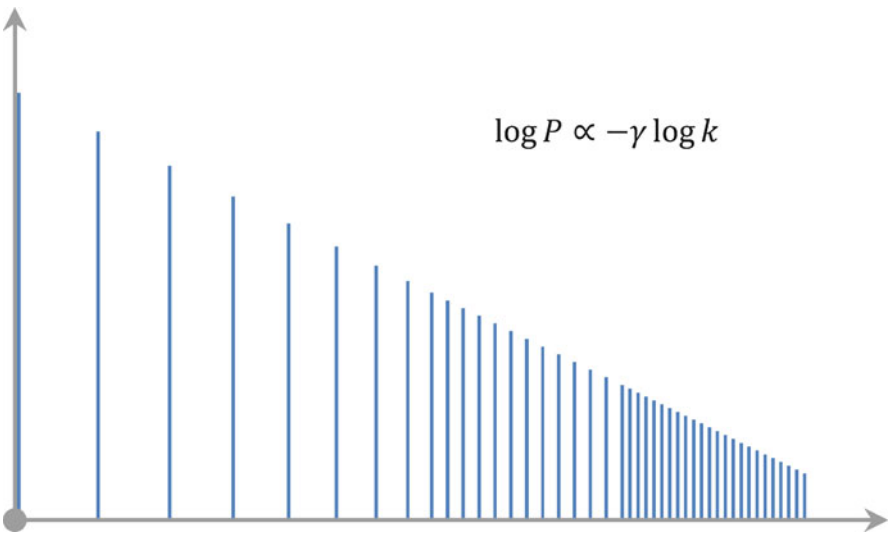

Fig. 8 A power-law distribution in the log-log scale. The interpretation of the axes is the same as in the previous figure

solvable problem, which is especially important when the considered graphs and datasets are very large.

However, in some situations, clusters based on connected components can be extremely large (e.g., comparable with the size of the whole graph, as indicated above),

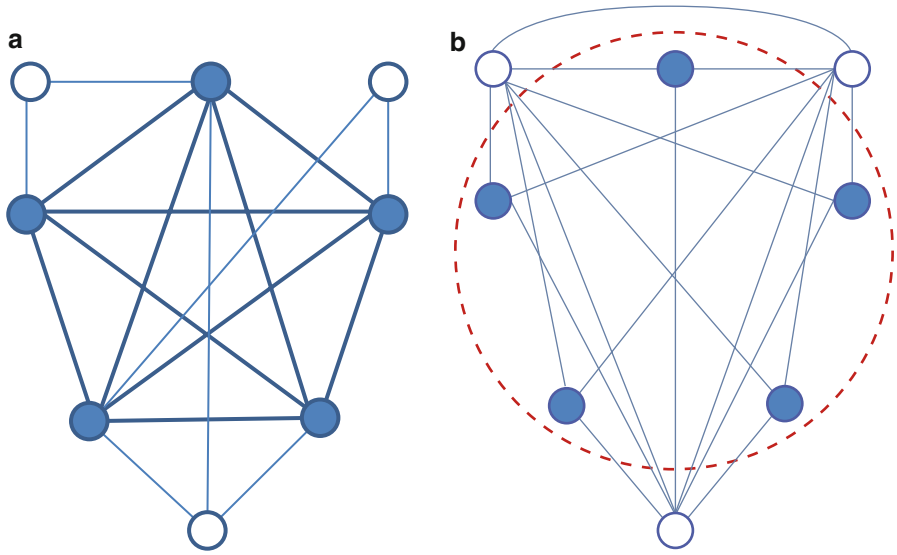

Fig. 9 Correspondence between the maximum clique and the maximum independent set in the complementary graph. (**a**) Clique in the graph (*size 5*). (**b**) Independent set in the complement graph (*size 5*)

and/or they may not be "tight enough," that is, the degree of connectivity within a cluster may not be sufficient to make reliable conclusions regarding the similarity of the data objects. This motivates one to look for substantially more "robust" clusters that have specific properties of their connectivity patterns. The corresponding graph structures that address these issues are discussed in the next subsections.

3.2 Cliques and Independent Sets

Given a subset $S \subseteq V$, $G(S)$ is denoted as the subgraph induced by S. A subset $C \subseteq V$ is a *clique* if $G(C)$ is a complete graph (i.e., it has all possible edges). Figure 9a shows an example of a clique in the graph.

One of the most well-known problems in combinatorial optimization, the *maximum clique problem* [26], is to find the largest clique in a graph. The maximum clique problem has several equivalent formulations as an integer programming (IP) problem, or as a continuous nonconvex optimization problem.

The simplest formulation of maximum clique problem is the following edge formulation [70]:

$$\max \sum_{i=1}^{N} x_i \tag{2}$$

s.t.

$$x_i + x_j \leq 1, \forall (i,j) \in \bar{E}$$

$$x_i \in \{0,1\}, i = 1, \ldots, N.$$

In the above formulation, $\bar{E}$ is the set of edges in the *complementary* graph $\bar{G}(V, \bar{E})$, defined as follows. If an edge $(i,j) \in E$, then $(i,j) \notin \bar{E}$, and if $(i,j) \notin E$, then $(i,j) \in \bar{E}$. Decision variables x_i defined as

$$x_i = \begin{cases} 1, & \text{if vertex } i \text{ is in the clique} \\ 0, & \text{otherwise.} \end{cases}$$

The objective (2) is to maximize the number of vertices, which are in the clique.

The maximum clique problem finds a clique of maximum cardinality. It can be easily transformed into the *maximum-weight clique problem*, which finds a clique of maximum weight, by introducing positive weights w_i associated with node i. Clearly, the objective of formulation (2) in the weighted case will be of the form

$$\max \sum_{i=1}^{N} w_i x_i$$

with the same set of constraints as in (2).

An *independent set* is a subset $I \subseteq V$ such that the subgraph $G(I)$ has no edges. Figure 9b shows an example of an independent set in the graph. The maximum independent set problem can be easily reformulated as the maximum clique problem in the *complementary* graph $\bar{G}(V, \bar{E})$. A maximum clique in $\bar{G}$ is a maximum independent set in G, so the maximum clique and maximum independent set problems can be easily transformed to each other and are essentially equivalent. Let $\mathcal{I}$ denote the set of all maximal independent sets of G. An alternative formulation is the following independent set formulation [70]:

$$\max \sum_{i=1}^{N} x_i \tag{3}$$

s.t.

$$\sum_{i \in S} x_i \leq 1, \forall S \in \mathcal{I}$$

$$x_i \in \{0,1\}, i = 1, \ldots, N,$$

where $S \subseteq V$ is a subset of graph G.

The advantage of formulation (3) over (2) is the fact that it is a tighter formulation, that is, the gap between the optimal values of (3) and its linear programming relaxation is smaller [70].

Clearly, locating cliques and/or independent sets in a graph representing a dataset provides important information in terms of identifying completely connected (or, on the contrary, completely disconnected) clusters. Therefore, cliques would naturally represent very *dense* clusters of similar objects. On the contrary, independent sets can be treated as groups of objects that differ from every other object in the group. This information may be important in some applications. In particular, it is often useful to find a *maximum clique or independent set* in the graph, since it would give the maximum possible size of the groups of "similar" or "different" objects.

Although finding large cliques in a graph representing a dataset may be useful in terms of identifying large tightly connected clusters, an important computational disadvantage of this approach is that the maximum clique problem (as well as the maximum independent set problem) is known to be NP-hard [44]. Moreover, it turns out that these problems are difficult to approximate [11, 46]. This makes these problems especially challenging in large graphs.

3.3 Clustering via Clique Partitioning

The problem of locating cliques and independent sets in a graph can be naturally extended to finding an optimal *partition* of a graph into a minimum number of distinct cliques or independent sets. These problems are referred to as *minimum clique partition* and *graph coloring* [43], respectively. Pardalos et al. [71] give various mathematical programming formulations of these problems. Clearly, as in the case of maximum clique and maximum independent set problems, minimum clique partition and graph coloring are reduced to each other by considering the complimentary graph, and both of these problems are NP-hard [44]. The example of optimal graph coloring is represented in Fig. 10. The simplest formulation of graph coloring for graph $G(V, E)$ is given below. Solving these problems for graphs representing real-life datasets is important from a data mining perspective, especially for addressing the clustering problem.

$$\min \sum_{k=1}^{N} y_k \tag{4}$$

s.t.

$$\sum_{k=1}^{N} x_{ik} = 1, \forall i \in V \tag{5}$$

$$x_{ik} + x_{jk} \leq 1, \forall (i, j) \in E \tag{6}$$

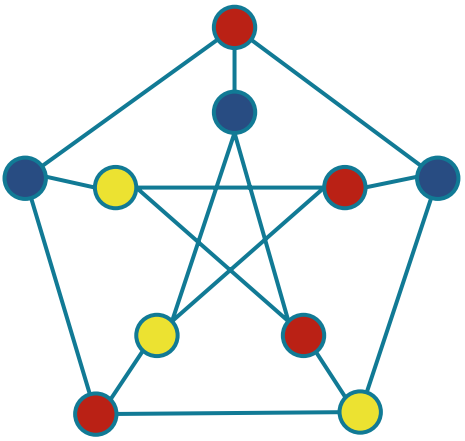

Fig. 10 Vertex coloring of the Petersen graph (minimum number of colors – 3)

$$y_k \geq x_{ik}, \forall i \in V, k = 1, \ldots, N \tag{7}$$

$$y_k, x_{ik} \in 0, 1, \forall i \in V, k = 1, \ldots, N. \tag{8}$$

In the above model, decision variables x_{ik} and y_k defined as

$$x_{ik} = \begin{cases} 1, & \text{if color } k \text{ is assigned to vertex } i \\ 0, & \text{otherwise} \end{cases}$$

$$y_k = \begin{cases} 1, & \text{if color } k \text{ is used} \\ 0, & \text{otherwise.} \end{cases}$$

Constraints (5) ensure that exactly one color is assigned to each vertex. Constraints (6) prevent adjacent vertices from having the same color. Constraints (7) ensure that no x_{ik} can be 1 unless color k is used. The optimal objective value gives the chromatic number of a graph ($\chi(G)$), and the sets $S_k = \{i | x_{ik} = 1\}, \forall k$, comprise a partition of the vertices the minimum number of independent sets.

Since the data elements assigned to the same cluster should be similar to each other, the goal of clustering is achieved by finding a clique partition of the graph, and the number of clusters will equal the number of cliques in the partition.

Similar arguments hold for the case of the graph coloring problem which should be solved when a dataset needs to be decomposed into the clusters of "different" objects (i.e., each object in a cluster is different from all other objects in the same cluster) that can be represented as independent sets in the corresponding graph. The number of independent sets in the optimal partition of graph G is referred to as the *chromatic number* ($\chi(G)$) of the graph.

3.4 Using Clique Relaxations for Graph-Based Clustering

Although cliques and independent sets address the issue of "robust" tightly connected clusters, a significant practical drawback of these structures is that they are often *too restrictive*, that is, all pairs of vertices need to be connected (disconnected). On the contrary, as mentioned above, a simple connectivity requirement is not restrictive enough to guarantee sufficient tightness of a cluster. In an attempt to provide a tradeoff between the clique-based and sparse connected component-based clusters, the concepts of *clique relaxations* are introduced.

Several concepts of clique relaxations first appeared in studying cohesive subgroups in social networks and were based on relaxing some of the desirable properties that a clique idealizes. For instance, instead of cliques and independent sets, one can consider *quasi-cliques* and *quasi-independent sets* and partition the graph on this basis. Quasi-cliques are subgraphs that are *dense enough* (i.e., they have a sufficiently high edge density, but the edge density does not have to be 1 as in the case of cliques). Formal definitions of a quasi-clique and other clique relaxations will be given below.

It is often reasonable to relate clusters to quasi-cliques, since they still represent sufficiently dense clusters of similar objects [50, 88]. Obviously, in the case of partitioning a dataset into clusters of "different" objects, one can use quasi-independent sets (i.e., subgraphs that are sparse enough) to define these clusters.

Other types of clique relaxations have also been introduced. Many of these definitions come from the studies of social networks; however, these concepts clearly have important data mining interpretations. The main idea behind these concepts is to "relax" certain properties of a clique while still maintaining sufficient connectivity and robustness characteristics of the obtained network structures. Note that in many cases, the size of these clique relaxations is substantially larger than the size of cliques, which provides a significant advantage in situations when a *large tightly connected cluster* needs to be identified. In addition, an important factor that motivates the use of these clusters instead of cliques is the fact that they are not as sensitive to potential errors in data collection and measurement. For instance, if one link in a clique-based cluster is missing because of an error, the structure of the clique is violated, and the corresponding cluster may not be adequately reflected in data mining results. However, if one uses quasi-clique-based clusters, the absence of several edges due to such errors is usually not as critical, since it will likely not have a significant effect on the required edge density of the cluster.

The ideas for the clique relaxation concepts discussed below originally come from the study of social networks; however, these definitions can be efficiently utilized in the data mining context. More specifically, there are three main directions for possible relaxations of the clique definition:

1. *Density-based relaxations* – relaxing the requirement for the edge density of a clique to be 1: *quasi-cliques (γ-dense subgraphs)* [3]

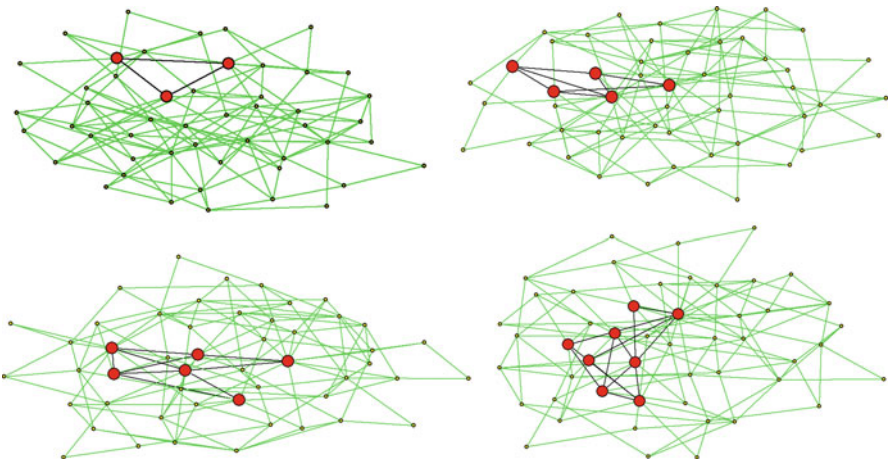

Fig. 11 Visual comparison of maximum clique ($\gamma = 1$) versus maximum quasi-clique ($\gamma = 0.7, 0.6, 0.5$) in the same uniform random graph $G(50, 0.1)$

2. *Degree-based relaxations* – relaxing the requirement that the degree of each node in a clique of size q to be $q - 1$: *k-plexes* [78]
3. *Path (diameter)-based relaxations* – relaxing the requirement that the length of the path between any two nodes in a clique to be 1: *k-cliques* [59], *k-clubs*, and *k-clans* [66]

As mentioned above, a *quasi-clique* (also referred to as a γ-clique or a γ-dense subgraph) is a subgraph that has the edge density of at least γ, where $\gamma \in (0, 1]$. Clearly, a quasi-clique becomes a clique when $\gamma = 1$. A more rigorous definition is provided below. Let $G = (V, E)$ be the graph with the set of vertices G and the set of nodes E. Denote the graph induced by the vertex subset $S \subseteq G$ by G_S. Under these notations, G_S is a quasi-clique (γ-dense subset) if $|E(G_S)| \geq \gamma \binom{|V(G)|}{2}$ (recalling that $\binom{|V(G)|}{2} = \frac{|V(G)|(|V(G)|-1)}{2}$) and it is connected. The illustrative example is shown in Fig. 11. The given graph $G(50, 0.1)$ is a random graph on 50 vertices with probability $p = 0.1$ that each edge exists. The size of the maximum clique is equal to 3, and the size of the maximum quasi-clique with $\gamma = 0.7$ is equal to 5, with $\gamma = 0.6$ is equal to 6 and with $\gamma = 0.5$ is equal to 8. As one can observe, the sizes of quasi-cliques are substantially larger than the size of the maximum clique, whereas the density of these clusters is still high. Therefore, clusters based on quasi-cliques may be more meaningful as they include a larger number of data elements, which are still "similar enough" to be included in the same cluster.

A *k-plex* is a subgraph in which the degree of each node is at least $q-k$ (assuming that q is the number of nodes in this subgraph). Figure 12 presents an illustrative example of k-plexes in a small graph on 7 vertices.

A *k-clique* is a subgraph where the length of the path between any two nodes is at most k (note that other nodes in this path are *not required* to belong to the k-clique). Consider the graph $G = (V, E)$ as in the previous definition and the subset $S \subseteq G$

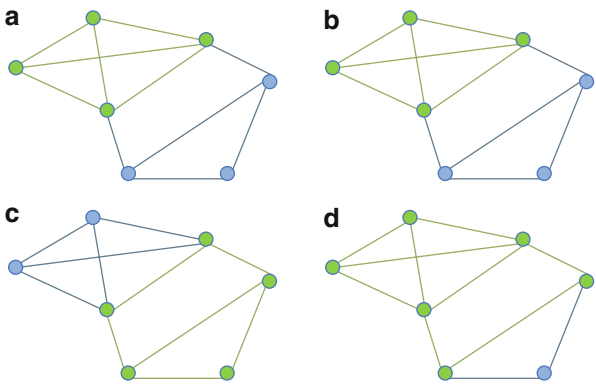

Fig. 12 Maximum k-plexes in the graph for different k. (**a**) Maximum 1-plex (clique) subgraph in the given graph. (**b**) Maximum 2-plex subgraph in the given graph. (**c**) Maximum 3-plex subgraph in the given graph. (**d**) Maximum 4-plex subgraph in the given graph

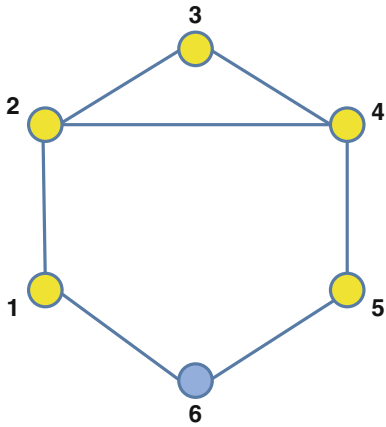

Fig. 13 The example of 2-clique but not 2-club

denoted by G_S. Then, G_S is a k-clique if $\forall i, j \in V$ it is given that $d_G(i, j) \leq k$. Note that distances between any two points in k-clique can be greater than k because some paths might pass through the complementary graph $G \backslash G_S$ to the k-clique. An example illustrating this fact is mentioned in [7] and given in Fig. 13. As one can see, there is the set $L = \{1, 2, 3, 4, 5\}$ that forms a 2-clique, but the diameter of this cluster is equal to 3. Therefore, the concept of k-clique does not necessarily embody the idea of "tightness" of the corresponding cluster of points in a graph. In order to ensure a higher level of tightness and low diameter of a cluster, the concept of a k-clique can be "upgraded" to a k-club (Fig. 14).

A *k-club* is a subgraph that has a *diameter* of at most k (note that in this definition, all the nodes in the shortest path that connects any pair of nodes within a k-club have to also belong to this k-club). Using the same notations as in previous definitions, it can be pointed out that G_S is a k-club if $\forall i, j \in V$ it is given that $d_{G_S}(i, j) \leq k$. Thus, k-club can be viewed as a "tighter" structure than a k-clique. It is more applicable for connectivity and clustering problems on networks where cluster "cohesiveness" and "robustness" issues play an important role.

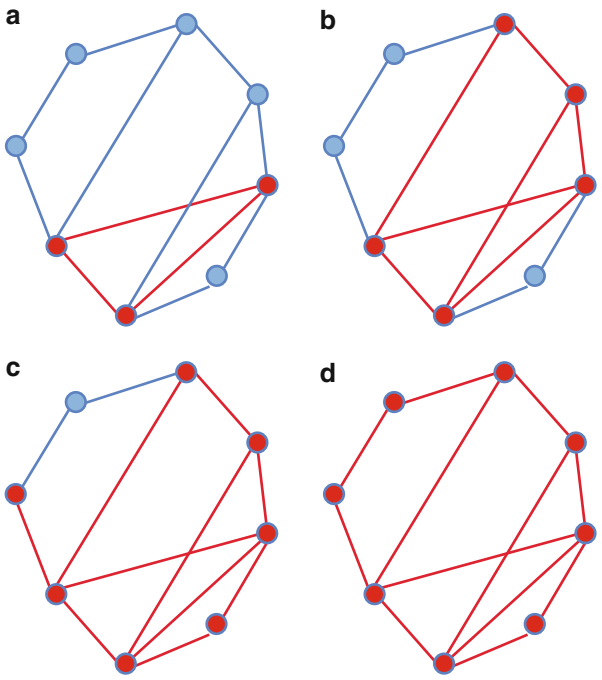

Fig. 14 Maximum k-clubs in the graph for different k. (**a**) Maximum 1-club (clique) subgraph in the given graph. (**b**) Maximum 2-club subgraph in the given graph. (**c**) Maximum 3-club subgraph in the given graph. (**d**) Maximum 4-club subgraph in the given graph

Although the given definitions are rather straightforward and intuitively clear, mathematically rigorous studies on related optimization aspects (e.g., mathematical programming formulations for finding the largest clique relaxation-based clusters in graphs) have started to appear only within the past few years.

For instance, an important combinatorial optimization problem associated with quasi-cliques is the maximum γ-clique problem. Pattillo et al. [72] proved that the decision version of the γ-clique problem is NP-complete for any fixed positive $\gamma \in (0, 1)$.

Furthermore, linear 0–1 formulations for the maximum γ-clique problem were proposed in [72]. Consider the problem of finding a maximum γ-clique in the graph $G = (V, E)$. If for some subgraph G_S one wants to check whether this subgraph is a γ-clique, one can define $x = (x_1, \ldots, x_N)$ as a 0–1 vector with $x_i = 1$ if node i belongs to G_S, or zero otherwise. Since $x_i^2 = x_i \ \ \forall i$, then the subgraph G_S is a γ-clique if it consists of $\frac{1}{2}\gamma \sum_{i,j=1,i\neq j}^{N} x_j x_i$ arcs. The number of arcs in the subgraph G_S can be calculated as

$$\frac{1}{2}x^T Ax = \sum_{i,j=1, i\neq j}^{N} a_{ij}x_j x_i.$$

Therefore, the problem of finding the maximum γ-clique in the graph G can be formulated as follows:

$$\max \sum_{i=1}^{N} x_i.$$

subject to

$$\sum_{i,j=1, i\neq j}^{N} a_{ij}x_j x_i \geq \gamma \sum_{i,j=1, i\neq j}^{N} x_j x_i.$$

This is the problem with a linear objective and one quadratic constraint. In [72], two possible linearizations of this problem were proposed.

The simplest linearization is rather straightforward with the main idea of introducing new variables $w_{ij} = x_i x_j$. The constraint $w_{ij} = x_i x_j$ is equivalent to

$$\begin{aligned} w_{ij} &\leq x_i, \\ w_{ij} &\leq x_j, \\ w_{ij} &\geq x_i + x_j - 1. \end{aligned}$$

Now, the problem can be formulated as the following linear 0–1 problem:

$$\max \sum_{i=1}^{N} x_i$$

subject to

$$\begin{aligned} \sum_{i,j=1, i<j}^{N} a_{ij}w_{ij} &\geq \gamma \sum_{i,j=1, i<j}^{N} w_{ij}, \\ w_{ij} &\leq x_i, \\ w_{ij} &\leq x_j, \\ w_{ij} &\geq x_i + x_j - 1. \end{aligned}$$

This problem contains $N(N-1)/2$ variables and $\frac{3}{2}N(N-1)+1$ constraints, that is, it has $O(N^2)$ entities. Besides this formulation, the problem of finding a maximum γ-clique can also be represented by the linear 0–1 formulation with $O(N)$ variables and constraints as described in [72], although this linearization is not as straightforward as the one mentioned above, since it explicitly uses the special structure of the considered problem. Despite the fact that the second linearization is more compact, it does not always provide better computational performance

compared to the $O(N^2)$ linearization, although it does perform faster on sparse graphs and relatively high values of γ, which is a typical setup in many situations that deal with finding dense clusters in sparse networks representing real-world datasets.

Besides the maximum quasi-clique problem, similar problems for other types of clique relaxations representing tight clusters (in particular, k-plexes and k-clubs) have been considered in the recent literature from the combinatorial optimization perspective. The maximum k-plex problem has been addressed by Balasundaram et al. in [15], where they provide the most compact formulation with N variables and constraints. It has also been proven in [15] that the decision version of this problem is *NP*-complete for any fixed positive integer k.

The maximum k-club problem was considered in several previous studies. The first mathematical programming formulations (linear 0–1 formulations containing $O(N^k)$ variables and constraints) and complexity analysis of this problem were addressed in [14]. In a recent study, Veremyev and Boginski in [82] developed a new compact linear 0–1 programming formulation for finding maximum k-clubs that substantially reduces the number of entities compared to the known formulations with $O(kN^2)$ entities and allows one to find exact solutions to problems with larger values of k, which can provide more flexibility in terms of modeling k-club-based clusters.

In conclusion, utilizing clique relaxations in the context of clustering in data mining appears to be a promising approach with attractive characteristics, since it provides sufficient flexibility in terms of the tightness and size of the obtained clusters. The chapter describes more details on clique relaxation models in graph theory.

4 Examples of Real-World Applications

The aforementioned network-based data mining techniques have been used in a variety of real-world applications during recent years. Although it is impossible to describe all the applications in detail within one chapter, several examples of networks and datasets that have been addressed in the recent literature will be briefly discussed.

4.1 Biological Networks

Nowadays, representing complex biological systems as networks and studying the properties of these networks has become a very popular tool in computational biology and biomedical applications. For instance, an overview of graph-based molecular data mining approaches is presented in [41]. An extensive recent volume [30] covers many topics related to clustering problems in biological network applications, such as the analysis of regulatory and interaction networks from clusters of

co-expressed genes, graph-based approaches for motif discovery, identifying critical nodes in protein-protein interaction networks, genetic graph partitioning algorithms, and so on. The most recent review of network-based approaches in biomedical applications is presented in [18].

As new advances in experimental design produce a large amount of data to describe biological interactions at a genome scale, this data is increasingly being deposited into biological databases. These interaction networks include, among others, protein interaction networks [54, 77], metabolic networks [33, 69], gene regulatory networks [6], and signal transduction networks [80]. Since there is evidence that preserved interaction pathways exist across organisms, efficient algorithms to analyze common pathways in these networks can make significant contributions to understanding of these networks. The most common representation of such networks is a graph with vertices representing biological entities and edges representing interactions between them. With this representation, one can look for the presence of special substructures in these graphs to answer various questions. By studying paths in these graphs, one can investigate the properties of pathways and their relationships, and by finding similar subgraphs, one can search for *network motifs* and study common substructures embedded in these networks. While path matching allows the study of individual "linear" paths or chains, graph matching allows one to investigate entire "nonlinear" pathways or functional modules. Efficient algorithms for solving these problems give biologists an opportunity to study evolutionary mechanisms such as pathway conservation, duplication, and specialization. Thus, one important strategy is to start from a given pathway of interest and find related pathways within the same organism. By comparing proximities and differences in the returned pathways, it is possible to evaluate various hypotheses concerning evolution. Although the path matching problem models linear interaction chains well, many biological pathways are more complex and may consist of multiple interacting components [86].

To solve the problem of finding similar paths in two graphs, a combined graph from the two given graphs should be constructed, so that each vertex in the combined graph represents a pair of related vertices, one from each of the two given graphs, and each pathway alignment is represented as a simple path in the combined graph. A randomized algorithm for finding simple paths by imposing acyclic edge orientations may be useful in this case. The main difficulty here arises from the presence of cycles in the given graphs. To avoid substantial repetitions of vertices in a path, the problem of finding similar paths in two graphs has been reduced to the NP-hard problem of finding high-scoring simple paths of a given length in a combined graph (see [54]).

In contemporary post-genomic era, protein interactions are well captured by so-called protein-protein interaction networks. In particular, protein-protein interaction networks (see Fig. 15) have been experimentally constructed for many known organisms. A protein interaction network is represented by a graph with the proteins as nodes, and an arc exists between two nodes if the proteins are known to interact. Also, it has been discovered that the degree distribution of many of these complex protein interaction networks follows a power law. For instance, the protein

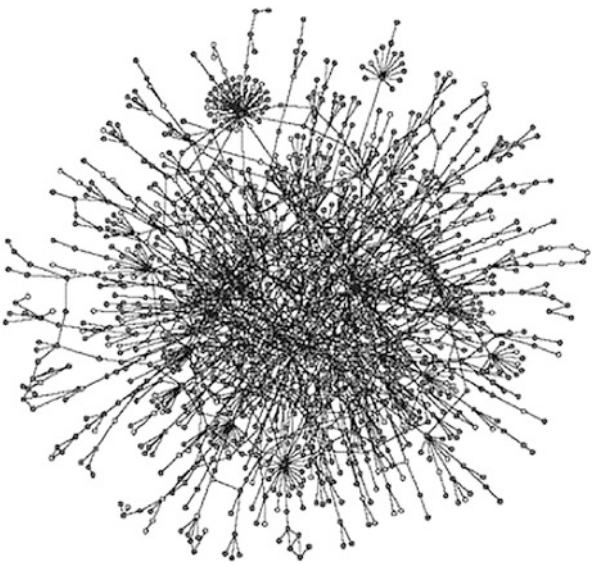

Fig. 15 A cluster in the protein-protein interaction network of the yeast [87]

interaction network of S. cerevisiae, as well as many other networks of this type, has been experimentally shown to have a power-law degree distribution. Due to the power-law structure, an average node degree is no longer representative, and the majority of nodes have few neighbors, while a smaller number of nodes have very high degrees. Many problems for biological networks can be solved via identification of large clusters or functional modules. Cliques have formed the basis for several studies that attempt to decompose a protein interaction network into functional modules and protein complexes. Protein complexes are groups of proteins that interact with each other at the same time and place, while functional modules are groups of proteins that are known to have pairwise interactions by binding with each other to participate in different cellular processes at different times. Clique models have been popular in this area as they represent tight clusters in a network, but sometimes they are too restrictive, as indicated earlier in this chapter. Therefore, employing clique relaxations instead of cliques may be useful in this case. Network-based clustering techniques utilizing clique relaxations (e.g., k-clubs) have been applied to studying those networks in [13] and [14].

There are several computational methods to measure sequence similarities between proteins. Some homology search algorithms capture similar database sequences to a query from a database and calculate the statistical significance of their similarities. However, retrieved proteins with evident similarities are often uninformative from a practical perspective. To cope with this problem, it is important to consider alternative approaches, such as signature identification and sequence clustering. Functionally or structurally related proteins often have locally conserved regions, referred to as functional motifs or signatures. Proteins sharing the same signatures do not always exhibit apparent similarities between

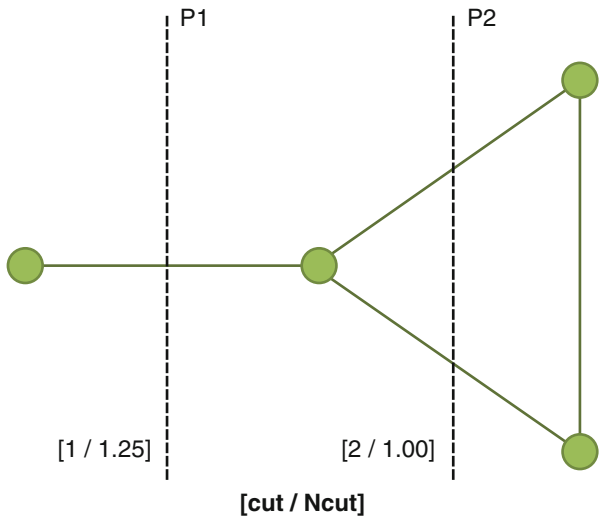

Fig. 16 An instance of a graph representation of protein sequence similarities. Graph partitions are shown by *dashed lines P1* and *P2* relatively. Each of these lines partitions vertices of the graph into two distinct sets. Numbers below the *dashed lines* show the capacities of their cuts and normalized cuts

their sequences. Therefore, they are referred to as *distantly related proteins*. Massive amounts of signature data have been accumulated in special signature databases. These databases enable one to accurately find proteins that share signatures and their relative positions on the protein sequences. Clustering protein sequences is an important approach to finding distant relationships. Distantly related proteins may be grouped together into a cluster even though they do not show apparent similarities. The most well-known and simplest method for clustering proteins is *single linkage* (SL). A detailed description of SL method can be found in [57]. Although the method is simple, fast, and widely used, it has difficulty in detecting an appropriate cutoff score for sequence similarity. More advanced algorithms employ a graph representation of a given set of proteins. Such algorithms are *SL-based clustering method*, named SL-KL (see [52]), and *p-quasi complete linkage algorithm* (see [61]). They allow some proteins to overlap between two or more groups. However, such clustering techniques usually carry high computational costs. Therefore, in order for the algorithms to be applicable to a large number of proteins, attention should be focused on approximate distantly related proteins without overlapping groups. This approach to clustering proteins is based on iterative partitioning. The key concept of graph partitioning is a "cut," which is a set of edges between two distinct sets of nodes (or proteins). When the sum of edge weights in a cut (referred to as the cut capacity) is small, the two sets are considered to be dissimilar.

Figure 16 shows a very simple example of graph representation of protein similarities, where for simplicity all edges have unit weights. The proteins are

naturally grouped into two sets on the left and right side of the cut P2. However, P1 is a partition with the lowest capacity of the cut, called the *minimum cut*, due to the sparseness of the set of edges (e.g., one edge) connecting the proteins on the left side of P1. Unfortunately, it is difficult for applications to find distant relationships in a protein set containing various kinds of signatures using minimum-cut partitioning. The *normalized cut partitioning* is proposed to avoid the tendency of inaccurate clustering by the minimum-cut partitioning. It uses the normalized cut capacity (Ncut) criterion, which is the ratio of the capacity of a cut to the sum of similarities in each of the two distinct sets. In the context of the graph in Fig. 16, P2 is a partition with the minimum normalized cut. In addition, a locally minimal cut criterion can also be used. All the aforementioned methods are described in detail in [53].

The genome is also a highly interactive system, and the expression of a gene depends on the activity of other genes. Gene expression data is often represented in a network format. Cliques are widely used for gene co-expression networks' clustering. In the case of gene expression data, nodes represent the genes and arcs represent the relations between co-expressed genes with correlation higher than a specified threshold (see [51]) based on microarray experiments. Quasi-clique-based clusters can also be identified in these networks [73].

The main challenge in the analysis of this data is to understand biological functions from the topology of the network. There is a rather useful approach based on efficient *sequence alignment* algorithms and a statistical theory to assess the significance of the results (see [35]). Another way to address these challenges leads to an algorithmic procedure referred to as *local graph alignment*, which is conceptually similar to sequence alignment. It is based on a scoring function measuring the statistical significance for families of mutually similar subgraphs. This scoring involves quantifying the significance of the individual subgraphs, as well as their mutual similarity. As a computational problem, graph alignment is more challenging than sequence alignment. Sequences can be aligned in polynomial time by using dynamic programming algorithms. For graph alignment, the existence of a polynomial-time algorithm is unlikely. Thus, an important issue for graph alignment is the construction of efficient heuristic search techniques, such as those described in [19].

4.2 Chemical Networks

Chemical datasets are often represented as graphs, since they are natural representations of chemical compounds. Such graphs can be used to model topological and geometric characteristics of chemical structures. The nodes correspond to atoms, and the edges correspond to bonds connecting the atoms. For instance, Fig. 17 shows ball-and-stick diagrams of chemical compounds pentacene (Fig. 17a) and lysozyme (Fig. 17b).

Chemical graph mining techniques have many applications in the drug discovery procedure that include structure-activity-relationship (SAR) model construction and bioactivity classification. Many data mining algorithms are based on the assumption

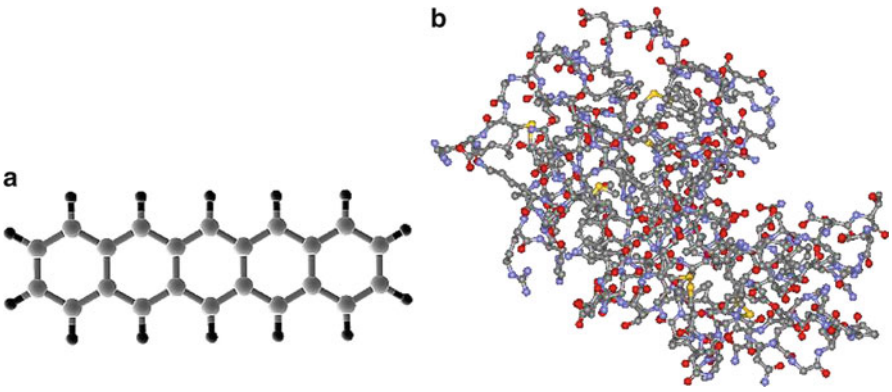

Fig. 17 Representation of chemical data. (**a**) Pentacene molecule. (**b**) Lysozyme molecule

that the properties of a chemical element are related to its structure. For further details, see [83]. Studying and mining chemical graphs can provide new perspectives to chemistry, biology, and toxicology.

In recent years, a lot of algorithms on frequent graph pattern mining and related chemical graph mining problems have been published. The amount and complexity of the available data is steadily increasing. Within molecular databases, it is interesting to find patterns (fragments) that appear at least in a certain percentage of graphs. Another problem is to find fragments that are frequent in one part of the database but infrequent in the other. Recently developed approaches for chemical graph data mining include *Subdue* (see [32]), *inductive logic programming* (see [40]), and *Gaston* (see [68]).

Chemical graph data mining can be considered in the context of search in the lattice of all possible subgraphs. In Fig. 18, a small example is shown based on a small molecule of nitrous acid (shown in the bottom of the figure). The four nodes are labeled corresponding to their atom types (H, N, and O), and the edges are labeled with their corresponding bond types (single and double bonds). All possible subgraphs of this small graph are shown in the figure. At the top, the empty graph is denoted by $\emptyset$. In the next row, all possible fragments containing just one atom are listed and so on. At the bottom of the figure, the complete molecule with three bonds is given. Many chemical graph mining algorithms are based on the common idea of performing search this subgraph lattice. They are interested in finding a subgraph (or several subgraphs) that is the most frequent in the considered graph. Building this lattice of frequent subgraphs involves two main steps: *candidate generation*, where new subgraphs are created out of smaller ones, and *support computation* where the frequency or support of the new subgraphs in the database is determined. Both steps are computationally challenging, and various algorithms and techniques have been developed to find frequent subgraphs in reasonable time. For more details, see [42].

Another approach utilized in chemical graph data mining is based on kernels. Kernel methods have emerged as an important class of machine-learning methods

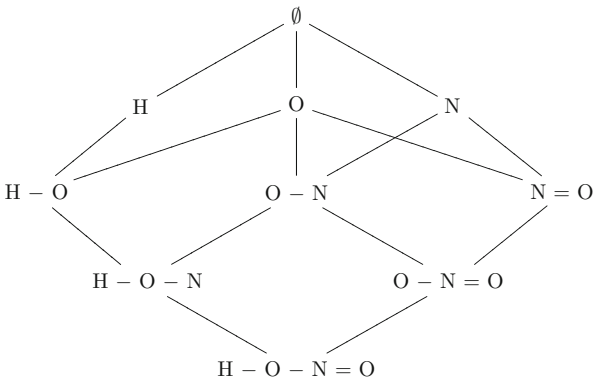

Fig. 18 The lattice of all subgraphs in nitrous acid HNO_2

suitable for variable-size structured chemical data (see [76]). Applications of kernel methods to molecular bond graphs require the construction of graph kernels, that is, functions that measure the similarity between graphs with labeled nodes and edges. Also, kernel methods can be applied to molecular clustering and regression problems, such as predicting the boiling point of alkanes and other organic substances [12].

4.3 Brain Networks

One more interesting application of network-based data mining is associated with studying human brain. It is a significant practical task since the results of this analysis are important in medical practice, especially studying brain diseases. The analysis of connectivity patterns of the brain function is extremely challenging because of the complex structure with the huge numbers of neurons and dynamic nature of connections between them. According to [67], the number of neurons is estimated to be 8.3×10^9, and the number of connections is approximately 6.6×10^{13}. The empirical analysis of such structures represented as a graph with neurons as nodes is computationally prohibitive.

Several aspects of the analysis of brain connectivity are studied in [48]. One can potentially use the following technique to analyze the graph representing the brain. Different functional units of the brain containing a large number of neurons can be considered as a nodes in the graph. This approach enables the use of different algorithms proposed for much smaller graphs. This method can be applied in order to study properties of the connections between these functional units. In [37], the authors applied this approach to study the cortical visual system of a macaque monkey represented by a graph.

A similar approach can also be applied to study some properties of the human brain. The major areas of the human brain and their functions are represented on Fig. 19. In [36], the authors investigated a graph corresponding to 147,456

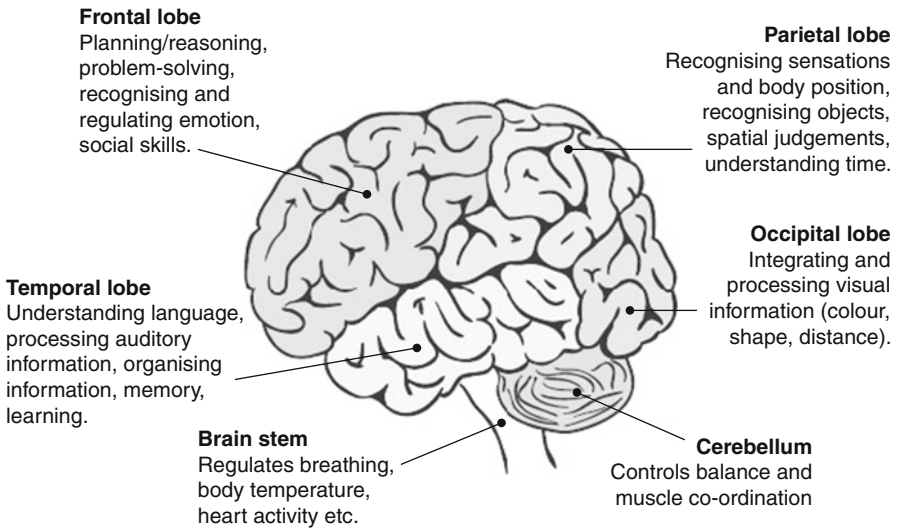

Fig. 19 Functional areas of the human brain

functional units of the human brain, which were selected by dividing the entire brain into a set of $64 \times 64 \times 36$ voxels of a small size. The set of edges of this graph were constructed using the time series defined by the recorded signals representing the activity of each functional unit. The correlation coefficients were calculated according to the formula similar to (9) using the time series representing the signals obtained from functional units, and the corresponding nodes were connected if the correlation exceeded a certain threshold value. It turned out that the resulting brain graphs also follow the power-law distribution with the parameter $\gamma \approx 2$ [36].

Investigating different characteristics of brain networks is a crucial practical task. One can study various structures (network motifs) in the brain networks as spanning trees, cliques/independent sets, clique relaxations, etc., which may provide a new insight into the process of signal propagation between the functional brain units and neurons. This information could potentially be very useful in studying brain disorders. More detailed information on network models in brain dynamics is given in the chapters [75].

4.4 Telecommunication/Information Exchange Networks

Large-scale information exchange networks arise in a great variety of areas, including large-scale wireless communication/sensor networks, the Internet/World Wide Web, and telephone traffic networks. The Internet, the World Wide Web, and the telephone call networks (also referred to as call graphs) were experimentally shown to have a power-law structure. Studying the structural properties and the global

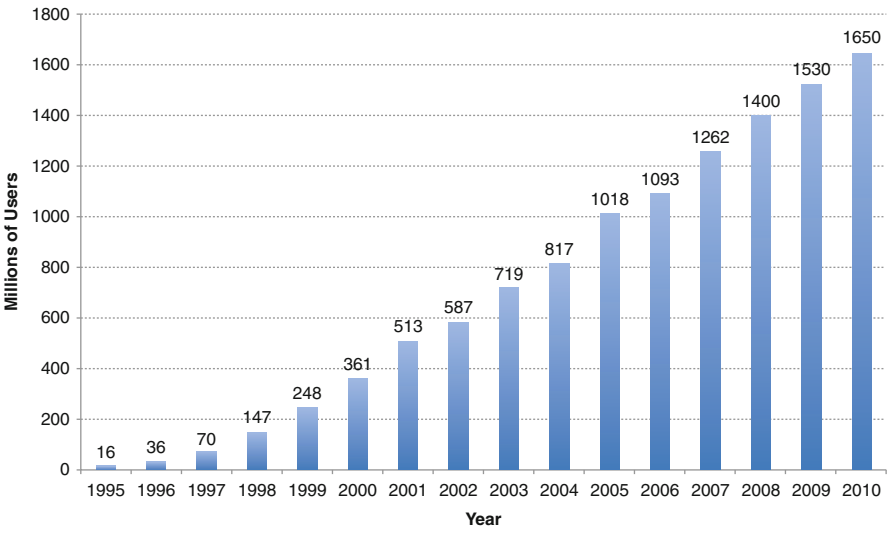

Fig. 20 Internet users in the world. Growth 1995–2010 [65]

organization of these networks has been addressed in recent literature. Graph theory has been applied for web search [28, 56], web mining [63, 64], and other problems arising in the Internet and World Wide Web [9]. In an interesting experimental study of the World Wide Web, Broder et al. [29] used two Al tavista crawls, each with about 200 million pages and 1.5 billion links and identified certain properties of the connected components of this graph. Studying the structural properties of the Internet is an exciting area of ongoing research [85]. The necessity of such research is justified because the usage of the Internet increases very rapidly in the world over years. The graph on the Fig. 20 represents the growth of number of Internet users through 1995–2010 [65]. The overall structural characteristics of the Internet are sometimes referred to as "robust yet fragile" [34], which implies the overall robustness with respect to random disruptions/errors, but potential vulnerability to targeted attacks on "hubs" (i.e., high-degree vertices).

Another well-known experimental study of a large communication network was conducted on the "call graph" representing daily telephone traffic data from AT&T telephone billing records [1]. The call graph is also very large, and the considered instance contained 53,767,087 vertices and over 170 million edges. The size of cliques and quasi-cliques in this graph was investigated, and it turned out that the size of the largest clique did not exceed 32, and the size of the largest quasi-clique with $\gamma \geq 0.5$ did not exceed 96. Since the size of the considered graphs is extremely large, the problems of identifying tight clusters, such as cliques and quasi-cliques, cannot be solved exactly due to computational challenges. Therefore, appropriate heuristic algorithms need to be applied to tackle these problems. For instance, in [1], the greedy randomized adaptive search procedure (GRASP) [38, 39] was successfully used.

4.5 Financial Networks

Stock market data can also be represented as a network and analyzed using combinatorial optimization techniques. One can represent each traded stock by a node in a network, and a given pair of nodes will be connected by an edge if the corresponding stocks exhibit a similar behavior over a certain period of time (this similarity can be measured as a correlation between the corresponding time series, and a threshold can be set, so that an edge would be place if this threshold is exceeded) [23].

Boginski et al. [24, 25] have previously done the work on studying the behavior of the US stock market and modeling the corresponding massive dataset as a large-scale network (referred to as the *market graph*).

It is possible to construct the market graph in a relatively simple way [23]. As it was mentioned, the set of nodes of such graphs corresponds to the set of stocks. For each pair of stocks i and j, the correlation coefficient C_{ij} is calculated using the following procedure. Let $P_i(t)$ denote the price of the instrument i at time t. Then,

$$R_i(t, \Delta t) = \log \frac{P_i(t + \Delta t)}{P_i(t)}$$

defines the natural logarithm of return of the stock i over the certain period of time $[t, t + \Delta t]$.

Correlation coefficients C_{ij} between all pairs of stocks i and j are calculated as

$$C_{ij} = \frac{\mathbb{E}[R_i R_j] - \mathbb{E}[R_i]\mathbb{E}[R_j]}{\sqrt{\mathbb{V}ar[R_i]\mathbb{V}ar[R_j]}}, \tag{9}$$

where $\mathbb{E}[R_i]$ is the average return of the instrument i over T considered time units [58, 60, 74]:

$$\mathbb{E}[R_i] = \frac{1}{N} \sum_{t=1}^{T} R_i(t).$$

If one defines a threshold $\theta \in [-1, 1]$, then an undirected edge between the nodes i and j is added to the graph if the corresponding coefficient $C_{ij} \geq \theta$. Usually, the value of θ is chosen to be significantly larger than zero, and therefore, an edge between two nodes reflects the fact that stocks i and j are significantly correlated.

The values of the correlation threshold θ can be changed in order to construct market graphs where the edges between the nodes reflect different correlation between the corresponding stocks. As the threshold value θ increases, the number of edges in the market graph decreases.

The edge density (as defined in Sect. 3) of the market graph is a measure of the fraction of pairs of stocks exhibiting a similar behavior over some certain time period. One can define different "levels" of this similarity by specifying different

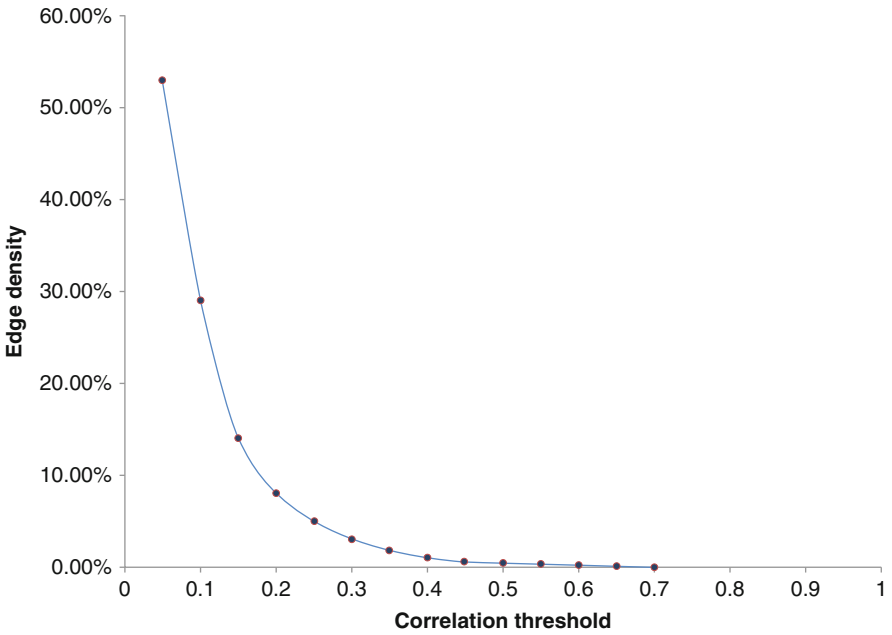

Fig. 21 Edge density of the market graph for different values of the correlation threshold

values of θ. Figure 21 represents the plot of the edge density of the market graph as a function of θ [23].

In [20], the authors also looked at the changes of the edge density of the market graph over time. The authors analyzed these dynamics for 11 overlapping 500-day period in 2000–2002 (the 1st period was the earliest, and the 11th period was the latest). A relatively large value of θ ($\theta = 0.5$) was chosen in order to consider only highly correlated stocks. It turned out that the edge density of the market graph corresponding to the 11th period was more than 8 times higher than for the first period. The corresponding plot is shown in Fig. 22. This jump of the edge density suggests that there is a trend to the globalization of the modern stock market. It means that the number of stocks that significantly affect the behavior of the others was consistently increasing during the considered time period, and it is possible to derive some regularities in the structure of the market.

In order to define the pattern of connections between stocks, the concept of degree distribution defined above in the previous sections can be utilized. It turns out that the degree distribution of the market graph has a well-defined power-law structure [23].

In [22], the authors proposed to relate combinatorial properties of the market graph to some properties of the stock market. For instance, one can apply algorithms for finding cliques or quasi-cliques in the market graph and relate them to groups of highly correlated stocks (the market graph should be constructed using a relatively

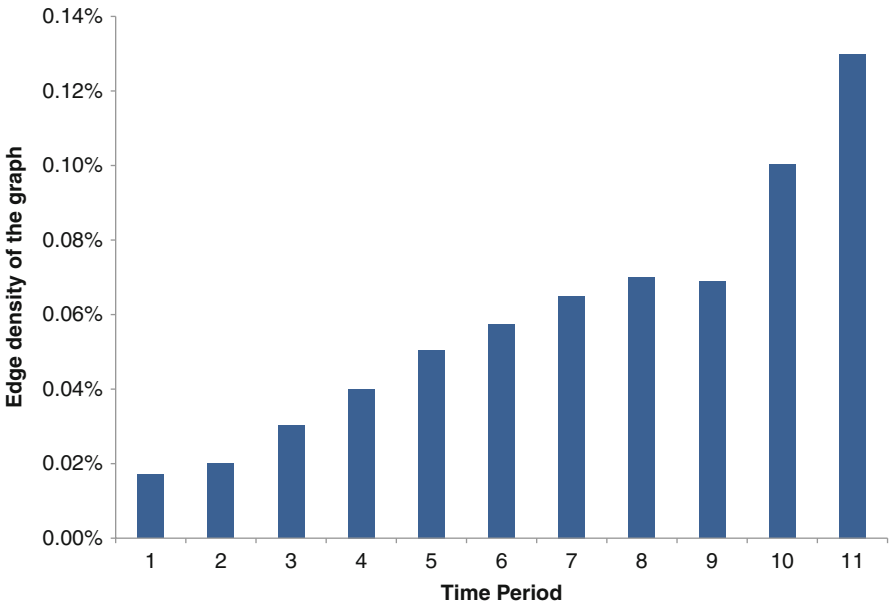

Fig. 22 Evolution of the edge density of the market graph during 2000–2002

large value of θ). A clique and quasi-clique (as defined in Sects. 3.2 and 3.4) of the graph are subsets of the this graph which are complete or "almost" complete graphs itself, respectively (i.e., has all possible edges or needed edge density is satisfied). Also, one can define the minimum clique partition problem for the market graphs, which solution (the minimum number of distinct cliques that the market graph) would represent the minimum number of clusters in the partition of the set of financial instruments.

It is worth mentioning that even though the problem of finding maximum clique in the graph is NP-hard, the exact solutions of this problem were found for different instances of the market graph, which was possible because the market graph is clustered (i.e., it contains dense groups of connected nodes).

Recall from Sect. 3.2 that an independent set is a subset of nodes which are not connected with any other node in this set. Therefore, this notion can be utilized for the market graphs in order to partition these graphs into clusters with stocks whose price fluctuations are not correlated or negatively correlated. It is rather important because finding independent sets in market graphs gives a method of choosing fully diversified portfolios. Partitioning the market graph in such a way that the number of clusters containing distinct diversified portfolios is minimal would represent the optimization problem of partitioning this graph to minimal number of independent sets. The optimal solution to this problem is a chromatic number corresponding to the graph coloring problem. As it was mentioned above, modern stock markets are usually highly correlated, so it is

much more likely to observe a relatively large cliques and quasi-cliques which represent correlated sets of the stocks in the market graph than a large independent sets which represent diversified portfolios in the modern stock market. These results are interesting since they support the idea of the globalization of the stock market.

Overall, finding cliques (as well as quasi-cliques, k-plexes, and other clique relaxations) and independent sets, along with the extensions similar to clique relaxations, in the market graph provides an efficient tool of performing data mining based on the stock market data, that is, partitioning the set of stocks into clusters of "similar" or "dissimilar" objects.

4.6 Social Networks

The process of formation of new communities, their integration and disintegration, attracting new members and their interaction with each other, and growth of the network and its progress, is among the most important research problems in social sciences. Political, religious, professional, and other forms of social organizations provide examples of such communities. A social network is a structure of people related directly or indirectly to each other through a common relation or interest. Therefore, it is very natural to represent these systems as graphs: the nodes are the people, groups, organizations, or other social entities and links represent relationships or flows between the nodes.

A number of recent studies have focused on the statistical properties of social networks. Researchers have concentrated particularly on a few properties that seem to be common to many social networks: the small-world property, power-law degree distribution (which was mentioned earlier in this chapter), and network transitivity. "Small-world effect" is the name given to the finding that the average distance between vertices in a network is short, usually scaling logarithmically with the total number of vertices. A characteristic that many networks have in common is clustering, or network transitivity, which is the property that two vertices that are both neighbors of the same third vertex have a higher probability of also being neighbors of each other. In the language of social networks, two of one's friends will have a greater probability of knowing each other than two people chosen at random from the population. Another important property that appears to be common in many networks is the property of community structure. Consider the case of social networks of friendships or other acquaintances between individuals. It is intuitively clear that such networks would contain "communities": subsets of vertices, within which vertex-to-vertex connections are dense, but between which connections are not as dense. A visual sketch of a network with such a community structure is shown in Fig. 23. The ability to detect community structures in a network clearly has practical applications. Detection of community structures can be performed using different approaches. A traditional method is hierarchical clustering that constructs a measure that tells one which edges are most central to communities. An alternative approach to the detection of communities is focused on the edges

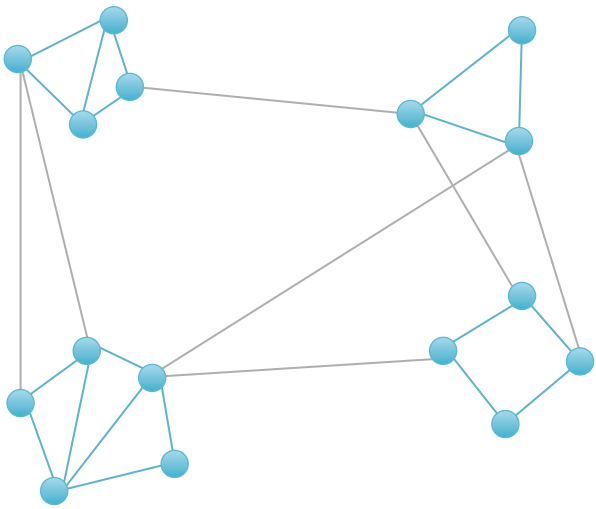

Fig. 23 A representation of a network with community structure. In this network, there are four communities of densely connected nodes (*circles* with *solid blue lines*), with a much lower density of connections (*gray lines*) between them

that are most "between" communities and strongly connected cores of communities (see [45]). In addition, the aforementioned concepts of clique relaxations (k-cliques, k-clubs, *k-plexes*) have their origins in the context of communities (tightly knit clusters) in social networks; therefore, efficient combinatorial algorithms for identifying these structures in networks have significant implications in this important application area.

Nowadays, social online communities are growing extensively, and the number of people using such services is increasing very fast; therefore, the necessity of social science research and related data mining techniques is unquestionable. More details on this emerging application of network-based data mining techniques can be found in [62].

5 Conclusion

Network-based data mining and related combinatorial optimization techniques are promising in a variety of applications, which have been demonstrated by multiple recent studies. In this chapter, several graph-theoretic concepts and problems have been discussed in the context of their interpretation from both data mining and combinatorial optimization perspectives. In addition, important real-world application areas have been briefly described, including biological/medical, chemical, telecommunication, financial, and social networks. Clearly, the research in this broad area is far from complete, and new methodological and application aspects will likely continue to be developed in the near future.

Cross-References

▸Combinatorial Optimization in Data Mining
▸Graph Theoretic Clique Relaxations and Applications
▸Small World Networks in Computational Neuroscience

Recommended Reading

1. J. Abello, P.M. Pardalos, M.G.C. Resende, On maximum clique problems in very large graphs, in *External Memory Algorithms* (American Mathematical Society, Providence, 1999), pp. 119–130
2. J. Abello, P.M. Pardalos, M.G.C. Resende (eds.), *Handbook of Massive Data Sets* (Kluwer, Dordrecht, 2002)
3. J. Abello, M.G.C. Resende, S. Sudarsky, Massive quasi-clique detection, in *LATIN 2002: Theoretical Informatics*. Lecture Notes in Computer Science (Springer, Berlin/New York, 2002), pp. 598–612
4. W. Aiello, F. Chung, L. Lu, A random graph model for power law graphs. Exp. Math. **10**, 53–66 (2001)
5. W. Aiello, F. Chung, L. Lu, Random evolution in massive graphs, in *Handbook on Massive Data Sets*, ed. by J. Abello, P. Pardalos, M. Resende (Kluwer, Dordrecht, 2002)
6. T. Akutsu, S. Kuhara, O. Maruyama, Identification of gene regulatory networks by strategic gene disruptions and gene overexpressions, in *Proceedings of the 9th Annual ACM-SIAM Symposium Discrete Algorithms* (SODA 1998), San Francisco, CA, 1998, pp. 695–702
7. R.D. Alba, A graph-theoretic definition of a sociometric clique. J. Math. Sociol. **6**, 113–26 (1973)
8. R. Albert, A.-L. Barabasi, Statistical mechanics of complex networks. Rev. Mod. Phys. **74**, 47–97 (2002)
9. R. Albert, H. Jeong, A.-L. Barabási, Diameter of the World-Wide Web. Nature **401**, 130–131 (1999)
10. D. Alderson, Catching the "Network Science" bug: insight and opportunity for the operations researcher. Oper. Res. **56**, 1047–1065 (2008)
11. S. Arora, S. Safra, Approximating clique is NP-complete, in *Proceedings of the 33rd IEEE Symposium on Foundations on Computer Science*, Pittsburg, PA, 24–27 Oct 1992, pp. 2–13
12. C.-A. Azencott, A. Ksikes, S.J. Swamidass, J.H. Chen, L. Ralaivola, P. Baldi, One- to four-dimensional kernels for virtual screening and the prediction of physical, chemical, and biological properties. J. Chem. Inf. Model. **47**, 965–974 (2007)
13. B. Balasundaram, S. Butenko, Network clustering, in *Analysis of Biological Networks*, ed. by B.H. Junker, F. Schreiber (Wiley, Hoboken, 2008), pp. 113–138
14. B. Balasundaram, S. Butenko, S. Trukhanov, Novel approaches for analyzing biological networks. J. Comb. Optim. **10**, 23–39 (2005)
15. B. Balasundaram, S. Butenko, I. Hicks, Clique relaxations in social network analysis: the maximum k-plex problem. Oper. Res. **59**(1), 133–142 (2011)
16. A.-L. Barabasi, *Linked* (Perseus Publishing, New York, 2002)
17. A.-L. Barabasi, R. Albert, Emergence of scaling in random networks. Science **286**, 509–511 (1999)
18. A.L. Barabasi, N. Gulbahce, J. Loscalzo, Network medicine: a network-based approach to human disease. Nat. Rev. Genet. **12**(1), 56–68 (2011)
19. J. Berg, M. Lassig, Local graph alignment and motif search in biological networks. Natl. Acad. Sci. USA **101**(41), 14689–14694 (2004)

20. V. Boginski, S. Butenko, P.M. Pardalos, Network-based techniques in the analysis of the stock market, in *Supply Chain and Finance* (World Scientific, Singapore, 2003), pp. 1–14
21. V. Boginski, S. Butenko, P.M. Pardalos, Modeling and optimization in massive graphs, in *Novel Approaches to Hard Discrete Optimization*, ed. by P.M. Pardalos, H. Wolkowicz (American Mathematical Society, Providence, 2003), pp. 17–39
22. V. Boginski, S. Butenko, P.M. Pardalos, On structural properties of the market graph, in *Innovations in Financial and Economic Networks*, ed. by A. Nagurney (Edward Elgar, Cheltenham/Northampton, 2003), pp. 29–45
23. V. Boginski, S. Butenko, P.M. Pardalos, Network models of massive datasets. Comput. Sci. Inf. Syst. **1**, 79–93 (2004)
24. V. Boginski, S. Butenko, P.M. Pardalos, Statistical analysis of financial networks. Comput. Stat. Data Anal. **48**(2), 431–443 (2005)
25. V. Boginski, S. Butenko, P.M. Pardalos, Mining market data: a network approach. Comput. Oper. Res. **33**, 3171–3184 (2006)
26. I.M. Bomze, M. Budinich, P.M. Pardalos, M. Pelillo, The maximum clique problem, in *Handbook of Combinatorial Optimization*, ed. by D.-Z. Du, P.M. Pardalos (Kluwer, Dordrecht, 1999), pp. 1–74
27. P.S. Bradley, U.M. Fayyad, O.L. Mangasarian, Mathematical programming for data mining: formulations and challenges. INFORMS J. Comput. **11**(3), 217–238 (1999)
28. S. Brin, L. Page, The anatomy of a large scale hypertextual web search engine, in *Proceedings of the 7th World Wide Web Conference*, Brisbane, Australia, 1998, pp. 107–117
29. A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, J. Wiener, Graph structure in the Web. Comput. Netw. **33**, 309–320 (2000)
30. S. Butenko, W.A. Chaovalitwongse, P.M. Pardalos, Clustering challenges in biological networks (World Scientific, New Jersry, 2009)
31. F. Chung, L. Lu, *Complex Graphs and Networks*. CBMS Lecture Series (American Mathematical Society, Providence, 2006)
32. D.J. Cook, L.B. Holder, Graph-based data mining. IEEE Intell. Syst. **15**(2), 32–41 (2000)
33. T. Dandekar, S. Schuster, B. Snel, Pathway alignment: application to the comparative analysis of glycolytic enzymes. Biochem. J. **343**, 115–124 (1999)
34. J.C. Doyle, D.L. Alderson, L. Li, S. Low, M. Roughan, S. Shalunov, R. Tanaka, W. Willinger, The "robust yet fragile" nature of the internet. Proc. Natl. Acad. Sci. **102**(41), 14497–14502 (2005)
35. R. Durbin, S.R. Eddy, A. Krogh, G. Mitchison, *Biological Sequence Analysis* (Cambridge University Press, Cambridge, 1998)
36. V.M. Eguiluz, D.R. Chialvo, G. Cecchi, M. Baliki, A.V. Apkarian, Scale-free structure of brain functional networks. Phys. Rev. Lett. **94**, 018102 (2005)
37. D.J. Felleman, D.C. Van Essen, Distributed hierarchical processing in the primate cerebral cortex. Cereb. Cortex **1**, 1–47 (1991)
38. T.A. Feo, M.G.C. Resende, A greedy randomized adaptive search procedure for maximum independent set. Oper. Res. **42**, 860–878 (1994)
39. T.A. Feo, M.G.C. Resende, Greedy randomized adaptive search procedures. J. Glob. Optim. **6**, 109–133 (1995)
40. P. Finn, S. Muggleton, D. Page, A. Srinivasan, Phannacophore discovery using the inductive logic programming system Progol. Mach. Learn. **30**, 241–271 (1998)
41. I. Fischer, T. Meinl, Graph based molecular data mining – an overview, in *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics* (IEEE, Piscataway, 2004), pp. 4578–4582
42. I. Fischer, T. Meinl, Graph based molecular data mining – an overview, in *IEEE International Conference on Systems, Man and Cybernetics*, The Hague, The Netherlands, 2004
43. M.R. Garey, D.S. Johnson, The complexity of near-optimal coloring. J. ACM **23**, 43–49 (1976)
44. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness* (Freeman, New York, 1979)
45. M. Girvan, M.E.J. Newman, Community structure in social and biological networks. Natl. Acad. Sci. **99**, 7821–7826 (2002)

46. J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$. Acta Math. **182**, 105–142 (1999)
47. B. Hayes, Graph theory in practice. Am. Sci. **88**, 9–13 (Part I), 104–109 (Part II) (2000)
48. C.C. Hilgetag, R. Kotter, K.E. Stephen, O. Sporns, Computational methods for the analysis of brain connectivity, in *Computational Neuroanatomy* (Humana, Totowa, 2002)
49. R.A. Jarvis, E.A. Patrick, Clustering using a similarity measure based on shared nearest neighbors. IEEE Trans. Comput. **C-22**(11), 1025–1034 (1973)
50. D. Jiang, J. Pei, Mining frequent cross-graph quasi-cliques. ACM Trans. Knowl. Discov. Data **2**(4), 1–42 (2009)
51. D. Jiang, C. Tang, A. Zhang, Cluster analysis for gene expression data: a survey. **16**(11), 1370–1386 (2004)
52. H. Kawaji, Y. Yamaguchi, H. Matsuda, A. Hashimoto, A graph based clustering method for a large set of sequences using a graph partitioning algorithm. Genome Inform. **12**, 93–102 (2001)
53. H. Kawaji, Y. Takenaka, H. Matsuda, Graph-based clustering for finding distant relationships in a large set of protein sequences. Bioinformatics **20**(2), 243–252 (2004)
54. B.P. Kelley, R. Sharan, R.M. Karp, Conserved pathways within bacteria and yeast as revealed by global protein network alignment. Proc. Natl. Acad. Sci. USA **100**, 11394–11399 (2003)
55. M.G. Kendall, *Rank Correlation Methods* (Griffin, Oxford, 1948)
56. J. Kleinberg. Authoritative sources in a hyperlinked environment. J. ACM **46**, 604–632 (1999)
57. E.V. Koonin, R.L. Tatusov, K.E. Rudd, Sequence similarity analysis of Escherichia coli proteins: functional and evolutionary implications. Proc. Natl Acad. Sci. USA **92**, 11921–11925 (1995)
58. L. Laloux, P. Cizeau, J.-P. Bouchad, M. Potters, Noise dressing of financial correlation matrices. Phys. Rev. Lett. **83**(7), 1467–1470 (1999)
59. R.D. Luce, Connectivity and generalized cliques in sociometric group structure. Psychometrika **15**, 169–190 (1950)
60. R.N. Mantegna, H.E. Stanley, *An Introduction to Econophysics: Correlations and Complexity in Finance* (Cambridge University Press, Cambridge/New York, 2000)
61. H. Matsuda, T. Ishihara, A. Hashimoto, Classifying molecular sequences using a linkage graph with their pairwise similarities. Theor. Comput. Sci. **210**, 305–325 (1999)
62. N. Memon, J.J. Xu, L.L. Hicks, H. Chen, *Data Mining for Social Network Data* (Springer, New York/London, 2010)
63. A. Mendelzon, P. Wood, Finding regular simple paths in graph databases. SIAM J. Comput. **24**, 1235–1258 (1995)
64. A. Mendelzon, G. Mihaila, T. Milo, Querying the World Wide Web. J. Digit. Libr. **1**, 68–88 (1997)
65. Miniwatts Marketing Group, Internet growth statistics (2008), http://www.internetworldstats.com/emarketing.htm
66. R.J. Mokken, Cliques, clubs and clans. Qual. Quant. **13**, 161–173 (1979)
67. J.M. Murre, D.P. Sturdy, The connectivity of the brain: multi-level quantitative analysis. Biol. Cybern. **73**, 529–545 (1995)
68. S. Nijssen, J.N. Kok, A quickstart in frequent structure mining can make a difference. LIACS, Leiden University, The Netherlands, Tech. Rep., April 2004
69. H. Ogata, W. Fujibuchi, S. Goto, A heuristic graph comparison algorithm and its application to detect functionally related enzyme clusters. Nucleic Acids Res. **28**, 4021–4028 (2000)
70. P.M. Pardalos, J. Xue, The maximum clique problem. J. Glob. Optim. 301–328 (1994)
71. P.M. Pardalos, T. Mavridou, J. Xue, The graph coloring problem: a bibliographic survey, in *Handbook of Combinatorial Optimization*, vol. 2, ed. by D.-Z. Du, P.M. Pardalos (Kluwer, Dodrecht, 1998), pp. 331–395
72. J. Pattillo, A. Veremyev, S. Butenko, V. Boginski, On the maximum quasi-clique problem. Discrete Appl. Math. 161(1–2), 244–257 (2013) doi: 10.1016/j.dam.2012.07.019, http://www.sciencedirect.com/science/article/pii/S0166218X12002843
73. J. Pei, D. Jiang, A. Zhang, Mining cross-graph quasi-cliques in gene expression and protein interaction data, in *Proceedings of the 21st International Conference on Data Engineering*, Tokyo, 2005, pp. 353–354

74. V. Plerou, P. Gopikrishnan, B. Rosenow, L.A.N. Amaral, H.E. Stanley, Universal and nonuniversal properties of cross correlations in financial time series. Phys. Rev. Lett. **83**(7), 1471–1474 (1999)
75. O.A. Prokopyev, V. Boginski, W. Chaovalitwongse, P.M. Pardalos, J.C. Sackellares, P.R. Carney, Network-based techniques in EEG data analysis and epileptic brain modeling, in *Data Mining in Biomedicine*, ed. by P.M. Pardalos, V. Boginski, A. Vazacopoulos (Springer, New York, 2007), pp. 559–573
76. L. Ralaivolaa, J.S. Swamidassa, H. Saigoa, P. Baldi, Graph kernels for chemical informatics. Neural Netw. **18**, 1093–1110 (2005)
77. J. Scott, T. Ideker, R.M. Karp, R. Sharan, Efficient algorithms for detecting signaling pathways in protein interaction networks. J. Comput. Biol. **13**, 133–144 (2006)
78. S.B. Seidman, B.L. Foster, A graph theoretic generalization of the clique concept. J. Math. Sociol. **6**, 139–154 (1978)
79. C. Spearman, The proof and measurement of association between two things. Am. J. Psychol. **15**(1), 72–101 (1904)
80. M. Steffen, A. Petti, J. Aach, Automated modelling of signal transduction networks. BMC Bioinform. **3**, 34 (2002)
81. P.-N. Tan, M. Steingach, V. Kumar, *Introduction to Data Mining* (Addison-Wesley, Boston, 2006)
82. A. Veremyev, V. Boginski, Identifying large robust network clusters via new compact formulations of maximum k-club problems. Eur. J. Obstet. Gyn. R. B. 218(2), 316–326 (2012)
83. N. Wale, X. Ning, G. Karypis, Trends in chemical graph data mining, in *Managing and Mining Graph Data* (Springer, New York, 2010), pp. 581–606
84. T. Washio, H. Motoda, State of the art of graph-based data mining. SIGKDD Explor. Newsl. **5**(1), 59–68 (2003)
85. W. Willinger, D. Alderson, J.C. Doyle, Mathematics and the internet: a source of enormous confusion and great potential. Not. Am. Math. Soc. **56**(5), 286–299 (2009)
86. Q. Yang, S.-H. Sze, Path matching and graph matching in biological networks. J. Comput. Biol. **14**(1), 56–67 (2007)
87. S.-H. Yook, Z.N. Oltvai, A.-L. Barabasi, Functional and topological characterization of protein interaction networks. Proteomics **4**, 928–942 (2004)
88. Z. Zeng, J. Wang, L. Zhou, G. Karypis, Coherent closed quasi-clique discovery from large dense graph databases, in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, 2006, pp. 797–802