1

A SURVEY OF COMPUTATIONAL APPROACHES TO RECONSTRUCT AND PARTITION BIOLOGICAL NETWORKS

LIPI ACHARYA, THAIR JUDEH, AND DONGXIAO ZHU

> "Everything is deeply intertwingled"
> *Theodor Holm Nelson*

1.1 INTRODUCTION

The above quote by Theodor Holm Nelson, the pioneer of information technology, states a deep interconnectedness among the myriad topics of this world. The biological systems are no exceptions, which comprise of a complex web of biomolecular interactions and regulation processes. In particular, the field of computational systems biology aims to arrive at a theory that reveals complicated interaction patterns in the living organisms, which result in various biological phenomenon. Recognition of such patterns can provide insights into the biomolecular activities, which pose several challenges to biology and genetics. However, complexity of biological systems and often an insufficient amount of data used to capture these activities make a reliable inference of the underlying network topology as well as characterization of various patterns underlying these topologies, very difficult. As a result, two problems that have received a considerable amount of attention among researchers are (1) reverse engineering of biological networks from genome-wide measurements and (2) inference of functional units in large biological networks (Fig 1.1).

Statistical and Machine Learning Approaches for Network Analysis, Edited by Matthias Dehmer and Subhash C. Basak.

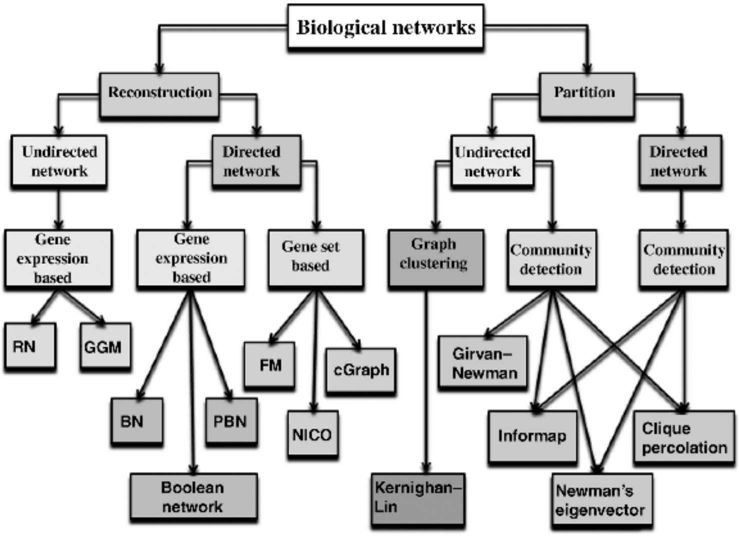


FIGURE 1.1 Approaches addressing two fundamental problems in computational systems biology (1) reconstruction of biological networks from two complementary forms of data resources, gene expression data and gene sets and (2) partitioning of large biological networks to extract functional units. Two classes of problems in network partitioning are graph clustering and community detection.

Rapid advances in high-throughput technologies have brought about a revolution in our understanding of biomolecular interaction mechanisms. A reliable inference of these mechanisms directly relates to the measurements used in the inference procedure. High throughput molecular profiling technologies, such as microarrays and second-generation sequencing, have enabled a systematic study of biomolecular activities by generating an enormous amount of genome-wide measurements, which continue to accumulate in numerous databases. Indeed, simultaneous profiling of expression levels of tens of thousands of genes allows for large-scale quantitative experiments. This has resulted in substantial interest among researchers in the development of novel algorithms to reliably infer the underlying network topology using gene expression data. However, gaining biological insights from large-scale gene expression data is very challenging due to the *curse of dimensionality*. Correspondingly, a number of computational and experimental methods have been developed to arrange genes in various groups or clusters, on the basis of certain similarity criterion. Thus, an initial characterization of large-scale gene expression data as well as conclusions derived from biological experiments result in the identification of several smaller components comprising of genes sharing similar biological properties. We refer to these components as *gene sets*. Availability of effective computational and experimental strategies have led to the emergence of gene sets as a completely new form of data for the reverse engineering of gene regulatory relationships. Gene set based approaches have gained more attention for their inherent ability to incorporate higher-order interaction mechanisms as opposed to individual genes.

There has been a sequence of computational efforts addressing the problem of network reconstruction from gene expression data and gene sets. Gaussian graphical models (GGMs) [1–3], probabilistic Boolean networks (PBNs) [4–7], Bayesian networks (BNs) [8,9], differential equation based [10,11] and mutual information networks such as relevance networks (RNs) [12,13], ARACNE [14], CLR [15], MRNET [16] are viable approaches capitalizing on the use of gene expression data, whereas collaborative graph model (cGraph) [17], frequency method (FM) [18], and network inference from cooccurrences (NICO) [19,20] are suitable for the reverse engineering of biological networks from gene sets.

After a biological network is reconstructed, it may be too broad or abstract of a representation for a particular biological process of interest. For example, given a specific signal transduction, only a part of the underlying network is activated as opposed to the entire network. A finer level of detail is needed. Furthermore, these parts may represent the functional units of a biological network. Thus, *partitioning* a biological network into different clusters or communities is of paramount importance.

Network partitioning is often associated with several challenges, which make the problem NP-hard [21]. Finding the optimal partitions of a given network is only feasible for small networks. Most algorithms heuristically attempt to find a *good* partitioning based on some chosen criteria. Algorithms are often suited to a specific problem domain. Two major classes of algorithms in network partitioning find their roots in computer science and sociology, respectively [22]. To avoid confusion, we will refer to the first class of algorithms as *graph clustering* algorithms and the second class of algorithms as *community detection* algorithms. For graph clustering algorithms, the relevant applications include very large-scale integration (VLSI) and distributing jobs on a parallel machine. The most famous algorithm in this domain is the Kernighan–Lin algorithm [23], which still finds use as a subroutine for various other algorithms. Other graph clustering algorithms include techniques based on spectral clustering [24]. Originally community detection algorithms focused on social networks in sociology. They now cover networks of interest to biologists, mathematicians, and physicists. Some popular community detection algorithms include Girvan–Newman algorithm [25], Newman's eigenvector method [21,22], clique percolation algorithm [26], and Infomap [27]. Additional community detection algorithms include methods based on spin models [28,29], mixture models [30], and label propagation [31].

Intuitively, reconstruction and partitioning of biological networks appear to be two completely opposite problems in that the former leads to an increase, whereas the latter results in a decrease of the dimension of a given structure. In fact, these problems are closely related and one leads to the foundation of the other. For instance, presence of hypothetical gene regulatory relationships in a reconstructed network provides a motivation for the detection of biologically meaningful functional modules of the network. On the other hand, prior to apply gene set based network reconstruction algorithms, a computational or experimental analysis is first needed to derive gene sets. In this chapter, we present a number of computational approaches to reconstruct biological networks from genome-wide measurements, and to partition large biological networks into subnetworks. We begin with an overview of directed and undirected networks, which naturally arise in biological systems. Next, we discuss about two complementary forms of genome-wide data, gene expression data and gene sets, both of which can be accommodated by existing network reconstruction algorithms. We describe the principal aspects of various approaches to reconstruct biological networks using gene expression data and gene sets, and discuss the pros and cons associated with each of them. Finally, we present some popular clustering and community algorithms used in network partitioning. The material on network reconstruction and partition is largely based on Refs. [2,3,6–8,13,17–20,32] and [21–23,25–27,33–36], respectively.

1.2 BIOLOGICAL NETWORKS

A network is a graph $G(V, E)$ defined in terms of a set of vertices V and a set of edges E. In case of biological networks, a vertex $v \in V$ is either a gene or protein encoded by an organism, and an edge $e \in E$ joining two vertices $v_1,\ v_2 \in V$ in the network represents biological properties connecting v_1 and v_2. A biological network can be directed or undirected depending on the biological relationship that used to join the pairs of vertices in the network. Both directed and undirected networks occur naturally in biological systems. Inference of these networks is a major challenge in systems biology. We briefly review two kinds of biological networks in the following sections.

1.2.1 Directed Networks

In directed networks, each edge is identified as an ordered pair of vertices. According to the Central Dogma of Molecular Biology, genetic information is encoded in double-stranded DNA. The information stored in DNA is transferred to single-stranded messenger RNA (mRNA) to direct protein synthesis [42]. Signal transduction is the primary mean to control the passage of biological information from DNA to mRNA with mRNA directing the synthesis of proteins. A signal transduction event is usually triggered by the binding of external ligands (e.g., cytokine and chemokine) to the transmembrane receptors. This binding results in a sequential activation of signal molecules, such as cytoplasmic protein kinase and nuclear transcription factors (TFs), to lead to a biological end-point function [42]. A signaling pathway is composed of a web of gene regulatory wiring in response to different extracellular stimulus. Thus, signaling pathways can be viewed as directed networks containing all genes (or proteins) of an organism as vertices. A directed edge represents the flow of information from one gene to another gene.

1.2.2 Undirected Networks

Undirected networks differ from directed networks in that the edges in such networks are undirected. In other words, an undirected network can be viewed as a directed network by considering an undirected pair of vertices (v_1, v_2) as two directed pairs (v_1, v_2) and (v_2, v_1). Some biological networks are better suited for an undirected representation. Protein–protein interaction (PPI) network is an undirected network, where each protein is considered as a vertex and the physical interaction between a pair of proteins is represented as an edge [43].

The past decade has witnessed a significant progress in the computational inference of biological networks. A variety of approaches in the form of network models and novel algorithms have been proposed to understand the structure of biological networks at both *global* and *local* level. While the grand challenge in a global approach is to provide an integrated view of the underlying biomolecular interaction mechanisms, a local approach focuses on identifying fundamental domains representing functional units of a biological network.

Both directed and undirected network models have been developed to reliably infer the biomolecular activities at a global level. As discussed above, directed networks represent an abstraction of gene regulatory mechanisms, while the physical interactions of genes are suitably modeled as undirected networks. Focus has also been on the computational inference of biomolecular activities by accommodating genome-wide data in diverse formats. In particular, gene set based approaches have gained attention in recent bioinformatics analysis [44,45]. Availability of a wide range of experimental and computational methods have identified coherent gene set compendiums [46]. Sophisticated tools now exist to statistically verify the biological significance of a particular gene set of interest [46–48]. An emerging trend in this field is to reconstruct signaling pathways by inferring the order of genes in gene sets [19,20]. There are several unique features associated with gene set based network inference approaches. In particular, such approaches do not rely on gene expression data for the reconstruction of underlying network.

The algorithms to understand biomolecular activities at the level of subnetworks have evolved over time. Community detection algorithms, in particular, originated with hierarchical partitioning algorithms that include the Girvan–Newman algorithm. Since these algorithms tend to produce a dendrogram as their final result, it is necessary to be able to rank the different partitions represented by the dendrogram. Modularity was introduced by Newman and Girvan to address this issue. Many methods have resulted with modularity at the core. More recently, though, it has been shown that modularity suffers from some drawbacks. While there have been some attempts to address these issues, newer methods continued to emerge such as Infomap. Research has also expanded to incorporate different types of biological networks and communities. Initially, only undirected and unweighted networks were the focus of study. Methods are now capable of dealing with both directed and weighted networks. Moreover, previous studies only concentrated on distinct communities that did not allow overlap. With the advent of the clique percolation method and other similar methods, overlapping communities are becoming increasingly popular. The aforementioned approaches have been used to identify the structural organization of a variety of biological networks including metabolic networks, PPI networks, and protein domain networks. Such networks have a power–law degree distribution and the quantitative signature of scale-free networks [49]. PPI networks, in particular, have been the subject of intense study in both bioinformatics and biology as protein interactions are fundamental for cellular processes [50].

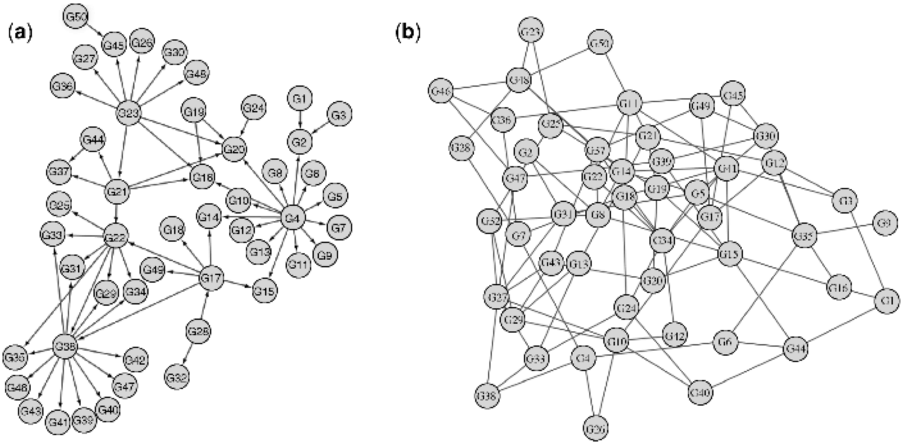


FIGURE 1.2 (a) Example of a directed network. The figure shows *Escherichia coli* gold standard network from the DREAM3 Network Challenges [37–39]. (b) Example of an undirected network. The figure shows an *in silico* gold standard network from the DREAM2 Network Challenges [40,41].

A common problem associated with the computational inference of a biological network is to assess the performance of the approach used in the inference procedure. It is quite assess as the structure of the true underlying biological network is unknown. As a result, one relies on biologically plausible simulated networks and data generated from such networks. A variety of *in silico* benchmark directed and undirected networks are provided by the dialogue for reverse engineering assessments and methods (DREAM) initiative to systematically evaluate the performance of reverse engineering methods, for example Refs. [37–41]. Figures 1.2 and 1.7 illustrate gold standard directed network, undirected network, and a network with community structure from the *in silico* network challenges in DREAM initiative.

1.3 GENOME-WIDE MEASUREMENTS

In this section, we present an overview of two complementary forms of data resources (Fig. 1.3), both of which have been utilized by the existing network reconstruction algorithms. The first resource is gene expression data, which is represented as matrix of gene expression levels. The second data resource is a gene set compendium. Each gene set in a compendium stands for a set of genes and the corresponding gene expression levels may or may not be available.

1.3.1 Gene Expression Data

Gene expression data is the most common form of data used in the computational inference of biological networks. It is represented as a matrix of numerical values,

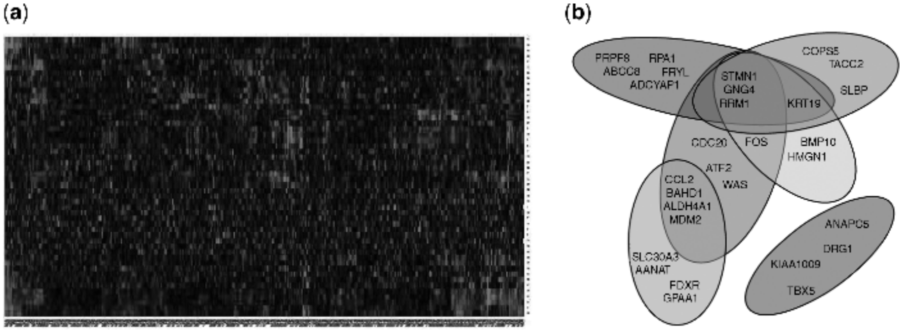


FIGURE 1.3 Two complementary forms of data accommodated by the existing network reconstruction algorithms. (a) Gene expression data generated from high-throughput platforms, for example, microarray. (b) Gene sets often resulted from explorative analysis of large-scale gene expression data, for example, cluster analysis.

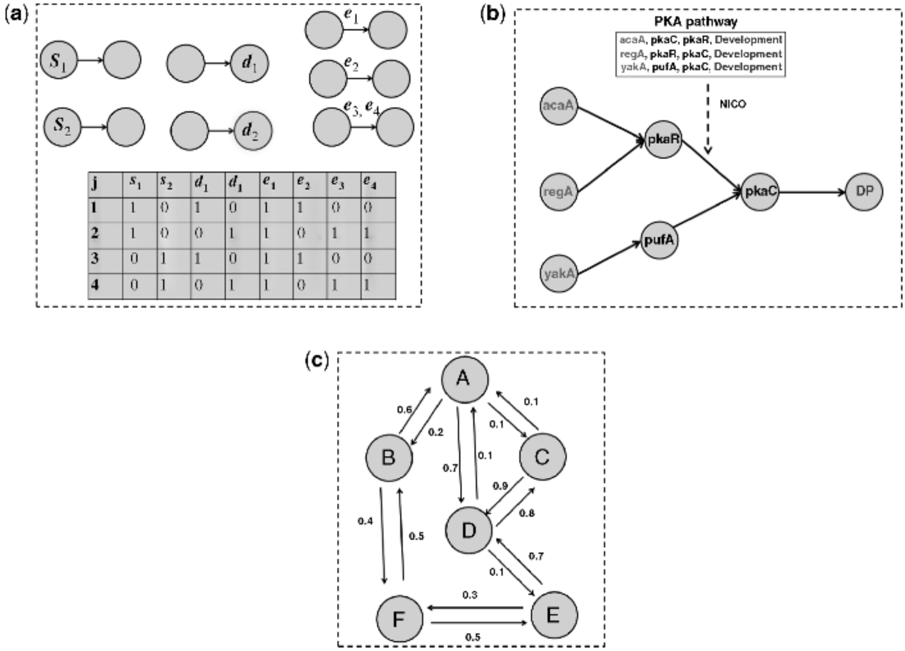
where each row corresponds to a gene, each column represents an experiment and each entry in the matrix stands for gene expression level. Gene expression profiling enables the measurement of expression levels of thousands of genes simultaneously and thus allows for a systematic study of biomolecular interaction mechanisms on genome scale. In the experimental procedure for gene expression profiling using microarray, typically a glass slide is spotted with oligonucleotides that correspond to specific gene coding regions. Purified RNA is labeled and hybridized to the slide. After washing, gene expression data is obtained by laser scanning. A wide range of microarray platforms have been developed to accomplish the goal of gene expression profiling. The measurements can be obtained either from conventional hybridization-based microarrays [51–53] or contemporary deep sequencing experiments [54,55]. Affymetrix GeneChip (www.affymetrix.com), Agilent Microarray (www.genomics.agilent.com), and Illumina BeadArray (www.illumina.com) are representative microarray platforms. Gene-expression data are accessible from several databases, for example, National Center for Biological Technology (NCBI) Gene Expression Omnibus (GEO) [56] and the European Molecular Biology Lab (EMBL) ArrayExpress [57].

1.3.2 Gene Sets

Gene sets are defined as sets of genes sharing biological similarities. Gene sets provide a rich source of data to infer underlying gene regulatory mechanisms as they are indicative of genes participating in the same biological process. It is impractical to collect a large number of samples from high-throughput platforms to accurately reflect the activities of thousands of genes. This poses challenges in gaining deep biological insights from genome-wide gene expression data. Consequently, experimental and computational methods are adopted to reduce the dimension of the space of variables [58]. Such characterizations lead to the discovery of clusters of genes or gene sets, consisting of genes which share similar biological functions. Some of the recent gene set based bioinformatics analyses include gene set enrichment analysis [46–48] and gene set based classification [44,45]. The major advantage of working with gene sets is their ability to naturally incorporate higher-order interaction patterns. In comparison to gene expression data, gene sets are more robust to noise and facilitate data integration from multiple sources. Computational inference of signaling pathways from gene sets, without assuming the availability of the corresponding gene expression levels, is an emerging area of research [17–20].

1.4 RECONSTRUCTION OF BIOLOGICAL NETWORKS

In this section, we describe some existing approaches to reconstruct directed and undirected biological networks from gene expression data and gene sets. To reconstruct directed networks from gene expression data, we present Boolean network, probabilistic Boolean network, and Bayesian network models. We discuss cGraph, frequency method and NICO approaches for network reconstruction using gene sets (Fig 1.4). Next, we present relevance networks and graphical Gaussian models for the reconstruction of undirected biological networks from gene expression data (Fig 1.5).



j	s_1	s_2	d_1	d_1	e_1	e_2	e_3	e_4
1	1	0	1	0	1	1	0	0
2	1	0	0	1	1	0	1	1
3	0	1	1	0	1	1	0	0
4	0	1	0	1	1	0	1	1

FIGURE 1.4 (a) Representation of inputs and Boolean data in the frequency method from Ref. [18]. (b) Network inference from PAK pathway [67] using NICO, in the presence of *a prior* known end points in each path [68]. (c) The building block of cGraph from Ref. [17].

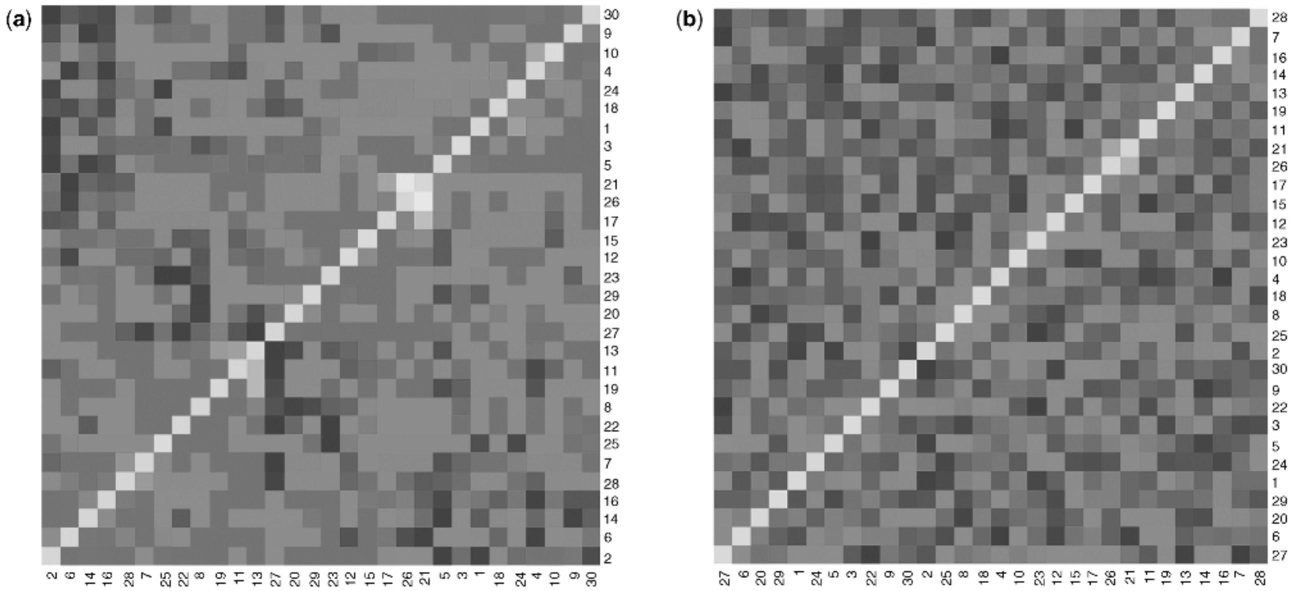


FIGURE 1.5 Comparison of correlation-based relevance networks (a) and partial correlation based graphical Gaussian modeling (b) performed on a synthetic data set generated from multivariate normal distribution. The figures represent estimated correlations and partial correlations between every pair of genes. Light to dark colors correspond to high to low correlations and partial correlations.

The review of models in case of directed and undirected networks is largely based on Refs. [6–8,17–20] and [2,3,13,32], respectively.

Although the aforementioned approaches for the reconstruction of directed networks have been developed for specific type of genome-wide measurements, they can be unified in case of binary discrete data. For instance, prior to infer a Boolean network, gene expression data is first discretized, for example, by assuming binary labels for each gene. Many Bayesian network approaches also assume the availability of gene expression data in a discretized form. On the other hand, a gene set compendium naturally corresponds to a binary discrete data set and is obtained by considering the presence or absence of genes in a gene set.

1.4.1 Reconstruction of Directed Networks

1.4.1.1 Boolean Networks

Boolean networks [4–6], present a simple model to reconstruct biological networks from gene expression data. In the model, a Boolean variable is associated with the state of a gene (ON or OFF). As a result, gene expression data is first discretized using binary labels. Boolean networks represent directed graphs, where gene regulatory relationships are inferred using boolean functions (AND, OR, NOT, NOR, NAND).

Mathematically, a Boolean network $G(V, F)$ is defined by a set of nodes $V = \{x_1, \ldots, x_n\}$ with each node representing a gene, and a set of logical Boolean functions $F = \{f_1, \ldots, f_n\}$ defining transition rules. We write $x_i = 1$ to denote that the ith gene is ON or expressed, whereas $x_i = 0$ means that it is OFF or not expressed. Boolean function f_i updates the state of x_i at time $t + 1$ using the binary states of other nodes at time t. States of all the genes are updated in a synchronous manner based on the transition rules associated with them, and this process is repeated.

Considering the complicated dynamics of biological networks, Boolean networks are inherently simple models which have been developed to study these dynamics. This is achieved by assigning Boolean states to each gene and employing Boolean functions to model rule-based dependencies between genes. By assuming only Boolean states for a gene, emphasis is given to the qualitative behavior of the network rather than quantitative information. The use of Boolean functions in modeling gene regulatory mechanisms leads to computational tractability even for a large network, which is often an issue associated with network reconstruction algorithms. Many biological phenomena, for example, cellular state dynamics, stability, and hysteresis, naturally fit into the framework of Boolean network models [59]. However, a major disadvantage of Boolean networks is their deterministic nature, resulting from a single Boolean function associated with a node. Moreover, the assumption of binary states for each gene may correspond to an oversimplification of gene regulatory mechanisms. Thus, Boolean networks are not a choice when the gene expression levels vary in a smooth continuous manner rather than two extreme levels, that is, "very high expression" and "very low expression." The transition rules in Boolean network models are derived from gene expression data. As gene expression data are noisy and often contain a larger number of genes than the number of samples, the inferred rules may not be reliable. This further contributes to an inaccurate inference of gene regulatory relationships.

1.4.1.2 Probabilistic Boolean Networks

To overcome the pitfalls associated with Boolean networks, probabilistic Boolean networks (PBNs) were introduced in Ref. [7] as their probabilistic generalization. PBNs extend Boolean networks by allowing for more than one possible Boolean function corresponding to each node, and offer a more flexible and enhanced network modeling framework.

In the underlying model presented in Ref. [7], every gene x_i is associated with a set of $l(i)$ functions

$$F_i = \left\{ f_1^{(i)}, \ldots, f_{l(i)}^{(i)} \right\}, \tag{1.1}$$

where each $f_j^{(i)}$ corresponds to a possible Boolean function determining the value of $x_i, i = 1, \ldots, n$. Clearly, Boolean networks follow as a particular case when $l(i) = 1$, for each $i = 1, \ldots, n$. The kth realization of PBN at a given time is defined in terms of vector functions belonging to $F_1 \times \ldots \times F_n$ as

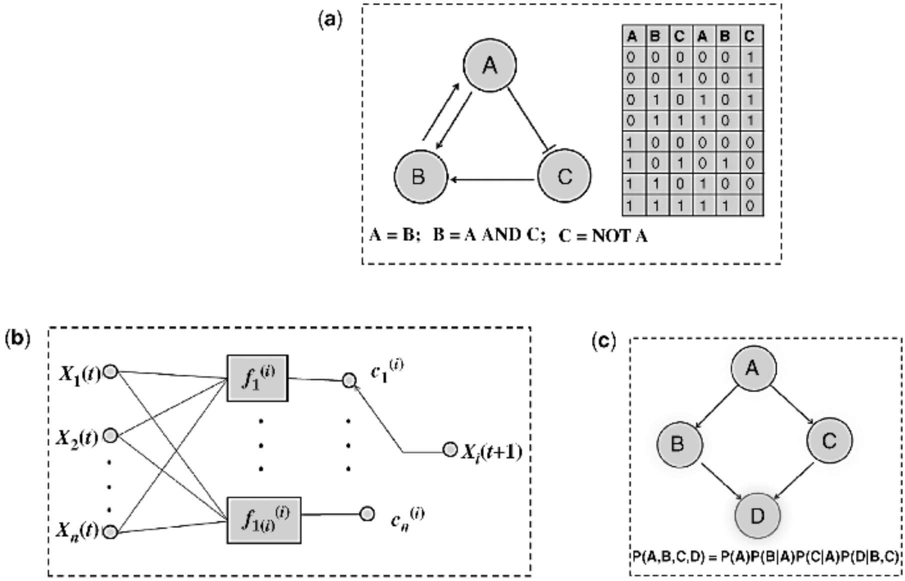
$$f_k = \left(f_{k_1}^{(1)}, \ldots, f_{k_n}^{(n)} \right), \tag{1.2}$$

where $1 \leq k_i \leq l(i)$, $f_{k_i}^{(i)} \in F_i$ and $i = 1, \ldots, n$. For a given $f = (f^{(1)}, \ldots, f^{(n)}) \in F_1 \times \ldots \times F_n$, the probability that jth function $f_j^{(i)}$ from F_i is employed in predicting the value of x_i, is given by

$$c_j^{(i)} = Pr\{f^{(i)} = f_j^{(i)}\} = \sum_{k: f_{k_i}^{(i)} = f_j^{(i)}} Pr\{f = f_k\}, \tag{1.3}$$

where $j = 1, \ldots, l(i)$ and $\sum_{j=1}^{l(i)} c_j^{(i)} = 1$. The basic building block of a PBN is presented in Figure 1.6. We refer to Ref. [7] for an extended study on PBNs.

It is clear that PBNs offer a more flexible setting to describe the transition rules in comparison to Boolean networks. This flexibility is achieved by associating a set of Boolean functions with each node, as opposed to a single Boolean function. In addition to inferring the rule-based dependencies as in the case of Boolean networks, PBNs also model for uncertainties by utilizing the probabilistic setting of Markov chains. By assigning multiple Boolean functions to a node, the risk associated with an inaccurate inference of a single Boolean function from gene expression data is greatly reduced. The design of PBNs facilitates the incorporation of prior knowledge. Although the complexity in case of PBNs increases from Boolean networks, PBNs are often associated with a manageable computational load. However, this is achieved at the cost of oversimplifying gene regulation mechanisms. As in the case of Boolean networks, PBNs may not be suitable to model gene regulations from smooth and continuous gene expression data. Discretization of such data sets may result in a significant amount of information loss.



A	B	C	A	B	C
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	0

FIGURE 1.6 Network reconstruction from gene expression data. (a) Example of a Boolean network with three genes from Ref. [60]. The figure displays the network as a graph, Boolean rules for state transitions and a table with all input and output states. (b) The basic building block of a probabilistic Boolean network from Ref. [7]. (c) A Bayesian network consisting of four nodes.

1.4.1.3 Bayesian Networks

Bayesian networks [8,9] are graphical models which represent probabilistic relationships between nodes. The structure of BNs embeds conditional dependencies and independencies, and efficiently encodes the joint probability distribution of all the nodes in the network. The relationships between nodes are modeled by a directed acyclic graph (DAG) in which vertices correspond to variables and directed edges between vertices represent their dependencies.

A BN is defined as a pair (G, Θ), where G represents a DAG whose nodes $X_1, X_2, \ldots, X_n$ are random variables, and Θ denotes the set of parameters that encode for each node in the network its conditional probability distribution (CPD), given that its parents are in the DAG. Thus, Θ comprises of the parameters

$$\theta_{x_i|Pa(x_i)} = Pr\{x_i|Pa(x_i)\}, \tag{1.4}$$

for each realization x_i of X_i conditioned on the set of parents $Pa(x_i)$ of x_i in G. The joint probability of all the variables is expressed as a product of conditional probabilities

$$Pr\{x_1, \ldots, x_n\} = \prod_{i=1}^{n} Pr\{x_i|Pa(x_i)\}. \tag{1.5}$$

The problem of learning a BN is to determine the BN structure B that best fits a given data set D. The fitting of a BN structure is measured by employing a *scoring function*. For instance, *Bayesian scoring* is used to find the optimal BN structure which maximizes the posterior probability distribution

$$P(B|D) = \frac{P(B, D)}{P(D)}. \tag{1.6}$$

Here, we define two Bayesian score functions *Bayesian Dirichlet (BD) score* from Ref. [61] and *K2 score* presented in Ref. [62].

BD score is defined as [61]

$$P(B, D) = P(B) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N_{ij} + N'_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + N'_{ijk})}{\Gamma(N'_{ijk})}, \tag{1.7}$$

where r_i represents the number of states of x_i, $q_i = \prod_{x_j \in Pa(x_i)} r_j$, N_{ijk} is the number of times x_i is in kth state and members in $Pa(x_i)$ are in jth state, $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$, $N_{ik} = \sum_{j=1}^{q_i} N_{ijk}$, N'_{ijk} are the parameters of Dirichlet prior distribution, $P(B)$ stands for the prior probability of the structure B and $\Gamma()$ represents the Gamma function.

The K2 score is given by [62]

$$P(B, D) = P(B) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{1.8}$$

We refer to Ref. [61,62] for further readings on Bayesian score functions.

BNs present an appealing probabilistic modeling approach to learn causal relationships and have been found to be useful for a significant number of applications. They can be considered as the best approach available for reasoning under uncertainty from noisy measurements, which prevent the over-fitting of data. The design of the underlying model facilitates the incorporation of prior knowledge and allows for an understanding of future events. However, a major disadvantage associated with BN modeling is that it requires large computational efforts to learn the underlying network structure. In many formulations learning a BN is an NP-hard problem, regardless of data size [63]. The number of different structures for a BN with n nodes, is given by the recursive formula

$$s(n) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} s(n-i) = n^{2^{O(n)}} \tag{1.9}$$

[62,64]. As $s(n)$ grows exponentially with n, learning the network structure by exhaustively searching over the space of all possible structures is infeasible even when n is small. Moreover, existence of equivalent networks presents obstacles in the inference of an optimal structure. BNs are inherently static in nature with no directed cycles. As a result, dynamic Bayesian networks (DBNs) have been developed to analyze time series data, which further pose computational challenges in structure learning.

Thus, a tractable inference via BNs relies on suboptimal heuristic search algorithms. Some of the popular approaches include K2 [62] and MCMC [65], which have been implemented in the Bayes Net Tool Box [66].

1.4.1.4 Collaborative Graph Model

As opposed to gene expression data, the collaborative graph or cGraph model [17] utilizes gene sets to reconstruct the underlying network structure. It presents a simple model by employing a directed weighted graph to infer gene regulatory mechanisms.

Let V denote the set of all distinct genes among gene sets. In the underlying model for cGraph [17], the weight W_{xy} of an edge from a gene x to another gene y satisfies

$$0 \leq W_{xy} \leq 1 \tag{1.10}$$

and

$$\sum_{y \in V, y \neq x} W_{xy} = 1. \tag{1.11}$$

Correspondingly, the weight matrix W can be interpreted as a transition probability matrix used in the theory of Markov chains. For network reconstruction, cGraph uses weighted counts of every pair of genes that appear among gene sets to approximate the weights of edges. Weight W_{xy} can be interpreted as $P(y|x)$, which is the probability of randomly selecting a gene set S containing gene x followed by randomly choosing y as a second gene in the set. Assuming that both, the gene set containing gene x and y were chosen uniformly, weights are approximated as

$$W_{xy} = \hat{P}(y|x) = \frac{\sum_{S:\{x,y\} \subset S} (\frac{1}{|S|-1})}{\sum_{S:x \in S} 1}. \tag{1.12}$$

Overall, cGraph is an inherently simple model, where a weighted edge measures the strength of a gene's connection with other genes. It is easy to understand, achievable at a manageable computational cost and appropriate for modeling pair wise relationships. However, cGraph adds a weighted edge between every pair of genes that appear together in some gene set and so the networks inferred by cGraph typically contain a large number of false positives and many interpretable functional modules.

1.4.1.5 Frequency Method

The frequency method presented in Ref. [18] reconstructs a directed network from a list of unordered gene sets. It estimates an ordering for each gene set by assuming

- tree structures in the paths corresponding to gene sets
- *a prior* availability of source and destination nodes in each gene set
- *a prior* availability of directed edges used to form a tree in each gene set, but not the order in which these edges appear in the tree.

Following the approach presented in Ref. [18], let us denote the set of source nodes, target nodes, and the collection of all directed edges involved in the network by S, T, and E, respectively. Each $l \in S \cup T \cup E$ can be associated with a binary vector of length N by considering $x_l(j) = 1$, if l is involved with the jth gene set, where N is the total number of gene sets. Let s_j be the source and d_j be the destination node in the jth gene set. To estimate the order of genes in the jth gene set, FM identifies e^* satisfying

$$e^* = \arg\max_{e \in E} \lambda_j(e), \tag{1.13}$$

where the score $\lambda_j(e)$ is defined as

$$\lambda_j(e) = x_{s_j}^T x_e - x_{d_j}^T x_e, \tag{1.14}$$

for each $e \in E$ with $x_e(j) = 1$. Note that $\lambda_j(e)$ determines whether e is closer to s_j than it is to d_j. The edge e^* is placed closest to s_j. The edge corresponding to the next largest score follows e^*. The procedure is repeated until all edges are in order [18].

FM is computationally efficient and leads to a unique solution of the network inference problem. However, the model makes strong assumptions of the availability of source and target genes in each gene set as well as directed edges involved in the corresponding path. Considering the real-world scenarios, it is not practical to assume the availability of such gene set compendiums. The underlying assumptions in FM make it inherently deterministic in nature. Moreover, FM is subject to failure in the presence of multiple paths between the same pair of genes.

1.4.1.6 EM-Based Inference from Gene Sets

We now describe a more general approach from Refs. [19,20] to network reconstruction from gene sets. It is termed as network inference from co-occurrences or NICO. Developed under the expectation–maximization (EM) framework, NICO infers the structure of the underlying network topology by assuming the order of genes in each gene set as missing information.

In NICO [19,20], signaling pathways are viewed as a collection of T-independent samples of first-order Markov chain, denoted as

$$Y = \left\{y^{(1)}, \ldots, y^{(T)}\right\}. \tag{1.15}$$

It is well known that Markov chain depends on an initial probability vector π and a transition matrix A. NICO treats the unobserved permutations $\{\tau^{(1)}, \ldots, \tau^{(T)}\}$ of $\{y^{(1)}, \ldots, y^{(T)}\}$ as hidden variables and computes the maximum-likelihood estimates of the parameters π and A via an EM algorithm. The E-step estimates expected permutations for each path conditioned on the current estimate of parameters, and the M-step updates the parameter estimates.

Let $x^{(m)}$ denote a path with N_m elements. NICO models r_m as a random permutation matrix drawn uniformly from the collection Ψ_{N_m} of all permutations of N_m elements.

In particular, the E-step computes the sufficient statistics

$$\overline{\alpha}^{(m)}_{t',\, t''} = \mathbb{E}\left[\sum_{t=2}^{N_m} r^{(m)}_{t,t'} r^{(m)}_{t-1,\, t''} | x^{(m)}, \hat{A}, \hat{\pi}\right] = \frac{\sum_{r\in\Psi_{N_m}} r_{t,t'} r_{t-1,\, t''} P\left[x^{(m)}|r, \hat{A}, \hat{\pi}\right]}{\sum_{r\in\Psi_{N_m}} P\left[x^{(m)}|r, \hat{A}, \hat{\pi}\right]} \tag{1.16}$$

and

$$\overline{r}^{(m)}_{1,\, t'} = \mathbb{E}\left[r^{(m)}_{1,t'}|x^{(m)}, \hat{A}, \hat{\pi}\right] = \frac{\sum_{r\in\Psi_{N_m}} r_{1,\, t'} P\left[x^{(m)}|r, \hat{A}, \hat{\pi}\right]}{\sum_{r\in\Psi_{N_m}} P\left[x^{(m)}|r, \hat{A}, \hat{\pi}\right]}, \tag{1.17}$$

where $P[x^{(m)}|r, \hat{A}, \hat{\pi}]$ is computed as

$$P\left[x^{(m)}|r, \hat{A}, \hat{\pi}\right] = P\left[y^{(m)}|\tau, \hat{A}, \hat{\pi}\right] = \hat{\pi}_{y^{(m)}_{\tau_1}} \prod_{t=2}^{N_m} \hat{A}_{y^{(m)}_{\tau_{t-1}} y^{(m)}_{\tau_t}}. \tag{1.18}$$

The M-step updates the parameters using the closed form expressions

$$(\hat{A}_{i,\, j})_{\text{new}} = \frac{\sum_{m=1}^{T} \sum_{t',t''=1}^{N_m} \overline{\alpha}^{(m)}_{t',t''} x^{(m)}_{t'',i} x^{(m)}_{t',j}}{\sum_{j=1}^{|S|} \sum_{m=1}^{T} \sum_{t',t''=1}^{N_m} \overline{\alpha}^{(m)}_{t',t''} x^{(m)}_{t'',i} x^{(m)}_{t',j}} \tag{1.19}$$

and

$$(\hat{\pi}_i)_{\text{new}} = \frac{\sum_{m=1}^{T} \sum_{t'=1}^{N_m} \overline{r}^{(m)}_{1,t'} x^{(m)}_{t',i}}{\sum_{i=1}^{|S|} \sum_{m=1}^{T} \sum_{t'=1}^{N_m} \overline{r}^{(m)}_{1,t'} x^{(m)}_{t',i}}, \tag{1.20}$$

where $|S|$ is the total number of distinct genes among gene sets. We refer to Refs. [19,20], for additional theoretical details.

NICO presents an appealing approach to reconstruct the most likely signaling pathway from unordered gene sets. The mature EM framework provides a theoretical foundation for NICO. It is well known that gene expression data are often noisy and expensive. In order to infer the network topology, NICO purely relies on gene sets and does not require the corresponding gene expression measurements. As opposed to a single gene or a pair of genes, gene sets more naturally capture the higher-order interactions. These advantages make NICO a unique approach to infer signaling pathways directly from gene sets. However, NICO has a nontrivial computational complexity. For large networks, the combinatorial nature of the E-step makes the exact computation infeasible. Thus, an important sampling based approximation of the E-step has been proposed [19,20]. Moreover, NICO assumes a linear arrangement of genes in each gene set without any feedback loops and so it is not applicable in real-world scenarios where signaling pathways are interconnected and regulated via feedback loops.

1.4.2 Reconstruction of Undirected Networks

1.4.2.1 Relevance Networks

Relevance networks [13] are based on measuring the strength of pairwise associations among genes from gene expression data. The pairwise association is measured in terms of Pearson's correlation coefficient. Given two genes x and y, Pearson's correlation coefficient is defined as

$$\hat{\rho}(x, y) = \frac{\sum_{i=1}^{N}(a_i - \overline{a})(b_i - \overline{b})}{\sqrt{\sum_{i=1}^{N}(a_i - \overline{a})^2}\sqrt{\sum_{i=1}^{N}(b_i - \overline{b})^2}}, \tag{1.21}$$

where $x = (a_1, \ldots, a_N)$ and $y = (b_1, \ldots, b_N)$ represent the N-dimensional observations for x and y with means $\overline{a}$ and $\overline{b}$, respectively. There also exists an information theoretic version of RN's, where correlation is replaced with mutual information (MI) for each pair of genes. MI between x and y is defined as [12]

$$\mathrm{MI}(x, y) = E(x) + E(y) - E(x, y), \tag{1.22}$$

where E stands for the entropy of a gene expression pattern and is given by

$$E(x) = -\sum_{i=1}^{n} p(a_i)\log_2(p(a_i)). \tag{1.23}$$

For further readings on RN's, tools for their inference and comparison with other mutual information network inference approaches, we refer to Refs. [12,69–71].

In order to detect truly coexpressed gene pairs in an *ad-hoc* way, the calculated correlation values are compared with a predefined correlation cut-off value. If a calculated correlation value exceeds the cut-off value, the corresponding genes are connected by an undirected edge. We now present a more reliable two-stage approach from Ref. [32], which simultaneously controls the statistical and biological significance of the inferred network. We only consider the case of Pearson's correlation, however, the method can be extended to the case of Kendall correlation coefficient and partial correlation coefficients [32]. Assuming a total of M genes, we simultaneously test $\Lambda = \binom{M}{2}$ pairs of two-sided hypotheses

$$H_0 : S_{x_i,x_j} \leq \mathrm{cor}_{\min} \quad \text{versus} \quad H_\alpha : S_{x_i,x_j} > \mathrm{cor}_{\min}, \tag{1.24}$$

for each $i, j = 1, \ldots, M$ and $i \neq j$. Here, S is the measure of strength of co-expression (Pearson's correlation in this case) between gene pairs and $\mathrm{cor}_{\min}$ is the minimum acceptable strength of coexpression. The sample correlation coefficient $\hat{S}$ ($\hat{\rho}$ in this case) serves as a decision statistic to decide the pairwise dependency of two genes. For large sample size N, the per comparison error rate (PCER) p-values for pairwise correlation is computed as

$$p_{\rho(x_i,x_j)} = 2\left(1 - \Phi\left(\frac{\tanh^{-1}\hat{\rho}(x_i, x_j)}{(N-3)^{-1/2}}\right)\right), \tag{1.25}$$

where Φ is the cumulative density function of a standard Gaussian random variable. The above expression is derived from an asymptotic Gaussian approximations to $\hat{\rho}(x_i, x_j)$. Note that the PCER p-value refers to the probability of type I error rate which is incurred in hypothesis testing for one pair of gene at a time. To simultaneously test a total of Λ hypotheses, the following FDR-based procedure is used. It guarantees that FDR associated with hypotheses testing is not larger than α.

For a fixed FDR level α and cor_{min}, the procedure consists of the following two stages.

- In Stage I, the null hypothesis

$$H_0 : S_{x_i,x_j} = 0 \ \text{ versus } \ H_\alpha : S_{x_i,x_j} \neq 0 \tag{1.26}$$

  is tested at FDR level α. This employs the step-down procedure of Benjamini and Hochberg [72].
- Let us assume a total of Λ_1 gene pairs cross Stage I. In Stage II, asymptotic PCER confidence intervals $I^\lambda(\alpha)$ are constructed for each value of S corresponding to Λ_1 pairs. These intervals are then converted into FDR confidence intervals using the formula $I^\lambda(\alpha) \rightarrow I^\lambda(\Lambda_1\alpha/\Lambda)$ [73]. For the case of Pearson's correlation, let $z = \tanh^{-1}(\hat{\rho})$. Then the intervals $I^\lambda(\alpha)$, for Λ_1 true Pearson's correlation coefficients ρ, are given by Ref. [32]

$$\tanh\left(z - \frac{z_{\alpha/2}}{(N-3)^{1/2}}\right) \leq \rho \leq \left(z + \frac{z_{\alpha/2}}{(N-3)^{1/2}}\right), \tag{1.27}$$

  where $P(N(0, 1) > z_{\alpha/2}) = \alpha/2$. A gene pair is declared to be both statistically and biologically significant if the corresponding FDR confidence interval and the interval $[-\text{cor}_{\text{min}}, \text{cor}_{\text{min}}]$ do not intersect.

RNs offer a simple and computationally efficient approach to infer undirected biological networks. However, RNs only infer a possible functional relevancy between gene pairs and not necessarily their direct association. A high correlation value may result from an indirect association, for example, regulation of a pair of genes by another gene. Thus, RNs are often dense with many interpretable functional modules. Limitations of RNs have been studied in Refs. [69,71].

1.4.2.2 Graphical Gaussian Models

To overcome the shortcomings of RNs, Gaussian graphical models [1–3] were introduced to measure the strength of direct pairwise associations. In GGMs, gene associations are quantified in terms of partial correlations. Indeed, marginal correlation measures a composite correlation between a pair of genes that includes the effects of all other genes in the network, whereas partial correlation measures the strength of direct correlation excluding the effects of all other genes.

In GGMs [1,2], it is assumed that data are drawn from a multivariate normal distribution $N(\mu, \Sigma)$. The partial correlation matrix Π is computed from the inverse

$\Omega = (\omega_{ij}) = \Sigma^{-1}$ of the covariance matrix as

$$\pi_{ij} = -\omega_{ij}/\sqrt{\omega_{ii}\omega_{jj}}. \tag{1.28}$$

Calculation of partial correlation matrix is followed by statistical tests, which determine the strength of partial correlation computed for every pair of genes. Significantly nonzero entries in the estimated partial correlation matrix are used to reconstruct the underlying network.

However, the above method is applicable only if the sample size (N) is larger than the number of genes (p) in the given data set, for otherwise the sample covariance matrix cannot be inverted. To tackle the case of small N and large p, a shrinkage covariance estimator has been developed [3], which guarantees the positive definiteness of the estimated covariance matrix and thus leads to its invertibility. The shrinkage estimator $\hat{\Sigma}$ is written as a convex combination of the following two estimators:

- unconstrained estimator $\hat{\Sigma}_{\mathrm{U}}$ of the covariance matrix, which often has a high variance
- constrained estimator $\hat{\Sigma}_{\mathrm{C}}$ of the covariance matrix, which has a certain bias but a low variance.

This is expressed as

$$\hat{\Sigma} = (1 - \lambda)\hat{\Sigma}_{\mathrm{U}} + \lambda\hat{\Sigma}_{\mathrm{C}}, \tag{1.29}$$

where $\lambda \in [0\ 1]$ represents the shrinkage parameter. The Ledoit–Wolf lemma [74] is used to estimate an optimal value of λ which minimizes the expected value of mean square error. Let $A = [a_{ij}]$ and $B = [b_{ij}]$ denote empirical covariance and correlation matrices, respectively. Then $\hat{\Sigma}$ is given by [3]

$$\hat{\Sigma}_{ij} = \begin{cases} a_{ii}, \text{ if } i = j \\ \hat{b}_{ij}\sqrt{a_{ii}a_{jj}}, \text{ otherwise} \end{cases} \tag{1.30}$$

where

$$\hat{b}_{ij} = \begin{cases} 1, \text{ if } i = j \\ b_{ij}\min(1, \max(0, 1 - \hat{\lambda}^*)), \text{ otherwise} \end{cases} \tag{1.31}$$

and

$$\hat{\lambda}^* = \frac{\sum_{i,j,i\neq j}\widehat{\mathrm{Var}}(b_{ij})}{\sum_{i,j,i\neq j} b_{ij}^2}. \tag{1.32}$$

For the list of constrained estimators and computation of $\widehat{\mathrm{Var}}(b_{ij})$, we refer to Ref. [3]. Overall, GGM is an appealing approach for the reverse engineering of undirected biological networks. It is theoretically sound, easy to understand and computationally efficient. GGM is particularly suitable to tackle low throughput data, where the number of samples is much larger than the number of variables. For high throughput molecular profiling data, the distribution-free shrinkage estimator guarantees to estimate an invertible covariance matrix. However, an edge in a network reconstructed via GGM only represents a possible functional relationship between corresponding genes without any indication of gene regulatory mechanisms.

Reconstruction of biological networks is fundamental in understanding the origin of various biological phenomenon. The computational approaches presented above play a crucial role in achieving this goal. However, the complexity arising due to a large number of variables and many hypothetical connections introduces further challenges in gaining biological insights from a reconstructed network. It is necessary to uncover the structural arrangement of a large biological network by identifying tightly connected zones of the network representing functional modules. In the following section, we present some popular network partitioning algorithms, which allow us to infer the biomolecular mechanisms at the level of subnetworks.

1.5 PARTITIONING BIOLOGICAL NETWORKS

Often a reconstructed network is too broad of a representation for a specific biological process. The partitioning of biological networks allows for the careful analysis of hypothesized biological functional units. Users may choose to partition high fidelity biological networks obtainable from a variety of sources such as the Kyoto Encyclopedia of Genes and Genomes (KEGG) database [75]. There is no universal definition for partitions, clusters, and especially communities. However, in this chapter we define a partition as a subnetwork (subgraph) of the given network (graph) such that (1) the internal connections of the partition from node to node are strong and (2) the external connections between other partitions are weak.

There are two major classes of partitioning algorithms called graph clustering algorithms and community detection algorithms [22]. Graph clustering algorithms originated from computer science and other closely related fields. Community detection algorithms have their origin in sociology, which now encompass applications in applied mathematics, physics, and biology.

For graph clustering algorithms, the number of clusters is a user-specified parameter. A graph clustering algorithm *must* always return the specified number of clusters regardless of whether the clusters are structurally meaningful in the underlying graph. These algorithms were developed for specific applications, such as placing the parts of an electronic circuit onto printed circuit cards or improving the paging properties of programs [23]. For other applications such as finding the communities of a biological network, specifying a number of clusters beforehand may be arbitrary and could result in an incorrect reflection of the underlying network topology. However, many techniques found in graph clustering algorithms have been modified to fulfill the needs of community detection algorithms rendering knowledge of graph clustering algorithms to be quite useful.

Community detection algorithms assume that the network itself divides into partitions or communities. The goal of a researcher is to find these communities. If the given network does not have any communities, this result is quite acceptable and yields valuable information about the network's topology. Community detection algorithms do not forcibly divide the network into partitions as opposed to graph clustering algorithms. On the contrary, community detection algorithms treat the communities as a network property similar to the degree distribution of a network.

The partitioning of biological networks is better served via community detection algorithms. Since there are instances where community detection algorithms adopt techniques from graph clustering algorithms, the study of graph clustering algorithms in and of itself is quite fruitful. We will provide a brief overview of the Kernighan–Lin algorithm [23] which is considered as one of the best clustering algorithms. The remainder of this chapter will then focus on community detection algorithms.

1.5.1 Directed and Undirected Networks

Most algorithms for network partitioning take an undirected network as input. In particular, the focus of community detection algorithms on undirected networks may have originated from the nature of social networks, which depict relationships between individuals that are by nature undirected. Often times, it is not trivial to extend an algorithm to handle both directed and undirected networks [21]. Many users simply ignore edge direction when using an undirected algorithm. However, vital information is often lost when ignoring the direction of edges as in the case of signaling pathways in biological systems. Ignoring edge direction causes the *E. coli* network to have six communities as opposed to none as seen in Figure 1.7.

1.5.2 Partitioning Undirected Networks

There are many algorithms that take undirected networks as input. For the purposes of this chapter, we will mainly focus on community detection algorithms. For graph clustering algorithms, we will explore the well-known Kernighan–Lin algorithm [23].

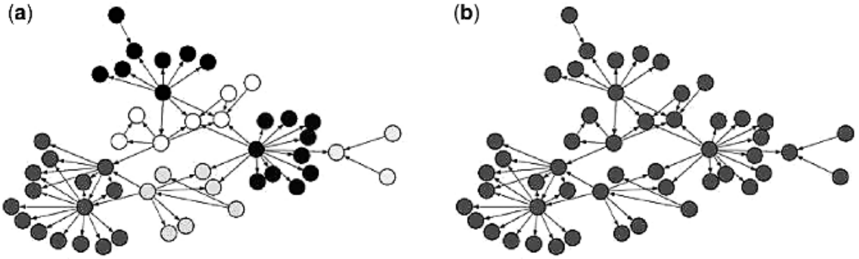


FIGURE 1.7 The *E. coli* network from the DREAM Initiative [39]. (a) The *E. coli* network is partitioned into six communities by ignoring edge direction. (b) The same *E. coli* network does not divide into any communities when edge direction is used. The disparity between the results is a strong indicator of the significance of edge direction. In both cases the appropriate version of Infomap was run for 100,000 iterations with a seed number of 1.

We will present the Girvan–Newman algorithm [25], Newman's eigenvector method, Infomap [27], and the clique percolation method [26].

To compare different algorithms, it is very helpful to have some gold standard networks whose true community divisions are known. A variety of different benchmarks are mentioned by Fortunato [21]. We choose a small gold standard network as a benchmark to illustrate the results of the algorithms presented. In particular, we select Zachary's karate club [76] as illustrated in Figure 1.8. For a period of 2 years, Zachary studied 34 karate club members. During this period, a disagreement arose between the club's instructor and the club's administrator. The club's instructor then left taking approximately half of the original club members. Zachary constructed a weighted network of their friendships, but we will use an unweighted network for our algorithm illustrations. Many community algorithms often use Zachary's network as a gold standard where they illustrate how accurate their algorithms could predict the eventual split of the club. Results for the Girvan–Newman algorithm, Newman's eigenvector method, Infomap, and the clique percolation method are presented in Figures 1.8 and 1.9.

1.5.2.1 Kernighan–Lin Algorithm

The Kernighan–Lin algorithm [23] is a famous algorithm used for network clustering. Developed in 1970, the Kernighan–Lin algorithm is still used often as a subroutine for more complex algorithms. The Kernighan–Lin algorithm was initially developed in order to partition electronic circuits on boards. Connections between these circuits are expensive so minimizing the number of connections is key. More formally, the Kernighan–Lin algorithm is a heuristic method that deals with the following combinatorics problem: given a weighted graph G, divide the $|V|$ vertices into k partitions no larger than a user-specified size m such that the total weight of the edges connecting the k partitions is minimized [23].

The major approach behind the algorithm is to divide the undirected graph G of $|V| = n_1 + n_2$ vertices into two subgraphs X and Y, $|X| = n_1$ and $|Y| = n_2$. Let c_{ij} be the cost from vertex i to vertex j. All c_{ii} equal zero (no self-loops are allowed in G) and $c_{ij} = c_{ji}$. The goal is to minimize the cost C of the edges connecting subgraphs X and Y, where for $x \in X$ and $y \in Y$

$$C = \sum_{X \times Y} c_{xy}. \tag{1.33}$$

For each node $x \in X$, let

$$D_x = \sum_{y \epsilon Y} c_{xy} - \sum_{z \epsilon X} c_{xz} \tag{1.34}$$

be the difference between the intracluster costs between vertex x and all vertices y, and the intercluster costs between vertex x and all other vertices in X. D_y is defined in a similar manner. Let

$$g = D_x + D_y - 2c_{xy} \tag{1.35}$$

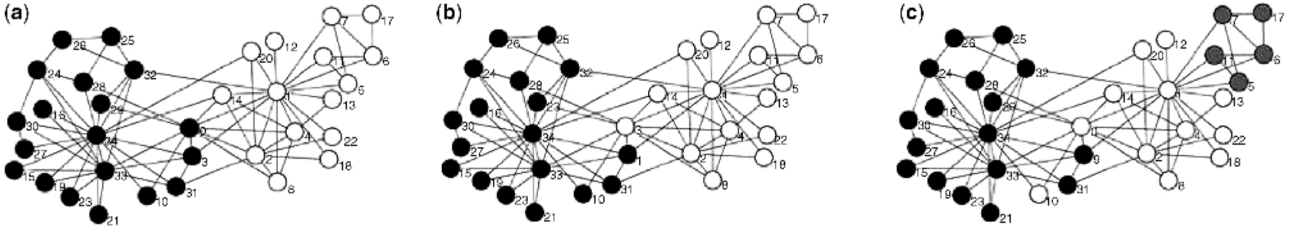


FIGURE 1.8 (a) The true partitioning of Zachary's karate club [76]. Newman's eigenvector method also returns this partition. (b) The Girvan–Newman algorithm mislabels a single node. (c) Infomap mislabels two nodes. It also subdivides the community shaded white into two subcommunities.

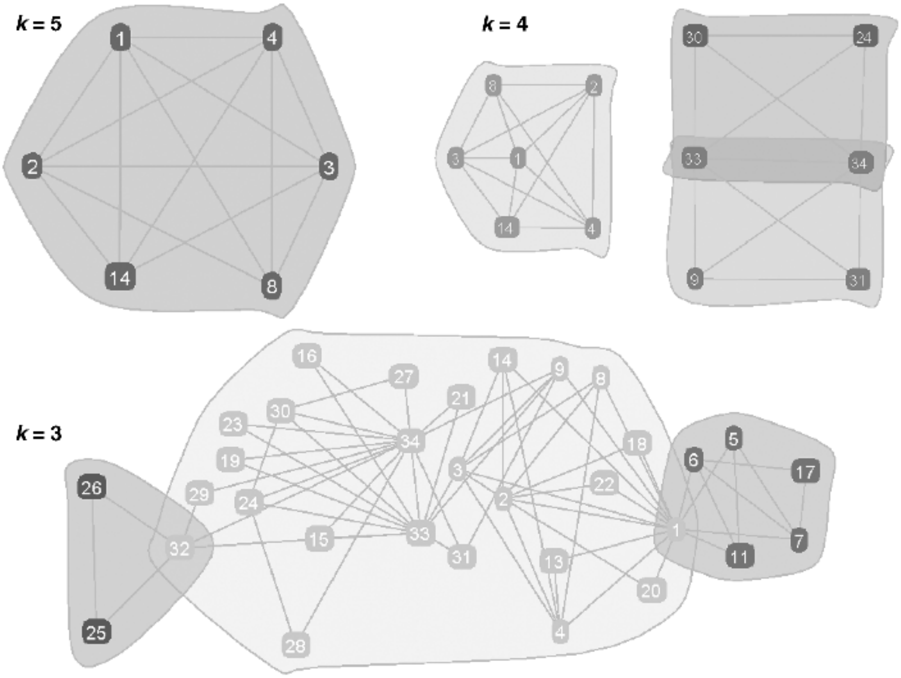


FIGURE 1.9 The partitioning of Zachary's karate club using CFinder [78]. There are one 5-community, three 4-communities, and three 3-communities. The 3-communities represent the most nodes with the exception of nodes 10 and 12. It also inaccurately places most of the opposing karate club members in a single community where the rival leaders represented by nodes 34 and 1 are in the same community.

be the gain for swapping two nodes x and y between their respective clusters. Let X and Y be the initial partitions of the graph G with $|X| = n_1$, $|Y| = n_2$, the number of vertices $|V| = n_1 + n_2$ and $n_1 \leq n_2$. The algorithm is as follows:

Algorithm 1.1

Kernighan–Lin Algorithm

```
Input: An undirected network $G$ and initial guesses for
        subnetworks $X$ and $Y$.
Output: Two subnetworks $X$ and $Y$ such that cost $C$ is minimized.
do {
  Calculate $D$ values for all $x \epsilon X$, $y \epsilon Y$
  Let $X^{'} = X, Y^{'} = Y$
  For $i = 1$ to $n_1$ {
     Select $x \epsilon X^{'}$ and $y \epsilon Y^{'}$ such that $g_i$ is maximized.
     Let $x'_i = x$ and $y'_i = y$.
     Remove $x$ from $X^{'}$ and $y$ from $Y^{'}$.
     Update the $D$ values of the remaining elements.
```

```
    }
    Select $k$ such that $G = \sum_{i=1}^{k} g_i$ is maximized.
    if $G \succ 0$
       swap the 1 to $k$ $x_i'$'s and $y_i'$'s between $X$ and $Y$
} until $G \leq 0$
```

The Kernighan–Lin algorithm has complexity $\mathrm{O}\left(|V|^2 \log |V|\right)$. It should be noted that the Kernighan–Lin algorithm is very sensitive to the initial guesses for the subnetworks X and Y. A random choice for initialization may yield a poor partition. It is often the case that a different algorithm provides an initial X and Y whereas the Kernighan–Lin algorithm improves upon the given X and Y. From the standpoint of biological networks, it may be highly unlikely to find a good guess for the initial partitions X and Y, especially if prior knowledge is lacking. Furthermore, the Kernighan–Lin algorithm by its nature imposes a minimum number of clusters. If a biological network does not possess any partitions, it should not be forced to have artificial partitions. Nevertheless, the Kernighan–Lin algorithm provides inspiration for a postprocessing method of communities introduced by Newman [22]. This postprocessing method can be used for different community algorithms as long as they optimize a quality function F. Newman uses modularity as his quality function, which will be introduced in Section 1.5.2.3.

Algorithm 1.2

Community Optimization

```
Input: An undirected network $G$ and initial guesses for
        subnetworks $X$ and $Y$.
Output: Two subnetworks $X$ and $Y$ such that the quality
         function $F$ is maximized.
do {
  For $i = 1$ to $|V|$ {
     Move the vertex $v$ from $X$ to $Y$ or $Y$ to $X$ such that
        the increase in $F$ is maximized. If no such $v$ exists,
        then select $v$ such that the decrease in $F$ is
        minimized.
     Remove the vertex $v$ from any further consideration.
     Store the intermediate partitioning results of the graph
        $G$ into subnetworks $X_i$ and $Y_i$ as $P_i$.
  }
  Select the partition $P_i$ that maximizes the increase in $F$.
  Let $X = X_i$ and $Y = Y_i$
} until $F$ can no longer be improved.
```

1.5.2.2 Girvan–Newman Algorithm

The Girvan–Newman algorithm [25] is one of the most well-known algorithms available for hierarchical clustering. These machine-learning algorithms are very popular and provide users with partitions of many different sizes. There are two major flavors in hierarchical clustering algorithms: agglomerative clustering and divisive clustering.

The Girvan–Newman algorithm [25] follows the spirit of divisive clustering algorithms. The Girvan–Newman algorithm departs from previous approaches by focusing on edges that serve as "bridges" between different communities. These edges have a high value for *edge betweenness,* which is an extension of vertex betweenness initially proposed by Freeman [77]. The authors defined three versions of edge betweenness: shortest-path betweenness, current-flow betweenness, and random-walk betweenness.

Agglomerative clustering is a bottom-up approach. Each node starts in its own cluster. Using a user-specified distance metric, the two most similar partitions are joined. This process continues until all nodes end up in a single partition. Agglomerative clustering algorithms are strong at finding the core of different communities but are weak in finding the outer layers of a community. Agglomerative clustering has also been shown to produce incorrect results for networks whose communities are known [33]. Divisive clustering algorithms, on the other hand, use a top-down approach. Such algorithms begin with the entire network as their input and recursively split the network into subnetworks. This process continues until every node is in its own partition as seen in Figure 1.10.

The focus for this section will be shortest-path betweenness as it provides the best combination of performance and accuracy [33]. In practice, it is also the most frequently used form of edge betweenness. To calculate shortest-path betweenness, all shortest paths between all pairs of vertices are calculated. For a given edge e, its betweenness score is a measure of how many shortest-paths possess edge e as a link. The authors provide a $\mathrm{O}\left(|V||E|\right)$ algorithm to calculate the shortest-path betweenness, where $|V|$ is the number of vertices and $|E|$ is the number of edges [33]. Overall, the Girvan–Newman algorithm has complexity $\mathrm{O}\left(|V||E|^2\right)$. The algorithm is as follows:

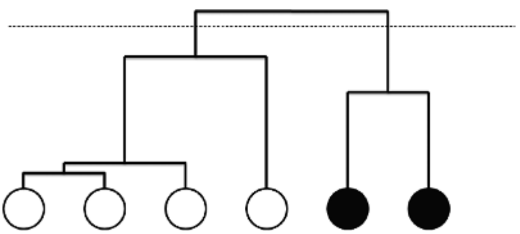


FIGURE 1.10 A dendrogram typically created by a divisive algorithm. The circles at the bottom represent the nodes of the graph. Using a top-down approach, the original graph is split until each node belongs in its own partition. The resulting number of partitions depends on where the dendrogram is *cut* . At the given cut line, there are two partitions colored white and black, respectively. Determining the proper cut line for a dendrogram is an active area of research.

Algorithm 1.3

Girvan–Newman Algorithm

```
Input: An undirected, unweighted network $G$.
Output: A hierarchy of different communities. The final number of
   communities is determined by where the dendrogram is cut.
For all edges in the graph, compute the shortest-path betweenness
   scores.
For $i = 1$ to $|E|$ {
   Remove the edge whose shortest-path betweenness score is
      maximal.
   Recompute the shortest-path betweenness scores for all edges
      affected by the removal.
}
```

The most important step in the Girvan–Newman algorithm is to recalculate the shortest-path betweenness scores. Once an edge is removed, the underlying network topology changes and so do the shortest paths of the network. In some cases an edge that had minimum shortest-path betweenness score in one iteration possesses the maximum score the very next iteration. Figure 1.11 illustrates the Girvan–Newman algorithm for a simple network.

The Girvan–Newman algorithm is very intuitive in that edges with a high edge-betweenness score serve as connections between different communities. It returns a varying number of communities based on where one cuts the dendrogram allowing for a more detailed analysis. It focuses on the flow of information in a network as shortest paths are one way to model the information flow of a network [21]. For biological networks this allows a researcher to examine a number of hypothesized functional biological units. There may be different biological insights by examining a larger community and its subcommunities. However, it is often the case that a researcher only seeks the best partitioning available among all candidate partitions. This leads to a major drawback concerning the Girvan–Newman algorithm as identifying where to cut the dendrogram to retrieve the final communities is an open question, especially if the number of communities is not known *a priori*. To remedy this situation, the authors introduced the concept of *modularity*, which will be discussed in more detail in Section 1.5.2.3. Another potential drawback associated with the Girvan–Newman algorithm is the lack of overlapping communities. In the case of biological networks, the lack

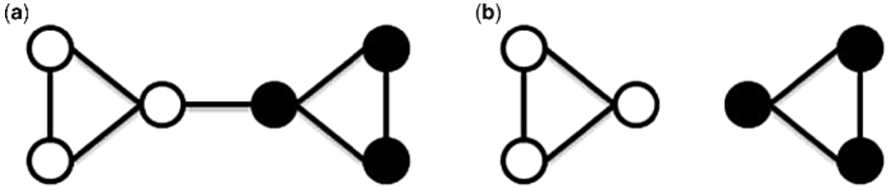


FIGURE 1.11 (a) The original graph consisting of six nodes and two communities. The central edge has the highest shortest-path betweenness score. (b) The network is divided into two communities after removal of the central edge.

of such a feature may be unreasonable as a gene may simultaneously participate in many different biological processes.

1.5.2.3 Newman's Eigenvector Method

In the preceding section, Newman and Girvan [33] introduced a new quality function called modularity in which a quality function assigns a score to a partitioning of a graph [21]. Whereas the Girvan–Newman algorithm used modularity to determine where to cut the dendrogram, there are many methods that optimize modularity directly including greedy techniques, simulated annealing, extremal optimization, and spectral optimization [21].

A major driving force behind modularity is that random graphs do not possess community structure [21]. Newman and Girvan proposed a model in which the original edges of the graph are randomly moved, but the overall expected degree of each node matches its degree in the original graph. In other words, modularity quantifies the difference of the number of edges falling within communities and the expected number of edges for an equivalent random network [22]. Modularity can be either negative or positive. High positive values of modularity indicate the presence of communities, and one can search for good divisions of a network by looking for partitions that have a high value for modularity. There are various modifications and formulas for modularity, but the focus for this section will be the modularity introduced by Newman [22].

For Newman's eigenvector method, Newman reformulates the problem by defining modularity in terms of the spectral attributes of the given graph. The eventual algorithm is very similar to a classical graph clustering algorithm called *Spectral Bisection* [21]. Suppose the graph G contains n vertices. Given a particular bipartition of the graph G, let $s_i = 1$ if vertex i belongs to the first community. If vertex i belongs to the second community, then $s_i = -1$. Let A_{ij} denote the elements of the adjacency matrix of G. Normally, A_{ij} is either 0 or 1, but it may vary for graphs where multiple edges are present. Placing edges at random in the network yields a number of expected edges $k_i k_j / 2m$ between two vertices i and j, where k_i and k_j are the degrees of their respective vertices. The number of undirected edges in the network is $m = \sum_{ij} A_{ij}/2$. The modularity Q is then defined as

$$Q = \frac{1}{4m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) s_i s_j. \tag{1.36}$$

As evident from Equation 1.36, a single term in the summation of modularity equals zero if vertices i and j belong to different communities. The modularity Q can be written in condensed form as

$$Q = \frac{1}{4m} s^T \, B \, s, \tag{1.37}$$

where the column vector s has elements s_i. Here, B is a symmetric matrix called the *modularity matrix* with entries equal to

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}. \tag{1.38}$$

The modularity matrix B has special properties akin to the graph Laplacian [22]. Each row and column sums to zero yielding an automatic eigenvector of $(1, 1, \dots)$ with eigenvalue 0. Modularity can now be rewritten as

$$Q = \frac{1}{4m} \sum_{i=1}^{n} (u_i^T \cdot s)^2 \beta_i, \tag{1.39}$$

where u_i is a normalized eigenvector of B with eigenvalue β_i. Let u_M denote the eigenvector with the largest eigenvalue β_M. Modularity can thus be maximized by choosing the values of s, where $s_i \epsilon \{-1, 1\}$, that maximize the dot product $u_M^T \cdot s$. This occurs by setting s_i to 1 when the corresponding element $u_{M_i} \succ 0$ and -1 otherwise. Newman's eigenvector method is as follows:

Algorithm 1.4

Newman's Eigenvector Method

```
Input: An undirected network $G$.
Output: Two partitions of graph $G$ such that the modularity $Q$ is
  maximized.
Find the eigenvector $u_M$ corresponding to the largest eigenvalue
  $\beta_M$ of the modularity matrix $B$.
Let $s_i = 1$ if $u_{M_i} \succ 0$ and $-1$ otherwise.
Return two partitions $X$ and $Y$. $X$ consists of all nodes whose
  corresponding $s_i$ equal to $1$. $Y$ consists of all nodes whose
  corresponding $s_i$ equal to $-1$.
```

Additional communities can be found by recursively applying Algorithm 1.4 to the discovered communities after a modification to Q [22]. Using the power method to find u_M, Newman's eigenvector method has complexity $O\left(|V|^2 \log |V|\right)$, where $|V|$ is the number of vertices in the graph [21]. Newman's eigenvector method excels in its speed. Another useful property of Newman's eigenvector method involves the values of u_M. The value $|u_{M_i}|$ corresponds directly to the strength of node i's membership in its cluster. Newman's eigenvector method also possesses a built-in stopping criterion. For a given graph G, if there are no positive eigenvalues, then G is a community in and of itself. Its major drawback is the same as spectral bisection where the algorithm gives the best results for the initial bisection of the graph [21]. Another major drawback involves the use of modularity as a quality function.

Fortunato [21] lists three major flaws for modularity. First, there are random graphs that may have partitions with high modularity, which undermines the very concept behind modularity. Second, modularity-based methods may suffer from a resolution limit. In other words, meaningful communities that are small with respect to the overall graph may be subsumed by larger communities. Finally, it has been shown that there exists an exponential number of partitions that have a high modularity, especially for networks possessing a strong hierarchical structure as most real networks do. Finding the global maximum may be computationally intractable.

1.5.2.4 Infomap

The inspiration behind Infomap [27] is to identify the partitions of a graph using as little information as needed to provide a coarse-grain description of the graph. Infomap uses a random walk to model information flow. A community is defined as a set of nodes for which the random walker spends a considerable time traversing between them. If the communities are well-defined, a random walker does not traverse between different communities often. A two-level description for a partition M is used where unique names are given to the communities within M, but individual node names across different communities may be reused. It is akin to map design where states have unique names but cities across different states may have the same name. The names for the communities and nodes are generated using a Huffman code. A good partitioning of the network thus consists of finding an optimal coding for the network. The map equation simplifies the procedure by providing a theoretical limit of how concisely a network may be described given a partitioning of the network. Using the map equation, the actual codes for different partitions do not have to be derived in order to choose the optimal among them. The objective becomes minimizing the minimum description length (MDL) of an infinite walk on the network. In other words, the MDL consists of the Shannon entropy of the random walk between communities and within communities [21]. The map equation is as follows:

$$L(M) = qH(Q) + \sum_{i=1}^{m} p^i H(P^i). \tag{1.40}$$

In the above equation, m is the number of communities. q is defined as

$$q = \sum_{i=1}^{m} q_i, \tag{1.41}$$

where each q_i is the probability per step that the random walker exits the ith community. $H(Q)$ is the movement entropy between communities and is calculated as

$$H(Q) = \sum_{i=1}^{m} \frac{q_i}{\sum_{j=1}^{m} q_j} \log \frac{q_i}{\sum_{j=1}^{m} q_j}. \tag{1.42}$$

The weight of the entropy of movements within the ith community, denoted by p^i, is defined as

$$p^i = q_i + \sum_{\alpha \epsilon i} p_\alpha. \tag{1.43}$$

Each p_α for node α in the ith community is the ergodic node visit frequency, that is, the average node visit frequencies for a random walk of infinite length. This is done using the power method. The entropy of movements within the ith community is calculated as

$$H(P^i) = \frac{q_i}{q_i + \sum_{\beta \epsilon i} p_\beta} \log \frac{q_i}{q_i + \sum_{\beta \epsilon i} p_\beta} + \sum_{\alpha \epsilon i} \frac{p_\alpha}{q_i + \sum_{\beta \epsilon i} p_\beta} \log \frac{p_\alpha}{q_i + \sum_{\beta \epsilon i} p_\beta}. \tag{1.44}$$

Algorithm 1.5

Infomap

```
Input: An undirected network $G$.
Output: A partition $M$ such that Equation 1.40 is minimized.
Assign each node into its own module.
do {
  Visit all of the modules in a random sequential order where at
     each module $i$ {
     Combine module $i$ to a neighboring module such that the
        Equation 1.40 decreases the most.
     If no such move exists, leave module $i$ as is.
  }
} until no move reduces Equation 1.40 any further.
```

Algorithm 1.5 is the core of Infomap version presented in [36]. There are two further subroutines that improve upon the results of the main algorithm listed in [36]. The three routines run for a user-specified number of iterations. The result returned is the best partition found among all of the iterations. It is important to note that while modularity focuses on the pairwise relationships between nodes, Infomap focuses on the flow of information within a network [21]. This underlying difference may often cause modularity-based methods and Infomap to generate different partitions. As Infomap uses a stochastic algorithm, it is not known how many iterations are needed before a good partitioning is found.

1.5.2.5 Clique Percolation Method

The clique percolation method [26] is a community detection algorithm that allows communities to share nodes. This feature is quite significant in the case of biological networks as a node in such networks often participates in many different biological processes. The inspiration behind the clique percolation method is that nodes within a community are highly connected to one another such that they form a *clique*. A clique is a subgraph in which any two nodes are connected by an edge. Between two different communities, the edges are few.

The authors define a *k-clique* community as a union of all *adjacent k-cliques* [26]. A k-clique is a complete subgraph consisting of k nodes such that there exists an edge between any two nodes in the subgraph. If two k-cliques share $k - 1$ nodes, they are said to be *adjacent*. Thus, a k-clique community is the union of all adjacent k-cliques. It is also important to define *connected components* as the final output consists of connected components. A graph is said to be *connected* if between any two vertices there exists at least one path connecting them [21]. A connected component is a maximal connected subgraph of a given graph [21]. A key aspect of the algorithm is building an $n \times n$ clique–clique overlap matrix M, where n is the number of maximal cliques. Each M_{ij} in the matrix represents the number of nodes shared by maximal clique i and maximal clique j. The algorithm is as follows:

Algorithm 1.6

Clique Percolation Method

```
Input: An undirected, unweighted network $G$ and the size $k$ of the
   $k$-cliques to find.
Output: A set of $k$-cliques communities.
Find all of the maximal cliques of the graph $G$.
Build a $n \times n$ clique-clique overlap matrix $M$.
Set all off-diagonal entries of matrix $M$ less than $k-1$ to zero.
Set all diagonal entries of $M$ less than $k$ to zero.
Return the connected components remaining in the matrix $M$ as the
  $k$-clique communities.
```

The major attraction of the clique percolation method is its ability to find overlapping communities. More importantly, the clique percolation method seems to possess the quality of making a clear distinction between graphs with community structure and random graphs [21]. A major drawback of the clique percolation method is that not all graphs have all of their nodes participating in a k-clique community [21]. It is often the case that leaf nodes are left out of communities. Another potential drawback involves the choice of k. One may not know *a priori* the value of k which yields meaningful structures, but the structure of the algorithm allows for finding multiple k-clique communities using the clique–clique overlap matrix M rather easily. Furthermore, finding the maximal cliques of a graph scales exponentially with the size of the graph. While the complexity of finding maximal cliques is known, there are additional factors involved for which the scalability of the clique percolation method cannot be expressed in closed form [21].

1.5.3 Partitioning Directed Networks

Algorithms which can take directed networks as input are often extensions of their undirected counterparts with additional criteria added to handle directed networks. Among the algorithms mentioned in the preceding section for undirected networks, Newman's eigenvector method, Infomap, and the clique percolation method have extensions allowing them to accept directed networks as input. Unfortunately, it is not always the case that the directed version of an algorithm is as rigorously developed as its undirected counterpart as will be seen below.

1.5.3.1 Newman's Eigenvector Method

Modularity-based methods have proven to be some of the most popular community detection algorithms. Most previous methods ignored edge direction when encountering a directed network. As seen in Figure 1.7, ignoring edge direction could lead to results that diverge greatly from the potential solution. Leicht and Newman [79] attempted to fill the gap for modularity-based algorithms by tweaking some key concepts for Newman's eigenvector method. Most of the notations used are the same as presented in Section 1.5.2.3 except where noted otherwise. Leicht and Newman first begin by modifying Equation 1.37 into

$$Q = \frac{1}{2m} s^T \, B \, s, \tag{1.45}$$

where s remains the column vector introduced in Section 1.5.2.3. The modularity matrix B is tweaked to account for edge direction and is given by

$$B_{ij} = A_{ij} - \frac{k_i^{\text{in}} k_j^{\text{out}}}{m}, \tag{1.46}$$

where A_{ij} is 1 in the presence of an edge from node j to node i and 0 otherwise. The term k_j^{out} is the out-degree or the number of edges leaving node j, k_i^{in} is the in-degree or the number of edges entering node i, and m is the total number of edges in the adjacency matrix of the graph G.

The modularity matrix B as presented in Equation 1.46 is asymmetrical, which may cause technical problems later on. To remedy this situation, the matrix B is replaced in Equation 1.45 with the sum of B and its transpose ensuring symmetry. Equation 1.45 now becomes

$$Q = \frac{1}{4m} s^T (B + B^T) s. \tag{1.47}$$

The algorithm to partition the graph G is essentially the same as Algorithm 1.4 except that the modularity matrix B defined in Equation 1.38 has been replaced with a symmetrical matrix $B + B^T$, where the latter B is defined in Equation 1.46. An advantage to this method is that essentially the underlying Newman's eigenvector method can be used unchanged except for some minor tweaks to account for edge direction. However, the given definition of modularity to incorporate edge direction is fundamentally flawed. Kim et al. [80] illustrated the shortcoming of the new definition for modularity as seen in Figure 1.12.

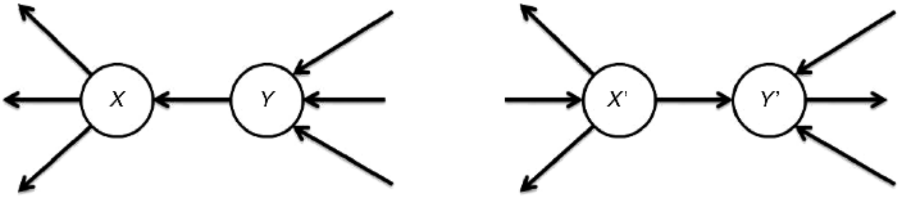


FIGURE 1.12 The two networks illustrate the problem with the directed version of modularity introduced by Leicht and Newman [79]. The in-degrees and out-degrees for nodes X and X' are the same. The same scenario holds for Y and Y'. The result is that the directed version of modularity is unable to distinguish between the two given networks [80].

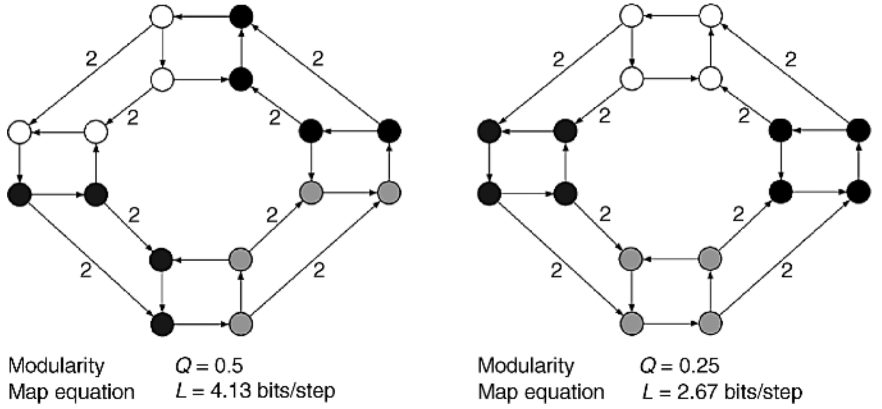

2
2
2
2
2
2
2
2
Modularity Q = 0.5
Map equation L = 4.13 bits/step
2
2
2
2
2
2
2
2
Modularity Q = 0.25
Map equation L = 2.67 bits/step

FIGURE 1.13 Rosvall and Bergstrom compared the performance of modularity and the map equation for the network illustrated above. Each method returned four different communities (two shaded black, one gray, and one white). On the left the network was partitioned by maximizing modularity. The corresponding value of the map equation is also listed. On the right the network was partitioned by minimizing Equation 1.40. In both partitions edges labeled with a 2 weigh twice as much as the unlabeled edges. For this simple network, one may observe that the map equation models the network's flow of information better than modularity.

1.5.3.2 Infomap

The extension of Infomap from the undirected case to the directed case is very straightforward. In the directed version of Infomap, a "teleportation probability" τ is introduced. With probability τ, the random walker jumps to a random node anywhere in the graph. This modification changes the undirected random walker into a directed "random surfer" akin to Google's PageRank algorithm. The default choice of 0.15 for τ is also akin to the damping factor $d = 0.85$ in Google's PageRank algorithm [27]. While the map equation remains the same, the exit probabilities q_i where $q = \sum_{i=1}^{m} q_i$ and m equals the number of communities, must be updated to include the contribution of τ. The underlying algorithm remains the same. For a sample comparison between the directed versions of modularity and Infomap, please refer to Figure 1.13.

1.5.3.3 Clique Percolation Method

In order to make the Clique percolation method work for directed networks, Palla et al. [34] extend the notion of k-cliques to *directed* k-cliques. For a directed acyclic graph, the edges of a directed k-clique always point from a node with a higher order to a node with a lower order. Equivalently, all nodes within the specified k-clique have different orders. The order of a node i within a k-clique is simply the sum of all edges leaving node i to the other nodes within the given k-clique. Palla et al. [34] illustrated a directed 4-clique as seen in Figure 1.14.

All other terminologies introduced in Section 1.5.2.5 also apply in this case. For example, two directed k-cliques are adjacent if they share $k - 1$ nodes. However, the directed case is more complicated as there are $3^{k(k-1)/2}$ ways in which links

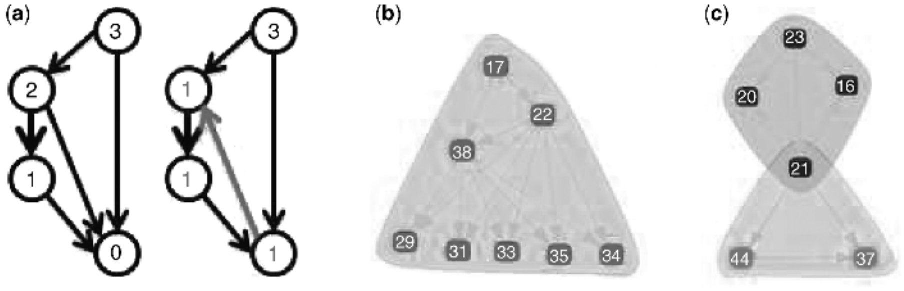


FIGURE 1.14 (a) A directed 4-clique graph without any cycles. The node labels refer to the order of the nodes which is the same as the number of edges leaving the node. (b) More than one node have the same order. The graph is not a directed 4-clique [34]. (c) The 3-communities of the *E. coli* network found using CFinder [78]. Many nodes in the *E. coli* network were left out of the final partitioning. Such an occurrence may prove problematic for analyzing biological networks in general.

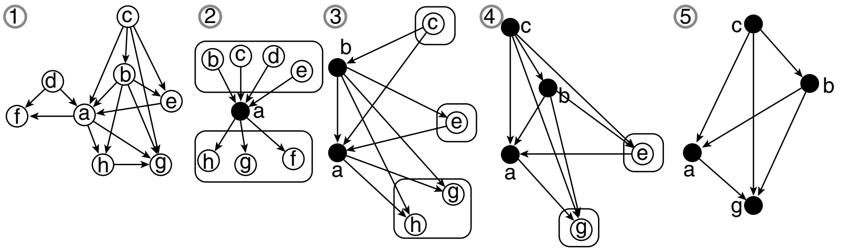


FIGURE 1.15 (1) The underlying network topology. (2) a is selected as the start node. The in-neighbors of a are placed in a container above a. The out-neighbors are placed in a container below a. (3) Select a new node from either container. In this case, b is selected. d and f are removed because they are not neighbors of b. e is placed in its own container as it is between a and b. (4) c is added. h is removed as it is not a neighbor of c. (5) g is added. Since e is not a neighbor of g, e is removed [34].

of a complete subgraph of size *k* can be directed [34]. The algorithm consists of the following two steps (1) the directed cliques of a given node are found and (2) the node and its links are removed from the network. Figure 1.15 from Ref. [34] illustrates the underlying algorithm. For graphs with cycles and a more detailed explanation of the algorithm, we refer to Ref. [34].

1.6 DISCUSSION

In this chapter, we discussed a number of approaches for the reconstruction and partition of biological networks. We considered the case of both directed and undirected biological networks in each of the above classes of problems. Network reconstruction algorithms presented in this chapter were further categorized based on the type of measurements used in the inference procedure. The type of measurements which we considered were gene expression data and symbol data comprising of gene sets. Gene expression data are numerical matrices containing gene expression levels measured from different experiments, whereas gene sets are sets of genes and do not assume the availability of the corresponding gene expression levels.

For the reconstruction of directed networks, we presented six approaches Boolean networks, probabilistic Boolean networks, Bayesian networks, cGraph, frequency method, and NICO. Among these approaches Boolean networks, probabilistic Boolean networks, and Bayesian networks accommodate gene expression data, whereas cGraph, frequency method, and NICO are suitable to infer the underlying network topologies from gene sets. For the reconstruction of undirected biological networks from gene expression data, we presented two approaches relevance networks and graphical Gaussian models. Nonetheless, the aforementioned algorithms for network inference using gene expression data are also applicable when the inputs are given in the form of gene set compendiums and vice versa. For instance, in order to apply a gene set based approach on gene expression data, an additional data discretization step can be incorporated to derive gene sets. Indeed, genes expressed in an experimental sample discretized using binary labels correspond to a gene set. Similarly, a gene set can be naturally represented as a binary sample by considering the presence or absence of a gene in the set. This equivalence can be used to infer a Bayesian network, Boolean network, or probabilistic Boolean network from a gene set compendium, and to infer mutual information networks, for example, mutual information version of relevance networks which accommodate discrete measurements. Similarly, gene sets obtained after discretizing gene expression data can be utilized to infer a network using NICO or cGraph. Overall, the equivalent representation of a gene set compendium as binary discrete data makes the network inference approaches applicable for both the types of input, gene sets or gene expression data. However, the approaches differ in their output, for example, directed versus undirected networks, and their computational efficiency. In general, the computational inference of undirected networks, for example, relevance networks and graphical Gaussian models is more efficient, as such approaches are based on estimating pairwise associations or similarities. For example, relevance networks measure the strength of pairwise interaction in terms of Pearson's correlation or mutual information, whereas graphical Gaussian models present a more appealing model by taking only direct interactions into account and use partial correlations to estimate the strength of pairwise associations. The two network inference approaches are frequently used in the field of information theory, pattern analysis and machine learning. In the inference of directed networks, Boolean networks and probabilistic Boolean networks present computationally efficient and simpler models, in comparison to Bayesian networks. However, the use of Boolean functions in both Boolean networks and probabilistic Boolean networks may cause the oversimplification of gene regulatory mechanisms. Nonetheless, Boolean networks find applications in many fields including biological systems, circuit theory, and computer science, for example, see Refs. [81,82]. Bayesian networks provide a sophisticated probabilistic modeling approach to infer gene regulatory mechanisms. As Bayesian networks suffer from nontrivial computational complexity, heuristics are applied to reduce the size of the search space of all possible Bayesian networks defined for a given number of nodes. Bayesian networks are used in a wide range of fields including medicine, document classification, information retrieval, image processing, and financial analysis. Among gene set based approaches, cGraph and frequency method are computationally efficient but they make stringent assumptions in the underlying models. For instance, cGraph adds a weighted edge between every pair of genes which appear in some gene set, whereas frequency method assumes *a prior* availability of the two end nodes in each gene set and directed edges involved in the corresponding paths. Expectation–maximization based NICO assumes a more general case and reconstructs the underlying network by inferring the order information for each unordered gene set. As the computational complexity of the approach grows exponentially with increase in the lengths of gene sets, an importance sampling based approximation of E-step has been proposed, which guarantees a polynomial time convergence of the EM algorithm. The above algorithms find applications in many real-world scenarios such as sociology, communication networks, and cognitive science.

We reviewed a variety of algorithms for network partitioning. The network partitioning algorithms were categorized as graph clustering algorithms and community detection algorithms. Graph clustering algorithms are applicable to very large-scale integration, distributing jobs on a parallel machine, and other applications found in computer science. Community detection algorithms, on the other hand, are more applicable to biological and social networks.

For graph clustering algorithms, we reviewed the well-known Kernighan–Lin algorithm. The Kernighan–Lin algorithm has complexity $\mathrm{O}\big(|V|^2 \log |V|\big)$. Although the Kernighan–Lin algorithm may not be directly applicable to biological networks, its "descendant," Algorithm 1.2, is directly applicable as a postprocessing step for many community detection algorithms [22].

The first community algorithm we presented was the Girvan–Newman algorithm with complexity $\mathrm{O}\big(|V||E|^2\big)$. The essence of the Girvan–Newman algorithm is that edges between communities have high edge-betweenness scores. By focusing on edge-betweenness, the Girvan–Newman algorithm focuses on the flow of the network as opposed to the immediate connection between nodes. Its major drawback is the lack of a proper criterion to determine the cut line of a dendrogram. Modularity was used to remedy the situation, but as seen in Section 1.5.2.3, modularity itself has its own drawbacks. An interesting solution may be replacing modularity as a quality function with the map equation introduced by Rosvall.

Next, we presented Newman's eigenvector method. This method is quite interesting as by defining modularity via Equation 1.37, the modularity matrix B defined in Equation 1.38 takes the position of the graph Laplacian in the spectral bisection algorithm. Newman's eigenvector method is considered to be quite fast with complexity $\mathrm{O}\big(|V|^2 \log |V|\big)$. The method focuses on the degrees and connections of immediate nodes as opposed to the flow of information in a given graph. A more useful aspect is that the value of $|u_{M_i}|$ for a node i corresponds directly to its participation strength in its community. The major drawback of Newman's eigenvector method is the same as spectral bisection method in which its core strength lies in finding the initial bipartition of a graph. There are also drawbacks involved with the choice of modularity as the quality function as seen in Section 1.5.2.3. Finally, extending Equation 1.37 to its directed counterpart Equation 1.45 does not incorporate edge direction correctly.

We then presented Infomap that utilizes information theory to compress good partitions and describe them using the least amount of bits possible. While modularity concentrates on the pairwise relationships between nodes, Infomap focuses on the flow of information within a network similar to the original Girvan–Newman algorithm. Its implementation for directed networks seems the most rigorous of all of the implementations presented. Since Infomap uses a stochastic algorithm, the number of iterations that are needed before a good partitioning is found is unknown.

Finally, we presented the clique percolation method. The clique percolation method is suitable for partitioning biological networks as it allows for overlap between different communities. It has some drawbacks as it may not place all nodes in a community, especially leaf nodes. The complexity of the clique percolation method cannot be expressed in closed form. Moreover, for the case of directed networks, the definition for directed k-clique is rather arbitrary.

Network reconstruction and partitioning are two fundamental problems in computational systems biology, which aim to provide a view of the underlying biomolecular activities at a global or local level. In general, the choice of a network reconstruction approach depends on various factors, for example, type of measurements, number of variables, sample size, amount of prior knowledge, and the type of interactions considered in an analysis. Similarly, network partitioning is dependent on the number of clusters overlapping with different clusters. Nevertheless, the two classes of problems are inherently related and one provides a foundation for the other. Structurally, a large biological network is an ensemble of smaller components or subnetworks. As tightly connected subnetworks often correspond to functional units of a biological network, network partitioning is essential to extract this finer level of detail. On the other hand, subnetworks together provide a global view of the underlying biological processes. In particular, gene set based approaches have recently gained attention in the computational inference of biological networks, for their natural ability to incorporate higher-order gene regulatory relationships. As gene expression data often have a small sample size and excessive noise, such data sets may not capture a complete picture of complex biomolecular activities. Network reconstruction and partitioning, two complementary problems in systems biology, together offer a potential avenue for future researches in gene set based inference of biological networks.

REFERENCES

1. H. Kishino, P.J. Waddell, Correspondence analysis of genes and tissue types and finding genetic links from microarray data, *Genome Inform.* **11**, 83–95 (2000).
2. J. Schäfer, K. Strimmer, An empirical Bayes approach to inferring large-scale gene association networks, *Bioinformatics* **21**, 754–764 (2005).

3. J. Schäfer, K. Strimmer, A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics, *Stat. Appl. Genet. Mol. Biol.* **4**, Article 32 (2005).
4. L. Glass, and S.A. Kauffman, The logical analysis of continuous non-linear biochemical control networks, *J. Theor. Biol.* **39**, 103–129 (1973).
5. S.A. Kauffman, Metabolic stability and epigenesis in randomly constructed genetic nets, *J. Theor. Biol.* **22**, 437–467 (1969).
6. H. Lähdesmäki, I. Shmulevich, O. Yli-Harja, On learning gene regulatory networks under the Boolean network model, *Mach. Learn.* 147–167 (2003).
7. I. Shmulevich, E.R. Dougherty, S. Kim, W. Zhang, Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks, *Bioinformatics* **18**(2), 261–274 (2002).
8. N. Friedman, M. Linial, I. Nachman, D. Peer, Using Bayesian networks to analyze expression data, *J. Comput. Biol.* **7**, 601–620 (2000).
9. E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, N. Friedman, Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data, *Nat. Genet.* **34**, 166–176 (2003).
10. T.S. Gardner, D. di Bernardo, D. Lorenz, J.J. Collins, Inferring genetic networks and identifying compound mode of action via expression profiling, *Science* **301**(5629), 102–105 (2003).
11. J. Tegner, M.K.S. Yeung, J. Hasty, J.J. Collins, Reverse engineering gene networks: integrating genetic perturbations with dynamical modeling, *Proc. Natl. Acad. Sci. U.S.A.* **100**(10), 5944–5949 (2003).
12. A.J. Butte, I.S. Kohane, Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements, *Pac. Symp. Biocomput.* **5**, 415–426 (2000).
13. A.S. Butte, I.S. Kohane, Relevance networks: a first step toward finding genetic regulatory networks within microarray data in *The Analysis of Gene Expression Data* (eds. G. Parmigiani, E.S. Garett, R.A. Irizarry, S.L. Zeger,), Springer, New York, pp. 428–446, 2003.
14. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Dalla Favera, A. Califano, ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinform.* Suppl 1, S7 (2006).
15. J.J. Faith, B. Hayete, J.T. Thaden, I. Mogno, J. Wierzbowski, G. Cottarel, S. Kasif, J.J. Collins, T.S. Gardner, Large-scale mapping and validation of *Escherichia coli* transcriptional regulation from a compendium of expression profiles, *PLoS Biol.* **5**(1), e8 (2007).
16. P.E. Meyer, K. Kontos, F. Lafitte, G. Bontempi, Information-theoretic inference of large transcriptional regulatory networks, *EUROSIP j. Bioinfor. Syst. Biol.* **2007** 79879, (2007).
17. J. Kubica, A. Moore, D. Cohn, J. Schneider, cGraph: a fast graph based method for link analysis and queries, *Proceedings of IJCAI Text-Mining and Link-Analysis Workshop*, Acapulco, Mexico, 2003.
18. M.G. Rabbat, J.R. Treichler, S.L. Wood, M.G. Larimore, Understanding the topology of a telephone network via internally sensed network tomography. *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 3, Philadelphia, PA, pp. 977–980, 2005.
19. M.G. Rabbat, M.A.T. Figueiredo, R.D. Nowak, Network inference from co-occurrences, *IEEE Trans. Inform. Theory* **54**(9), 4053–4068 (2008).

20. D. Zhu, M.G. Rabbat, A.O. Hero, R. Nowak, M.A.G. Figueirado, *De novo* reconstructing signaling pathways from multiple data source, in *New Research on Signaling Transduction* (B.R. Yanson, ed.), Nova Publisher, New York, 2007.
21. S. Fortunato, Community detection in graphs, *Phys. Rep.*, **486**, 75–174 (2010).
22. M.E.J. Newman, Modularity and community structure in networks. *PNAS* **103**(23), 8577–8582 (2006).
23. B.W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, *Bell Sys. Tech. J.* **49**, 291–307 (1970).
24. U. Luxburg, A tutorial on spectral clustering in *Statistics and Computing*, **17**(4), 395–416 (2007).
25. M. Girvan, M.E.J. Newman, Community structure in social and biological networks, *Proc. Nat. Acad. Sci. U.S.A.* **99**(12), 7821–7826 (2002).
26. G. Palla, I. Derenyi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, *Nature* **435**(7043), 814–818 (2005).
27. M. Rosvall, C.T. Bergstrom, Maps of random walks on complex networks reveal community structure, *Proc. Nat. Acad. Sci.* **105**(4), 1118–1123 (2008).
28. F.Y. Wu, The Potts model, *Rev. Mod. Phys.* **54**(1), 235–268 (1982).
29. F.Y. Wu, Potts model and graph theory, *J. Stat. Phys.* **52**(1), 99–112 (1988).
30. M.E.J. Newman, E. Leicht, Mixture models and exploratory analysis in networks. *PNAS*, **104**(23), 9564–9569 (2007).
31. U.N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, *Phys. Rev. E* **76**(3), 036106 (2007).
32. D. Zhu, A.O. Hero, Z.S. Qin, A. Swaroop, High throughput screening of co-expressed gene pairs with controlled False Discovery Rate (FDR) and Minimum Acceptable Strength (MAS), *J. Comput. Biol.* **12**(7), 1027–1043 (2005).
33. M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, *Phys. Rev. E* **69**(2), 026113 (2004).
34. G. Palla, I. Farkas, P. Pollner, I. Derényi, T. Vicsek, Directed network modules, *New J. Phys.* **9**(6), 186 (2007).
35. M. Rosvall, D. Axelsson, C.T. Bergstrom, The map equation, *Eur. Phys. J. Special Top.* **178**, 13–23 (2009).
36. M. Rosvall, C.T. Bergstrom, Mapping change in large networks, *PLoS ONE* **5**(1), e8694 (2010).
37. D. Marbach, T. Schaffter, C. Mattiussi, D. Floreano, Generating realistic *in silico* gene networks for performance assessment of reverse engineering methods, *J. Comput. Biol.* **16**(2), 229–239 (2009).
38. D. Marbach, R.J. Prill, T. Schaffter, C. Mattiussi, D. Floreano, G. Stolovitzky, Revealing strengths and weaknesses of methods for gene network inference. *Proc. Nat. Acad. Sci. U.S.A.* **107**(14), 6286–6291 (2010).
39. R.J. Prill, D. Marbach, J. Saez-Rodriguez, P.K. Sorger, L.G. Alexopoulos, X. Xue, N.D. Clarke, G. Altan-Bonnet, G. Stolovitzky, Towards a rigorous assessment of systems biology models: the DREAM3 challenges, *PLoS ONE* **5**(2), e9202 (2010).
40. P. Mendes, Framework for comparative assessment of parameter estimation and inference methods in systems biology, in *Learning and Inference in Computational Systems Biology*

(N.D. Lawrence, M. Girolami, M. Rattray, G. Sanguinetti, eds.), MIT Press, Cambridge, MA, pp. 33–58, 2009.

41. G. Stolovitzky, R.J. Prill, A. Califano, Lessons from the DREAM2 challenges, in *Annals of the New York Academy of Sciences* (G. Stolovitzky, P. Kahlem, A. Califano, eds.), vol. 1158, pp. 159–195, 2009.

42. B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter, *Molecular Biology of the Cell*, 4th ed., Garland Publisher 2002.

43. J.-P. Vert, Reconstruction of biological networks by supervised machine learning approaches, in *Elements of Computational Systems Biology* (H.M. Lodhi, S.H. Muggleton, eds.), John Wiley & Sons, Inc., Hoboken, NJ, 2010.

44. H. Pang A. Lin, M. Holford, B.E. Enerson, B. Lu, M.P. Lawton, E. Floyd, H. Zhao, Pathway analysis using random forests classification and regression, *Bioinformatics* **22**, 2028–2036 (2006).

45. H. Pang, H. Zhao, Building pathway clusters from Random Forests classification using class votes, *BMC Bioinform.* **9**(87) (2008).

46. A. Subramanian, P. Tamayo, V.K. Mootha, S. Mukherjee, B.L. Ebert, M.A. Gillette, A. Paulovich, S.L. Pomeroy, T.R. Golub, E.S. Lander, J.P. Mesirov, Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc. Nat. Acad. Sci. U.S.A.* **102**, 15545–15550 (2005).

47. G. Jr. Dennis, B.T. Sherman, D.A. Hosack, J. Yang, W. Gao, H.C. Lane, R.A. Lempicki, DAVID: database for annotation, visualization and integrated discovery. *Genome Biol.* **4**(5), P3 (2003).

48. D.W. Huang, B.T. Sherman, R.A. Lempicki, Systematic and integrative analysis of large gene lists using DAVID Bioinformatics Resources, *Nat. Protoc.* **4**(1), 44–57 (2009).

49. S. Wuchty, E. Ravasz, A. Barabáasi, The architecture of biological networks, in *Complex Systems Science in Biomedicine* (E. Micheli-Tzanakou, T. Deisboeck, J. Kresh, eds.), Springer US, 2006.

50. A. Zhang, *Protein Interaction Networks: Computational Analysis*, Cambridge University Press, Cambridge, UK, 2009.

51. K.L. Gunderson, S. Kruglyak, M.S. Graige, F. Garcia, B.G. Kermani, C. Zhao, D. Che, T. Dickinson, E. Wickham, J. Bierle, D. Doucet, M. Milewski, R. Yang, C. Siegmund, J. Haas, L. Zhou, A. Oliphant, J.B. Fan, S. Barnard, M.S. Chee, Decoding randomly ordered DNA arrays, *Genome Res.* **14**, 870–877 (2004).

52. D.J. Lockhart, H. Dong, M.C. Byrne, M.T. Follettie, M.V. Gallo, M.S. Chee, M. Mittmann, C. Wang, M. Kobayashi, H. Horton, E.L. Brown, Expression monitoring by hybridization to high-density oligonucleotide arrays, *Nat. Biotechnol.* **14**, 1675–1680 (1996).

53. M. Schena, D. Shalon, R.W. Davis, P.O. Brown, Quantitative monitoring of gene expression patterns with a complementary DNA microarray, *Science* **270** (5235), 368–371 (1995).

54. J. Shendure, R.D. Mitra, C. Varma, G.M. Church, Advanced sequencing technologies: methods and goals, *Nat. Rev. Genet.* **5**(5), 335–44 (2004).

55. J. Shendure, H. Ji, Next-generation DNA sequencing, *Nat. Biotechnol.*, **26**, 1135–1145 (2008).

56. T. Barrett, D.B. Troup, S.E. Wilhite, P. Ledoux, C. Evangelista, I.F. Kim, M. Tomashevsky, K.A. Marshall, K.H. Phillippy, P.M. Sherman, R.N. Muertter, M. Holko, O. Ayanbule, A. Yefanov, A. Soboleva, NCBI GEO: archive for functional genomics data sets: 10 years on, *Nucleic Acids Res.* **39**, D1005–D1010 (2010) (http://www.ncbi.nlm.nih.gov/geo).

57. H. Parkinson, U. Sarkans, N. Kolesnikov, N. Abeygunawardena, T. Burdett, M. Dylag, I. Emam, A. Farne, E. Hastings, E. Holloway, N. Kurbatova, M. Lukk, J. Malone, R. Mani, E. Pilicheva, G. Rustici, A. Sharma, E. Williams, T. Adamusiak, M. Brandizi, N. Sklyar, A. Brazma, ArrayExpress update: an archive of microarray and high-throughput sequencing-based functional genomics experiments. *Nucleic Acids Res.* **39**, D1002–D1004, (2010) (http://www.ebi.ac.uk/arrayexpress).

58. P.M. Kim, B. Tidor, Subsystem identification through dimensionality reduction of large-scale gene expression data, *Genome Res.* **13**(7), 1706–1718 (2003).

59. S. Huang, Gene expression profiling, genetic networks and cellular states: an integrating concept for tumorigenesis and drug discovery, *J. Molec. Med.* **77**, 469–480 (1999).

60. T. Schlitt, A. Brazma, Modelling in molecular biology: describing transcription regulatory networks at different scales, *Philos. Trans. R. Soc. Biol. Sci.* **361**(1467), 483–494 (2006).

61. D. Heckerman, D. Geiger, M. Chickering, Learning Bayesian networks: The combination of knowledge and statistical data, *Mach. Learn.* **20**, 197–243 (1995).

62. G.F. Cooper, E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, *Mach. Learn.* **9**(4), 309–347 (1992).

63. D.M. Chickering, Optimal structure identification with greedy search, *J. Mach. Learn. Res.*, **3**, 507–554 (2002).

64. R.W. Robinson, Counting unlabeled acyclic digraphs, in *Combinatorial Mathematics V* (C.H.C. Little ed.), Lecture Notes in Mathematics, 622, pp. 28–43, Springer, Berlin, 1977.

65. K. Murphy, Active learning of causal bayes net structure, Technical Report, UC Berkeley, 2001.

66. K. Murphy, Bayes Net Toolbox v5 for MATLAB, MIT AI Lab, Cambridge, MA, 2003.

67. V. Driessche, J. Demsar, E.O. Booth, P. Hill, P. Juvan, B. Zupan, A. Kuspa, G. Shaulsky, Epistasis analysis with global transcriptional phenotypes, *Nat. Genet.* **37**(5), 471–477 (2005).

68. D. Zhu, M.L. Dequéant, H. Li, Comparative analysis of distance based clustering methods, in *Analysis of Microarray Data: A Network Based Approach*, (F. Emmert-Streib, M. Dehmer, ed.), Wiley-VCH, Weinheim, Germany, 2007.

69. G. Altay, F. Emmert-Streib, Revealing differences in gene network inference algorithms on the network level by ensemble methods, *Bioinformatics* **26**(14), 1738–1744 (2010).

70. P.E. Meyer, F. Lafitte, G. Bontempi, Minet: an open source R/Bioconductor package for mutual information based network inference, *BMC Bioinform.* **9**, 461 (2008).

71. F. Emmert-Streib, G. Altay, Local network-based measures to assess the inferability of different regulatory networks, *IET Syst. Biol.* **4**(4), 277–288 (2010).

72. Y. Benjamini, Y. Hochberg, Controlling the false discovery rate: a practical and powerful approach to multiple testing, *J. R. Stat. Soc. Ser. B.* **57**(1), 289–300 (1995).

73. Y. Benjamini, D. Yekutieli, False discovery rate adjusted multiple confidence intervals for selected parameters, *J. Am. Stat. Assoc.* **100**, 71–80 (2004).

74. O. Ledoit, M. Wolf, Improved estimation of the covariance matrix of stock returns with an application to portfolio selection, *J. Emp. Finance* **10**, 603–621 (2003).

75. M. Kanehisa, S. Goto, M. Hattori, K.F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, M. Hirakawa, From genomics to chemical genomics: new developments in KEGG, *Nucleic Acids Res.* **34** (Database issue), D354–D357 (2006).

76. W.W. Zachary, An information flow model for conflict and fission in small groups, *J. Anthropol. Res.* **33**, 452–473 (1977).
77. L.C. Freeman, A set of measures of centrality based on betweenness, *Sociometry* **40**, 35–41 (1977).
78. B. Adamcsek, G. Palla, I.J. Farkas, I. Derenyi, T. Vicsek, CFinder: locating cliques and overlapping modules in biological networks, *Bioinformatics* **22**(8), 1021–1023 (2006).
79. E. Leicht, M.E.J. Newman, Community structure in directed networks, *Phys. Rev. Lett.* **100**(11), 118703 (2008).
80. Y. Kim, S. Son, H. Jeong, Finding communities in directed networks. *Phys. Rev. E* **81**(1), 016103 (2010).
81. P. Dunne, *The Complexity of Boolean Networks*, Academic Press, CA, 1988.
82. R.J. Tocci, R.S. Widmer, *Digital Systems: Principles and Applications*, 8 edn., Prentice Hall, NJ, 2001.