STATISTICAL AND MACHINE LEARNING APPROACHES FOR NETWORK ANALYSIS

STATISTICAL AND MACHINE LEARNING APPROACHES FOR NETWORK ANALYSIS

Edited by

MATTHIAS DEHMER

UMIT – The Health and Life Sciences University, Institute for Bioinformatics and Translational Research, Hall in Tyrol, Austria

SUBHASH C. BASAK

Natural Resources Research Institute
University of Minnesota, Duluth
Duluth, MN, USA

WILEY

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey
Published simultaneously in Canada

Library of Congress Cataloging-in-Publication Data:

ISBN: 978-0-470-19515-4

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

To Christina

CONTENTS

PREFACE

An emerging trend in many scientific disciplines is a strong tendency toward being transformed into some form of information science. One important pathway in this transition has been via the application of network analysis. The basic methodology in this area is the representation of the structure of an object of investigation by a graph representing a relational structure. It is because of this general nature that graphs have been used in many diverse branches of science including bioinformatics, molecular and systems biology, theoretical physics, computer science, chemistry, engineering, drug discovery, and linguistics, to name just a few. An important feature of the book "Statistical and Machine Learning Approaches for Network Analysis" is to combine theoretical disciplines such as graph theory, machine learning, and statistical data analysis and, hence, to arrive at a new field to explore complex networks by using machine learning techniques in an interdisciplinary manner.

The age of network science has definitely arrived. Large-scale generation of genomic, proteomic, signaling, and metabolomic data is allowing the construction of complex networks that provide a new framework for understanding the molecular basis of physiological and pathological states. Networks and network-based methods have been used in biology to characterize genomic and genetic mechanisms as well as protein signaling. Diseases are looked upon as abnormal perturbations of critical cellular networks. Onset, progression, and intervention in complex diseases such as cancer and diabetes are analyzed today using network theory.

Once the system is represented by a network, methods of network analysis can be applied to extract useful information regarding important system properties and to investigate its structure and function. Various statistical and machine learning methods have been developed for this purpose and have already been applied to networks. The purpose of the book is to demonstrate the usefulness, feasibility, and the impact of the methods on the scientific field. The 11 chapters in this book written by internationally reputed researchers in the field of interdisciplinary network theory cover a wide range of topics and analysis methods to explore networks statistically.

The topics we are going to tackle in this book range from network inference and clustering, graph kernels to biological network analysis for complex diseases using statistical techniques. The book is intended for researchers, graduate and advanced undergraduate students in the interdisciplinary fields such as biostatistics, bioinformatics, chemistry, mathematical chemistry, systems biology, and network physics. Each chapter is comprehensively presented, accessible not only to researchers from this field but also to advanced undergraduate or graduate students.

Many colleagues, whether consciously or unconsciously, have provided us with input, help, and support before and during the preparation of the present book. In particular, we would like to thank Maria and Gheorghe Duca, Frank Emmert-Streib, Boris Furtula, Ivan Gutman, Armin Graber, Martin Grabner, D. D. Lozovanu, Alexei Levitchi, Alexander Mehler, Abbe Mowshowitz, Andrei Perjan, Ricardo de Matos Simoes, Fred Sobik, Dongxiao Zhu, and apologize to all who have not been named mistakenly. Matthias Dehmer thanks Christina Uhde for giving love and inspiration. We also thank Frank Emmert-Streib for fruitful discussions during the formation of this book.

We would also like to thank our editor Susanne Steitz-Filler from Wiley who has been always available and helpful. Last but not the least, Matthias Dehmer thanks the Austrian Science Funds (project P22029-N13) and the Standortagentur Tirol for supporting this work.

Finally, we sincerely hope that this book will serve the scientific community of network science reasonably well and inspires people to use machine learning-driven network analysis to solve interdisciplinary problems successfully.

Matthias Dehmer
Subhash C. Basak

CONTRIBUTORS

Lipi Acharya, Department of Computer Science, University of New Orleans, New Orleans, LA, USA

Enrico Capobianco, Laboratory for Integrative Systems Medicine (LISM) IFC-CNR, Pisa (IT); Center for Computational Science, University of Miami, Miami, FL, USA

Christina Chan, Departments of Chemical Engineering and Material Sciences, Genetics Program, Computer Science and Engineering, and Biochemistry and Molecular Biology, Michigan State University, East Lansing, MI, USA

Ricardo de Matos Simoes, Computational Biology and Machine Learning Lab, Center for Cancer Research and Cell Biology, School of Medicine, Dentistry and Biomedical Sciences, Queen's University Belfast, UK

Frank Emmert-Streib, Computational Biology and Machine Learning Lab, Center for Cancer Research and Cell Biology, School of Medicine, Dentistry and Biomedical Sciences, Queen's University Belfast, UK

Damien Fay, Computer Laboratory, Systems Research Group, University of Cambridge, UK

Hirosha Geekiyanage, Genetics Program, Michigan State University, East Lansing, MI, USA

Elisabeth Georgii, Department of Information and Computer Science, Helsinki Institute for Information Technology, Aalto University School of Science and Technology, Aalto, Finland

Hamed Haddadi, Computer Laboratory, Systems Research Group, University of Cambridge, UK

Thair Judeh, Department of Computer Science, University of New Orleans, New Orleans, LA, USA

Reinhard Kutzelnigg, Math.Tec, Heumühlgasse, Wien, Vienna, Austria

Elisabetta Marras, CRS4 Bioinformatics Laboratory, Polaris Science and Technology Park, Pula, Italy

Andrew W. Moore, School of Computer Science, Carnegie Mellon University, USA

Richard Mortier, Horizon Institute, University of Nottingham, UK

Chikoo Oosawa, Department of Bioscience and Bioinformatics, Kyushu Institute of Technology, Iizuka, Fukuoka 820-8502, Japan

Matthias Rupp, Machine Learning Group, Berlin Institute of Technology, Berlin, Germany, and, Institute of Pure and Applied Mathematics, University of California, Los Angeles, CA, USA; currently at the Institute of Pharmaceutical Sciences, ETH Zurich, Zurich, Switzerland.

Kazuhiro Takemoto, Department of Bioscience and Bioinformatics, Kyushu Institute of Technology, Iizuka, Fukuoka 820-8502, Japan; PRESTO, Japan Science and Technology Agency, Kawaguchi, Saitama 332-0012, Japan

Andrew G. Thomason, Department of Pure Mathematics and Mathematical Statistics, University of Cambridge, UK

Antonella Travaglione, CRS4 Bioinformatics Laboratory, Polaris Science and Technology Park, Pula, Italy

Koji Tsuda, Computational Biology Research Center, National Institute of Advanced Industrial Science and Technology AIST, Tokyo, Japan

Steve Uhlig, School of Electronic Engineering and Computer Science, Queen Mary University of London, UK

Tim vor der Brück, Department of Computer Science, Text Technology Lab, Johann Wolfgang Goethe University, Frankfurt, Germany

Xuewei Wang, Department of Chemical Engineering and Material Sciences, Michigan State University, East Lansing, MI, USA

Dongxiao Zhu, Department of Computer Science, University of New Orleans; Research Institute for Children, Children's Hospital; Tulane Cancer Center, New Orleans, LA, USA

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.

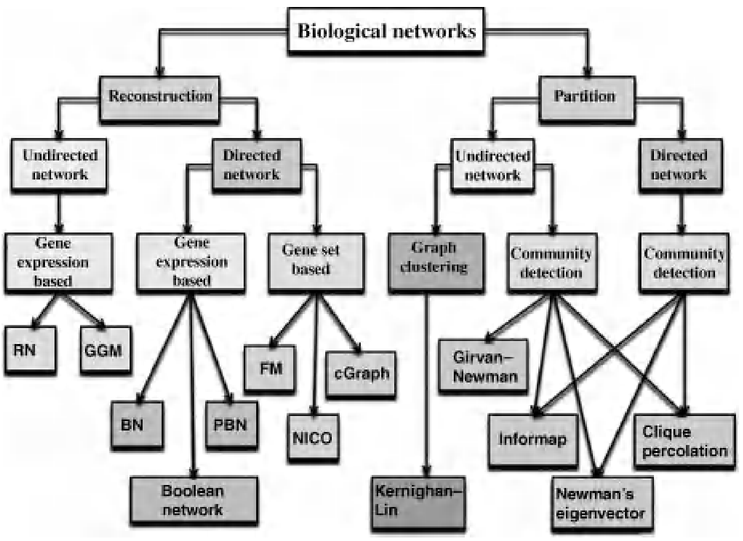


FIGURE 1.1 Approaches addressing two fundamental problems in computational systems biology (1) reconstruction of biological networks from two complementary forms of data resources, gene expression data and gene sets and (2) partitioning of large biological networks to extract functional units. Two classes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

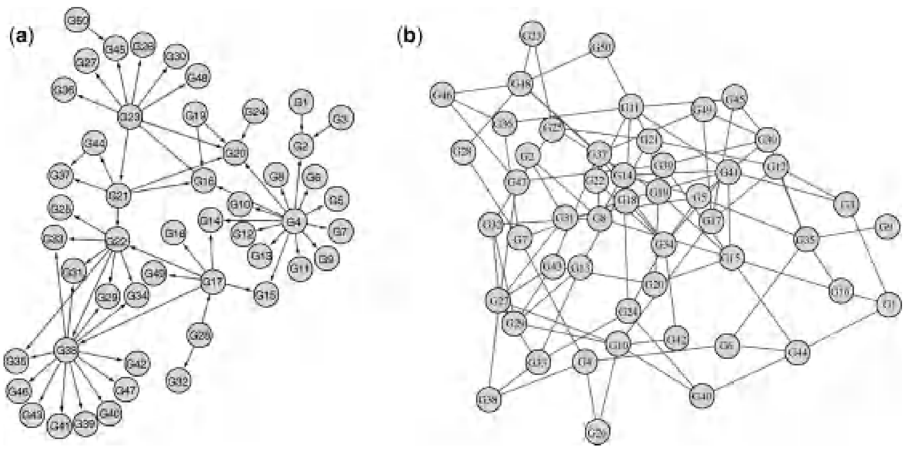


FIGURE 1.2 (a) Example of a directed network. The figure shows *Escherichia coli* gold standard network from the DREAM3 Network Challenges [37–39]. (b) Example of an undirected network. The figure shows an *in silico* gold standard network from the DREAM2 Network Challenges [40,41].

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,

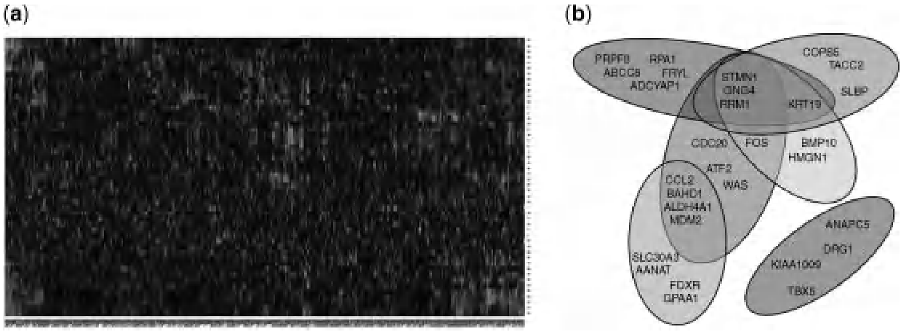


FIGURE 1.3 Two complementary forms of data accommodated by the existing network reconstruction algorithms. (a) Gene expression data generated from high-throughput platforms, for example, microarray. (b) Gene sets often resulted from explorative analysis of large-scale gene expression data, for example,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

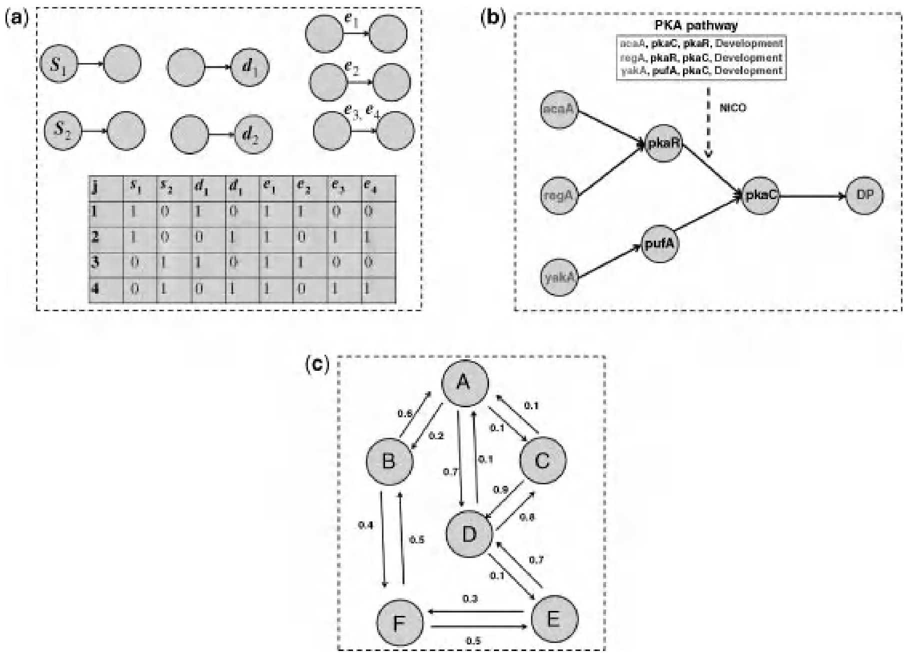
where each row corresponds to a gene, each column represents an experiment and each entry in the matrix stands for gene expression level. Gene expression profiling enables the measurement of expression levels of thousands of genes simultaneously and thus allows for a systematic study of biomolecular 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].

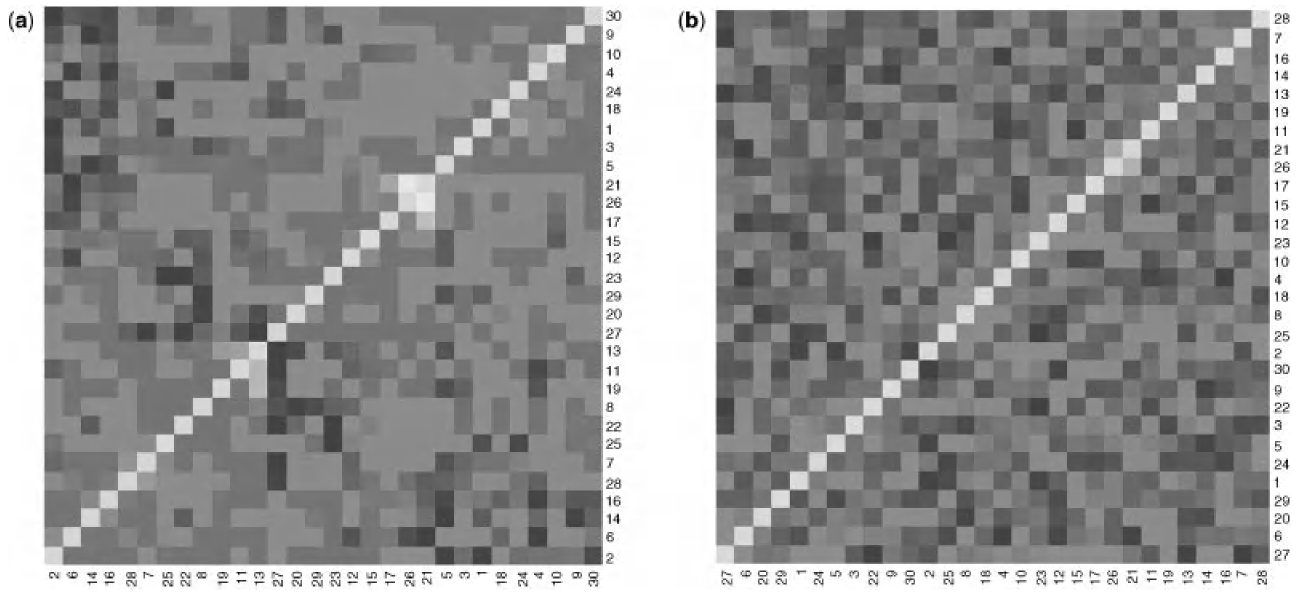


FIGURE 1.5 Comparison of correlation-based relevance networks (a) and partial correlation based graphical Gaussian modeling (b) performed on a synthetic data set generated from multivariate normal distribution. The figures represent estimated correlations and partial correlations between every pair of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

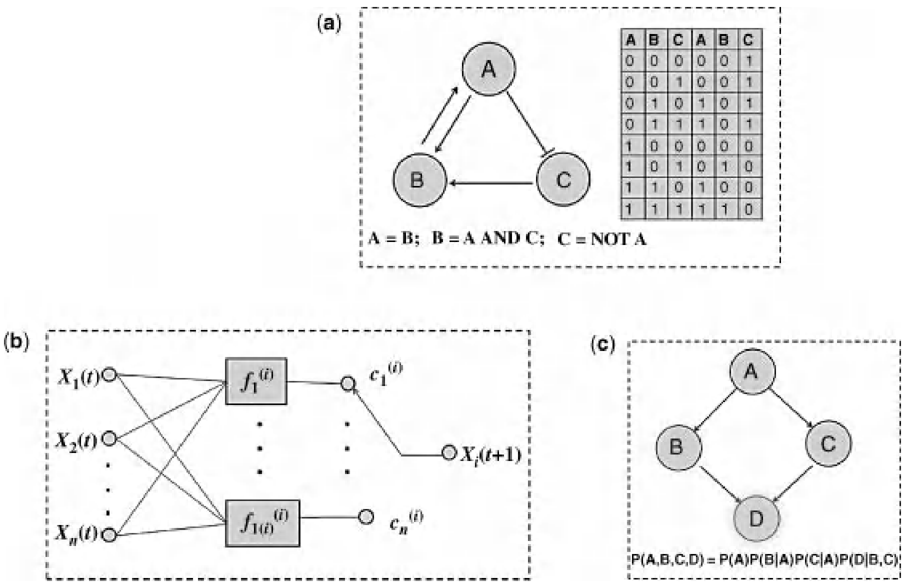
$$f_k = \left(f_{k_1}^{(1)}, \ldots, f_{k_n}^{(n)} \right), \tag{1.2}$$

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

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

and

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

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

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

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

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

and

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

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

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

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 $\text{cor}_{\min}$, 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 $I^\lambda(\alpha) \to I^\lambda(\Lambda_1\alpha/\Lambda)$ [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 $[-\text{cor}_{\min}, \text{cor}_{\min}]$ 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].

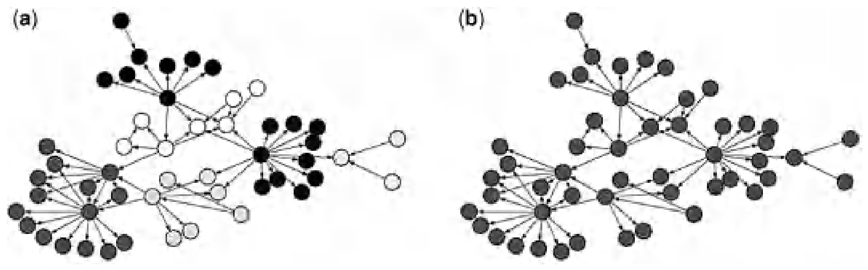


FIGURE 1.7 The *E. coli* network from the DREAM Initiative [39]. (a) The *E. coli* network is partitioned into six communities by ignoring edge direction. (b) The same *E. coli* network does not divide into any communities when edge direction is used. The disparity between the results is a strong indicator 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

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

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

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

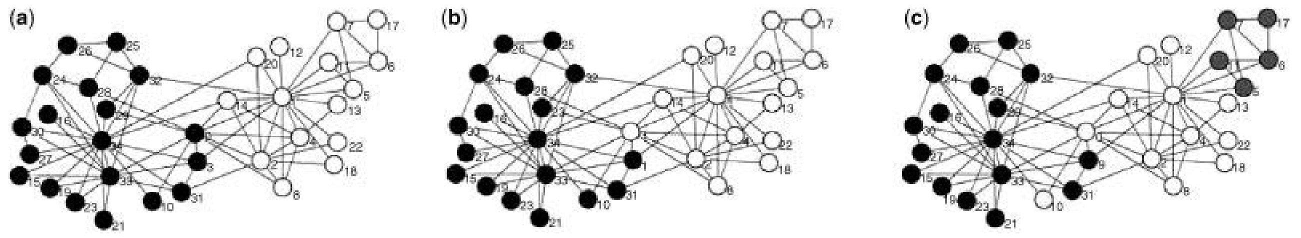


FIGURE 1.8 (a) The true partitioning of Zachary's karate club [76]. Newman's eigenvector method also returns this partition. (b) The Girvan–Newman algorithm mislabels a single node. (c) Infomap mislabels two nodes. It also subdivides the community shaded white into two subcommunities.

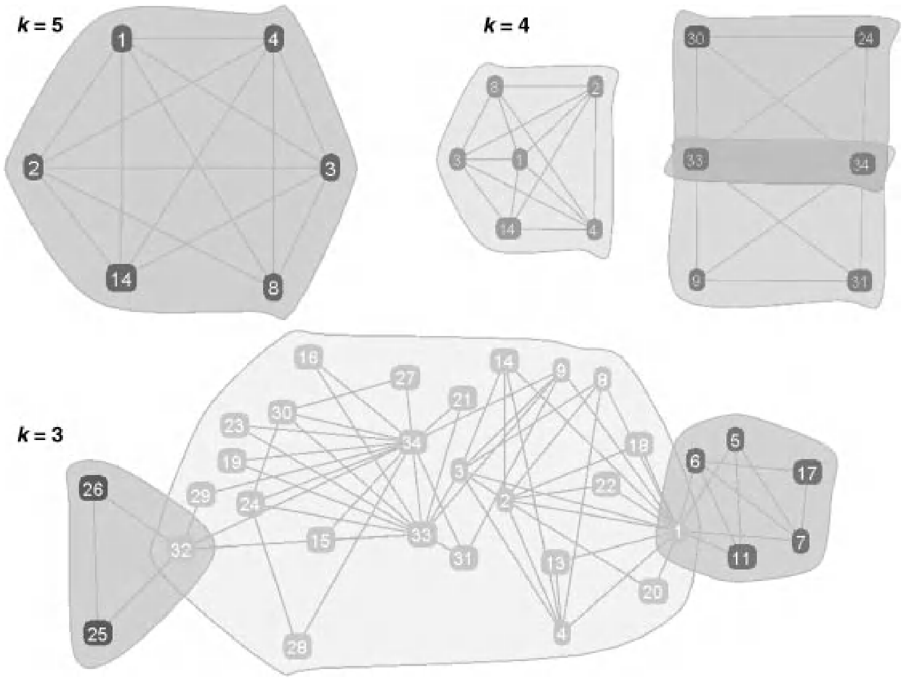


FIGURE 1.9 The partitioning of Zachary's karate club using CFinder [78]. There are one 5-community, three 4-communities, and three 3-communities. The 3-communities represent the most nodes with the exception of nodes 10 and 12. It also inaccurately places most of the opposing karate club members in a 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 $O\left(|V|^2 \log |V|\right)$. 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 $O(|V||E|)$ 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 $O(|V||E|^2)$. The algorithm is as follows:

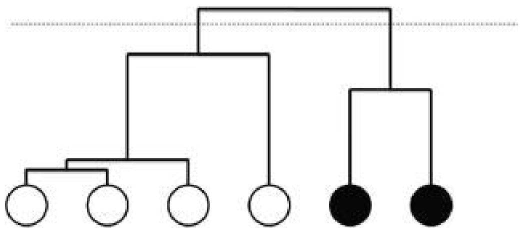


FIGURE 1.10 A dendrogram typically created by a divisive algorithm. The circles at the bottom represent the nodes of the graph. Using a top-down approach, the original graph is split until each node belongs in its own partition. The resulting number of partitions depends on where the dendrogram is *cut* . 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

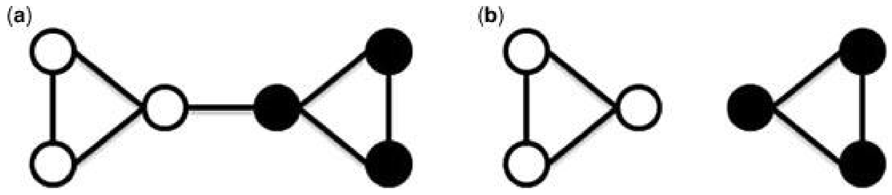


FIGURE 1.11 (a) The original graph consisting of six nodes and two communities. The central edge has the highest shortest-path betweenness score. (b) The network is divided into two communities after removal of the central edge.

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 $(1, 1, \ldots)$ 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.

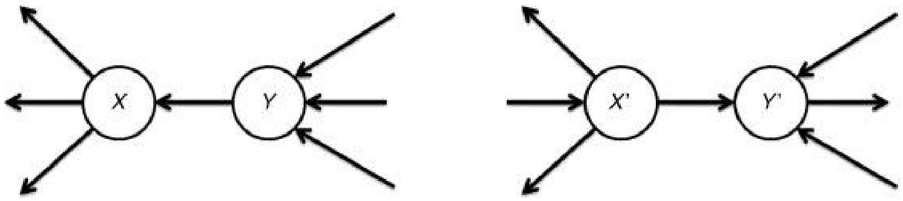


FIGURE 1.12 The two networks illustrate the problem with the directed version of modularity introduced by Leicht and Newman [79]. The in-degrees and out-degrees for nodes X and X' are the same. The same scenario holds for Y and Y'. The result is that the directed version of modularity is unable to distinguish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

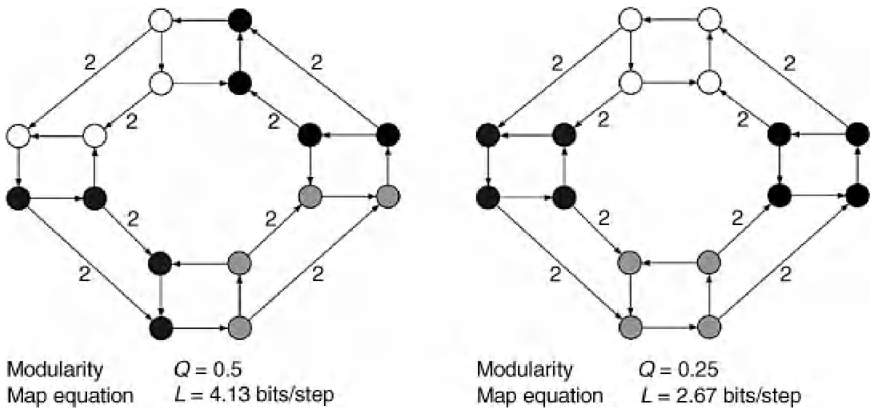

2
Modularity Q = 0.5
Map equation L = 4.13 bits/step
Modularity Q = 0.25
Map equation L = 2.67 bits/step

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

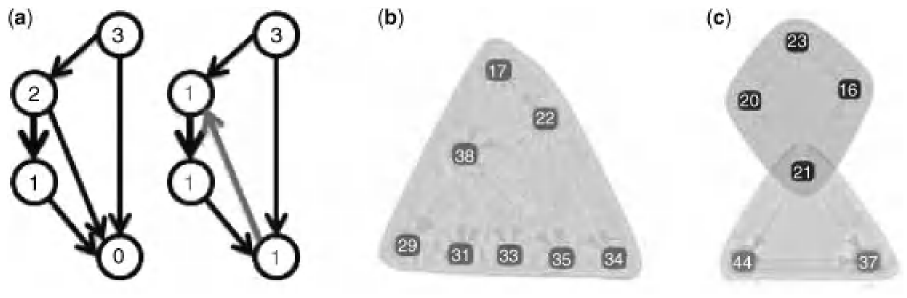


FIGURE 1.14 (a) A directed 4-clique graph without any cycles. The node labels refer to the order of the nodes which is the same as the number of edges leaving the node. (b) More than one node have the same order. The graph is not a directed 4-clique [34]. (c) The 3-communities of the *E. coli* network 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.

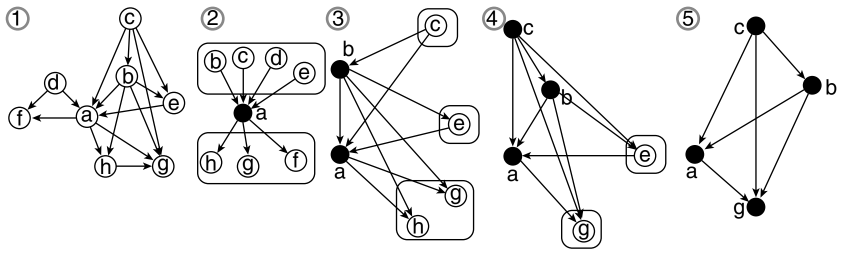


FIGURE 1.15 (1) The underlying network topology. (2) a is selected as the start node. The in-neighbors of a are placed in a container above a. The out-neighbors are placed in a container below a. (3) Select a new node from either container. In this case, b is selected. d and f are removed because they 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.

2

INTRODUCTION TO COMPLEX NETWORKS: MEASURES, STATISTICAL PROPERTIES, AND MODELS

Kazuhiro Takemoto and Chikoo Oosawa

2.1 INTRODUCTION

In real-world systems, several phenomena occur as a result of intricate interactions among several elements. Networks describe the relationships among elements, and are, thus, simple and powerful tools for describing complicated systems. The concept of networks is universal and can be applied to a wide range of fields (mathematics, computer science, economy, sociology, chemistry, biology, etc.). In recent years, considerable data on interaction has been accumulated using several new technologies and high-throughput methods. Thus, networks have become quite important for understanding real-world systems and extracting knowledge of complicated interactions.

In this chapter, we provide an overview of network science. In particular, we discuss an example of real-world networks and network representation in Section 2.2 and also mention classical network modes such as random networks and lattice networks, which are the basis of network analysis. In Section 2.4 and subsequent sections, we discuss the remarkable statistical properties of networks, and explain network measures. In addition, well-known network models are mentioned for providing a clearer understanding of the statistical properties of networks.

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

TABLE 2.1 Examples of Nodes and Edges in Networks Based on Ref. [4]

Network	Nodes	Edges
Internet	Computer or router	Cable or wireless data connection
World Wide Web	Web page	Hyperlink
Citation relationship	Article, patent, or legal case	Citation
Power grid	Generating station or substation	Transmission line
Friendship network	Person	Friendship
Metabolism	Metabolite	Metabolic reaction
Food web	Species	Predation

Several excellent books on network science have already been published (e.g., Refs. [1–4]). In addition to these books, this chapter aims to facilitate a clearer understanding of network science.

2.2 REPRESENTATION OF NETWORKS

Networks are represented as sets of nodes and edges drawn between the nodes; thus, networks are essentially similar to "*graphs*" in mathematics. Examples of networks representing real-world systems are listed in Table 2.1.

There are several types of networks (graphs); the major types are shown in Figure 2.1.

In the simplest case, networks are represented as shown in Figure 2.1a. The relationship (or interaction) between two given nodes is represented in a network by drawing edges between the nodes. In this case, multiedges, which refer to multiple edges between the same pair of nodes, and self-edges, in which the source is identical to the target, are neglected for simplicity. However, the above-mentioned edges are important in certain networks. For instance, multiedges are necessary if there are different types of interactions between the same pair. Further, self-edges are considered as self-regulations.

In the above case, we assume that the relationships are symmetric. However, the direction of a relationship is often observed in real-world systems. In food webs,

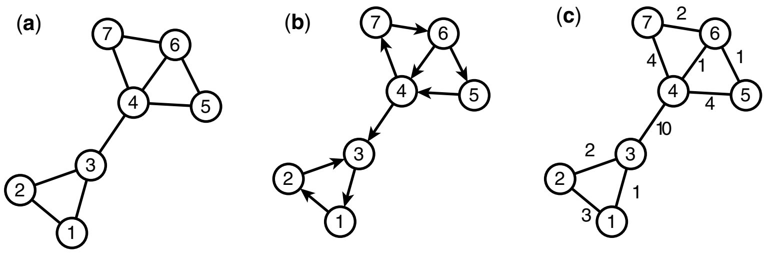


FIGURE 2.1 Examples of small networks: a simple network (a), a directed network (b), and a weighted network (c) in which the numbers along the edges correspond to the weights. The integers in circles correspond to the labels for nodes.

for example, lions prey on gazelles but gazelles never eat lions. These asymmetric interactions are represented as directed networks (see Fig. 2.1b).

In addition to direction, the weight (or strength) of edges is also important in real-world systems although it is not considered in the above networks. In the World Wide Web, for example, the weight of hyperlinks for famous sites (e.g., Google and Yahoo!) may be different from those of personal sites that are visited by only a few people. Such systems are represented as weighted graphs (see Fig. 2.1c).

As with the networks mentioned above, several types of networks are generally expressed using the adjacency matrix, where multiedges are considered as weighted edges (e.g., three edges drawn between a node pair are regarded as an edge with weight 3). The weights of unweighted edges are considered as 1.

$$A_{ij} = \begin{cases} w & \text{(If node } i \text{ connects to node } j \text{ with weight } w) \\ 0 & \text{(Otherwise)} \end{cases} \tag{2.1}$$

In simple networks (Fig. 2.1a), the edges have similar weights and $A_{ij} = A_{ji}$ and $A_{ii} = 0$ are satisfied. The representation of these networks can be well utilized due to their simplicity.

In this chapter, we consider the same simple network (i.e., Fig. 2.1a), unless specified otherwise.

2.3 CLASSICAL NETWORK

How are real-world networks constructed? Although it is difficult to answer this question, we have used ideal models to discuss network structure.

These models are not in complete agreement with real-world networks (see Section 2.4 and subsequent sections for details). However, these models are the basis of complex networks, and they contribute toward understanding network science in the future. Next, we briefly discuss some classical network models.

2.3.1 Random Network

In 1960, the Erdös–Rényi model (the so-called "random network model") was proposed by two mathematicians, Erdös and Rényi, with the assumption that networks are randomly constructed because their identities are averaged when the system has many elements [5]. This model is considered in several fields such as sociology, ecology, and mathematical biology because of its simplicity.

In addition, the random network model serves as a foundation for network analysis. Although random networks conflict with real-world networks (see Section 2.4 and subsequent sections for details), we can evaluate the significance of statistical properties observed in real-world networks on the basis of the discrepancy between random networks and real-world ones.

Model networks are generated as follows: supposing N nodes, the edges are drawn between the nodes with probability p. In the case of simple networks, the expected

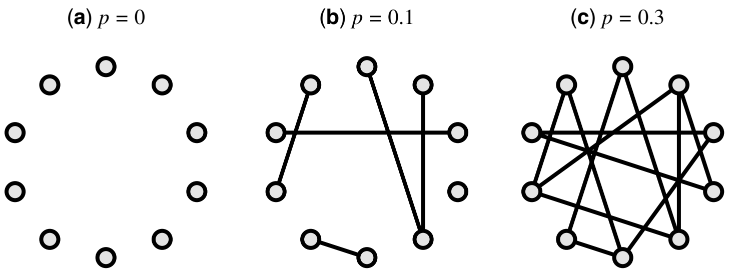


FIGURE 2.2 Schematic diagram of Erdös–Rényi networks for $p = 0$ (a), $p = 0.1$ (b), and $p = 0.3$ (c).

number of edges E is expressed as

$$E = p\binom{N}{2} = p\frac{N(N-1)}{2} \tag{2.2}$$

because the total number of possible edges (combinations) is ${}_NC_2 = N(N-1)/2$. Figure 2.2 shows a schematic diagram of these model networks. As shown in this figure, networks change with the probability p.

This probability p, which serves as the model parameter, is estimated as

$$p = \frac{2E}{N(N-1)}. \tag{2.3}$$

The number of edges E and the number of nodes N are observable in real-world networks. Thus, these parameters are used for generating hypothetical networks for real-world networks.

2.3.2 Lattice Network

Another famous network model involves lattice networks. Examples of networks of this type are shown in Figure 2.3. These networks are regularly constructed as shown in this figure; thus, they are different from random networks.

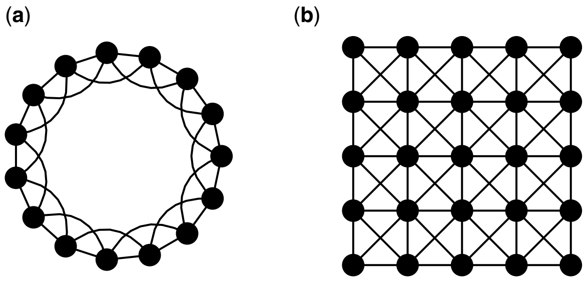


FIGURE 2.3 Examples of lattice networks. An one-dimensional lattice (a) and a two-dimensional lattice (b).

Lattice networks are relatively unsuitable for hypothetical models in network analysis because they seem to be artificial. However, they are useful when considering spatial dimensions such as distance. By assuming the regular networks in mathematical models, further, it may be easy to derive exact solutions of the models because of the regularity of lattice networks. Thus, in addition to the random network model, this model is utilized in various research fields.

2.4 SCALE-FREE NETWORK

Real-world networks possess remarkable statistical properties that are not explained by random networks and lattice networks. A representative example is the scale-free properties that relate to the distribution of node degrees.

2.4.1 Degree Distribution

The node degree, the simplest measure of a network, is defined as the number of edges (neighbors) that a node has. In the case of a simple network consisting of N nodes, in which $A_{ij} = A_{ji} = 1$, if an edge is drawn between nodes i and j, the degree of node i, k_i, is expressed as

$$k_i = \sum_{j=1}^{N} A_{ij}. \tag{2.4}$$

A simple question might arise, regarding the way in which degree is distributed in real-world networks. Barabási and Albert [6] answered this question by defining degree distribution. This distribution is defined as

$$P(k) = \frac{1}{N} \sum_{i=1}^{N} \delta(k_i - k), \tag{2.5}$$

where $\delta(x)$ is the Kronecker's delta function. This function returns 1 when $x = 0$ and returns 0 otherwise. Hence, the term $\sum_{i=1}^{N} \delta(k_i - k)$ corresponds to the number of nodes with degree k.

2.4.2 Degree Distribution of Random Network

The degree distribution of Erdös–Rényi random networks corresponds to the probability that a node has k edges. Since an edge is independently drawn between two given nodes with probability p, we can express the degree distribution as a binomial distribution:

$$P^{\text{rand}}(k) = \binom{N-1}{k} p^k (1-p)^{N-1-k}, \tag{2.6}$$

where N is the total number of nodes in the network. In addition, assuming that N is sufficiently large and $\lambda = p(N-1)$ is a constant, this binomial distribution can be rewritten as a Poisson distribution [7]

$$P^{\text{rand}}(k) \simeq \frac{\lambda^k}{k!} e^{-\lambda}, \tag{2.7}$$

which is a bell-shaped curve with a peak at $k = \lambda$. In addition to this, it is clear that lattice networks also represent degree distribution with a peak because each node has the same degree.

2.4.3 Power–Law Distribution in Real-World Networks

As mentioned above, in random networks, the node degree follows a normal distribution. Contrary to expectations, however, it was found that the degree distributions $P(k)$ of several real-world networks follow a power–law distribution,

$$P(k) \propto k^{-\gamma}, \tag{2.8}$$

where γ is a constant, the so-called "degree exponent," and is empirically found to be $2 < \gamma < 3$ (see Refs. [2,3,7–9]).

This power–law distribution indicates that a few nodes (hubs) integrate numerous nodes while most of the remaining nodes do not. Figure 2.4 shows a network with $P(k) \propto k^{-3}$. As shown in this figure, the hubs integrate the nodes with a small degree.

In such networks, the average degree is not representative although it can be calculated in networks of a finite size. Thus, networks that represent this statistical property are called "*scale-free networks*."

This statistical property is critically different from that of random networks.

The scale-free property (power–law distribution) is often used in the sense of scale invariance (i.e., self-similarity) because $f(Cx) = (Cx)^{\alpha} = C^{\alpha} f(x)$ is satisfied, where C and α are constants, when considering a power–law function $f(x) = x^{\alpha}$. In fact, Song et al. [11], using the extended box-counting method, which calculates the fractal dimension of a self-similar material, showed that real-world scale-free networks have self-similarity.

However, the relationship between self-similarity and scale-free networks has sometimes been criticized [12]. For example, we can generate a self-dissimilar network whose degree distribution follows the power–law [13]. Moreover, metabolic networks, a type of biological network, are suggested to be self-dissimilar because the degree distributions among different biosynthetic modules are very different [14]. This self-dissimilar property is called "*scale-richness*," as opposed to scale-freeness.

For the above reasons, we need to discuss the relationship between power–law distributions and self-invariance (self-similarity) carefully in the case of complex networks.

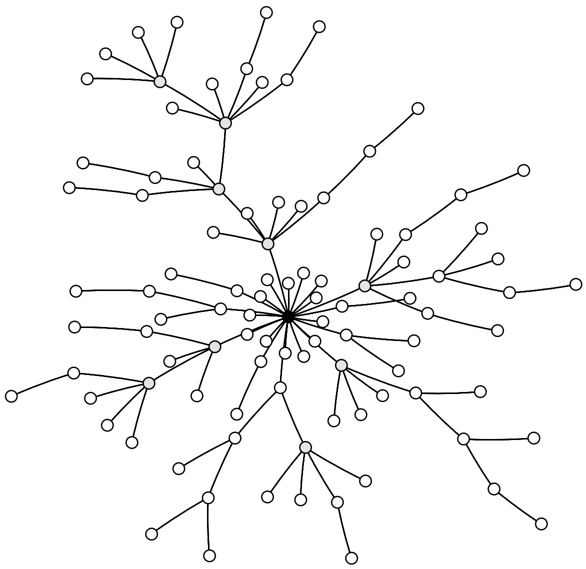


FIGURE 2.4 An artificially constructed network with $P(k) \propto k^{-3}$ ($N = 100$). The degrees of the white nodes are less than 5. The degrees of the gray nodes are greater than 5 but less than 10. The degree of the black node is greater than 10. This figure is illustrated using the yEd Graph Editor [10].

2.4.4 Barabási–Albert Model

This model was proposed by Barabási and Albert in 1999 [6] to explain scale-free networks and is called the "Barabási–Albert (BA) model."

This model has two mechanisms. One is growth, which refers to the addition of a new node and its connection to existing nodes. The other is preferential attachment, which refers to the connection of new nodes to the existing nodes with the probability

$$\Pi_i = \frac{k_i}{\sum_j k_j}. \tag{2.9}$$

Therefore, the new nodes tend to connect to large-degree nodes, resulting in the emergence of hubs.

The BA model network is generated by the following procedure.

Step (i) Starting with m_0 isolated nodes (Fig. 2.5, $t = 0$), a new node is connected to the m_0 isolated nodes (Fig. 2.5, $t = 1$).

Step (ii) Next, a new node is added and is connected to m ($\leq m_0$) existing nodes, which are selected with the probability $\Pi_i = k_i / \sum_j k_j$ (Fig. 2.5, $t \geq 2$).

Step (iii) Step (ii) is repeated until the network size reaches the target size N.

In addition, m_0 and m are fixed parameters of the algorithm.

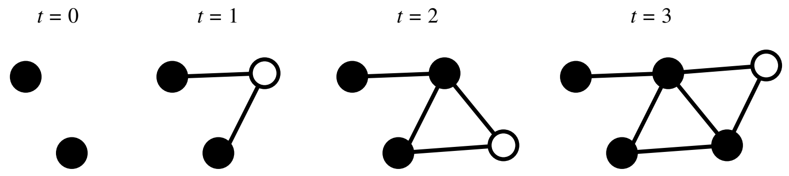


FIGURE 2.5 Formation of a Barabási–Albert network in the case of $m_0 = m = 2$. The black and white nodes correspond to existing and new nodes, respectively.

To obtain the analytical solution of the degree distribution of the BA model, mean-field based analysis [6,15,16] is used. This is a continuous approximation based on mean-field approximation, in which the many-body problem is considered as a one-body problem and which is widely used in the area of statistical mechanics. Using the mean-field analysis, we can easily obtain analytical solutions.

The analytical solution of degree distribution is derived as follows.

Assuming that the degree of node i, k_i, and time t are continuous, the time evolution of k_i is

$$\frac{\mathrm{d}}{\mathrm{d}t}k_i = m\Pi_i = m\frac{k_i}{\sum_j k_j} \tag{2.10}$$

because the degree of node i increases by m with the probability Π_i. The term $\sum_j k_j$ denotes the sum of the degrees of all nodes. Since m edges are added to the network at intervals of time t, this sum is directly derived as $\sum_j k_j = 2mt$.

Substituting this term in Equation 2.10, we have $\frac{\mathrm{d}}{\mathrm{d}t}k_i = k_i/[2t]$. The solution of this equation with the initial condition $k_i(t = s) = m$ is

$$k_i = m\sqrt{\frac{t}{s}}, \tag{2.11}$$

where s is the time when node i is added to the network; this corresponds to the number of nodes whose degrees are equal to or more than k_i. Therefore, s equals the number of nodes whose degrees are equal to or more than a given degree k. Since the total number of nodes is expressed as t, s/t is the cumulative distribution:

$$P(\geq k) = \frac{s}{t} = \frac{m^2}{k^2}. \tag{2.12}$$

Since $\frac{\mathrm{d}}{\mathrm{d}k}P(< k) = P(k)$, the degree distribution is

$$P(k) = \frac{2m^2}{k^3}, \tag{2.13}$$

demonstrating that the power–law degree distribution with the fixed degree exponent 3 is independent of the parameter m.

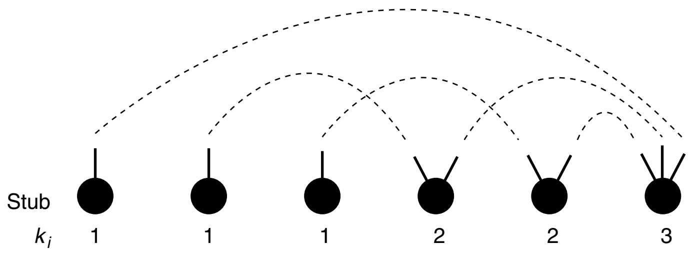


FIGURE 2.6 Schematic diagram of the configuration model. The dashed lines indicates an example of connections.

2.4.5 Configuration Model

The BA model represents the power–law degree distribution: $P(k) \propto k^{-3}$. However, this model is partially limited because the degree exponent γ obtained from real-world networks lies in the range $2 \leq \gamma \leq 3$.

When generating networks based on given degree distributions (e.g., $P(k) \propto k^{-2.2}$), the configuration model [17] is useful.

The configuration model network is generated by the following procedure (see also Fig. 2.6).

Step (i) In a set of N initially disconnected nodes, each node i is assigned a number k_i of stubs (nodes with edges unconnected with other nodes), where k_i is drawn from the probability distribution $P(k) \propto k^{-\gamma}$ to satisfy $m \leq k_i \leq N^{1/2}$ and even $\sum_i k_i$.

Step (ii) The network is constructed by randomly selecting stubs and connecting them to form edges by considering the preassigned degrees and avoiding multiple and self-connections.

Of course, we can use a given degree distribution instead of power–law degree distributions.

2.5 SMALL-WORLD NETWORK

Interestingly, in networks, the distance between a given node pair is known to be surprisingly small although the network size is very large. This property is referred to as the "*small-world property*" and was originally known as the "six degrees of separation" in sociology [18]. For example, the small-world property has been experimentally confirmed in the social network formed by the communication via internet tools such as instant-messaging systems [19].

2.5.1 Average Shortest Path Length

The distance between a node pair can be measured using the average shortest path length of a network, which is defined as

$$L = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j=1}^{N} d(i, j), \tag{2.14}$$

where $d(i, j)$ indicates the shortest path length between nodes i and j. In addition, $d(i, i) = 0$ and $d(i, j) = \infty$, if there is no shortest path between nodes i and j. Thus, the average shortest path length is only calculated in connected networks, in which there are shortest paths between all node pairs.

The ER random network model helps in explaining a small average shortest path length [7] when the probability p is not too small. When considering the breadth-first search from a node on the random network constructed with N nodes and probability p, the total number of nodes within a distance l is approximately expressed as

$$\sum_{i=0}^{l} \langle k \rangle^{i} = \frac{\langle k \rangle^{l+1} - 1}{\langle k \rangle - 1} \approx \langle k \rangle^{l}, \tag{2.15}$$

where $\langle k \rangle = p(N-1)$ and is the average degree of random networks. Equating $\langle k \rangle^{l}$ with N, we find that the average shortest path length is proportional to the logarithm of N:

$$L \approx \frac{\ln N}{\ln \langle k \rangle} \tag{2.16}$$

In fact, the average shortest path length is almost similar to that of the random network model (see Table 2.2 and Refs. 2,7,18 for details).

On the other hand, lattice networks have a large average shortest path length because they are embedded in dimensional spaces. For example, the one-dimensional lattice (Fig. 2.3a) and two-dimensional lattice (Fig. 2.3b) display $L \propto N$ and $L \propto N^{1/2}$, respectively. It is clear that the lattice networks have no small-world property.

TABLE 2.2 The Number of Nodes, Average Degree $\langle k \rangle$, Average Path Length L, and Clustering Coefficient C of Representative Real-World Networks

Network	N	$\langle k \rangle$	L	L^{rand}	C	C^{rand}
World Wide Web	153,127	35.21	3.1	3.35	0.1078	0.00023
Power grid	4,941	2.67	18.7	12.4	0.08	0.005
Metabolism	282	7.35	2.9	3.04	0.32	0.026
Food web	154	4.75	3.40	3.23	0.15	0.03

L^{rand} and C^{rand} are the expected average path length and clustering coefficient estimated from the random network model, respectively. This table is a modification of Table I in Ref. [7].

2.5.2 Ultrasmall-World Network

In addition, in the case of scale-free networks, there is a different relationship between the average shortest path length and network size. It is expected that the average shortest path length is smaller because of hubs (high-degree nodes). This relationship is characterized by the degree exponent γ. When $\gamma > 3$, $L \propto \ln N$ as random networks. However, when $2 < \gamma < 3$,

$$L \propto \ln \ln N. \tag{2.17}$$

That is, scale-free networks are more small-world than random networks, indicating an ultrasmall-world property [20].

In addition, the average shortest path length L^{nc} of random networks with a given degree distribution (e.g., the configuration model) [21] is estimated as

$$L^{\mathrm{nc}} = \frac{\ln(\langle k^2 \rangle - \langle k \rangle) - 2\langle \ln k \rangle + \ln N - \gamma_e}{\ln(\langle k^2 \rangle / \langle k \rangle - 1)} + \frac{1}{2}, \tag{2.18}$$

where $\langle \dots \rangle$ indicates the average over nodes and $\gamma_e \approx 0.5772$ is the Euler's constant. Of course, the above equation can be applied to the ER random network model and the BA model.

2.6 CLUSTERED NETWORK

2.6.1 Clustering Coefficient

Highly connected subnetworks are embedded in real-world networks by groups and modules. To measure the clustering effects, the clustering coefficient [18] was proposed. This measure denotes the density among neighbors of node i, and is defined as the ratio of the number of edges among the neighbors to the number of all possible connections among the neighbors:

$$C_i = \frac{M_i}{\binom{k_i}{2}} = \frac{2M_i}{k_i(k_i - 1)}, \tag{2.19}$$

where M_i is the number of edges among the neighbors of node i (see Fig. 2.7). The overall tendency of clustering is measured by the average clustering coefficient

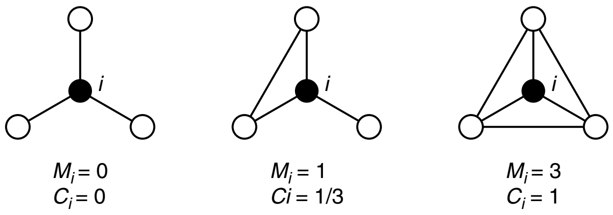


FIGURE 2.7 The clustering coefficients of node i with degree of 3 (i.e., $k_i = 3$).

$C = \frac{1}{N}\sum_{i=1}^{N} C_i$. A high average clustering coefficient implies that the network is clustered.

In random networks, since the probability that an edge is drawn between a given node pair is p, the clustering coefficient is equivalent to the probability p (the model parameter) $C^{\mathrm{rand}} = p$. Assuming that real-world networks are randomly constructed, the clustering coefficient is estimated as

$$C^{\mathrm{rand}} = p = \frac{2E}{N(N-1)} \approx \frac{\langle k \rangle}{N}, \tag{2.20}$$

where E and N are the number of edges and nodes, respectively.

However, the clustering coefficients of most real-world networks are higher than their expected value (see Table 2.2 and Refs. 2,7,18 for details), indicating that the ER random network is not clustered.

The BA model networks are also not clustered, and their clustering coefficient (see Refs. 15,16 for derivation) is

$$C^{\mathrm{BA}} = \frac{m-1}{8}\frac{(\ln N)^2}{N} \tag{2.21}$$

Since the average degree $\langle k \rangle$ is expressed as $2m$, the clustering coefficient is rewritten as

$$C^{\mathrm{BA}} \approx \frac{\langle k \rangle}{N}\frac{(\ln N)^2}{16}. \tag{2.22}$$

This clustering coefficient is a bit higher than that of random networks because of the presence of hubs. In the BA model, a newly added node is likely to connect to high-degree nodes selected by the preferential attachment mechanism. When considering the preferential selection of two nodes, the possibility that two selected nodes are connected is relatively high because these nodes have many neighbors. If the added node connects to such a node pair, the number of edges among neighbors increases, leading to higher clustering coefficients. However, the increase in clustering coefficient due to this process is very small (i.e., $(\ln N)^2$); thus, the clustering coefficient rapidly decays with increasing N.

In addition, the average clustering coefficient C^{nc} of random networks with a given degree distribution (e.g., the configuration model) [17,22] is estimated as

$$C^{\mathrm{nc}} = \frac{(\langle k^2 \rangle - \langle k \rangle)^2}{\langle k \rangle^3 N}. \tag{2.23}$$

The lattice networks are useful for explaining highly clustered networks. For example, the clustering coefficient of each node is 0.5 ($= [2 \times 3]/[4 \times 3]$) in a

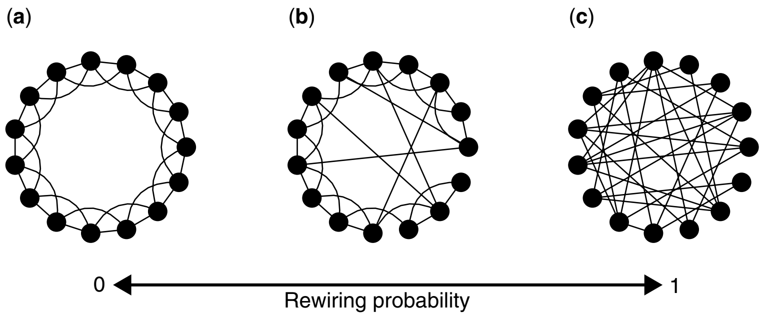


FIGURE 2.8 Schematic diagram of the Watts–Strogatz model.

one-dimensional lattice (Fig. 2.3a). This model implies that the constraint of space dimensions contributes to the network clustering observed in real-world networks.

2.6.2 Watts–Strogatz Model

The ER random network model and lattice network model describe the small-world networks and clustered networks, respectively. However, real-world networks possess both statistical properties.

To fill the gap between model networks and real-world networks, Watts and Strogatz [18] proposed a simple network model (referred to as the Watts–Strogatz (WS) model).

The WS model network is constructed as follows: (i) a one-dimensional lattice is prepared (Fig. 2.8a). (ii) A node and the edge connecting it to its nearest neighbor are selected in a clockwise sense. (iii) With the probability p, the edges are rewired and a new target node selected at random. The processes (ii) and (iii) are repeated until one lap is completed.

The model networks are similar to random networks when $p = 1$. That is, the WS model expresses the transition from lattice networks to random networks, and it includes the random network model and the lattice network models.

Small-world and highly clustered networks emerge when the probability p is in-between 0 and 1 (e.g., $0.01 < p < 0.1$). This is because the rewiring of edges is a short-cut path in a lattice network. As mentioned above, lattice networks represent a large average path length; however, the short-cut paths help decrease the average path length. Since the decay of the clustering coefficient by rewiring is slower than that of the average path length, the WS model reproduces small-world and highly clustered networks. However, the node degree follows a normal-like distribution because new target nodes are randomly selected when rewiring an edge, indicating that the WS model does not correspond to a scale-free network.

In addition, the analytical solutions of statistical properties such as clustering coefficient and average shortest path length by changing the rewiring probability p are described in Refs. [23,24]

2.7 HIERARCHICAL MODULARITY

A high clustering coefficient indicates highly interconnected subnetworks (modules) in real-world networks. However, the next question that arises is regarding the way in which these modules are organized in the network.

Ravasz et al. [25] and Ravasz and Barabási [26] provided an answer for this question by using the degree-dependent clustering coefficient.

This statistical property is defined as

$$C(k) = \frac{\sum_{i=1}^{N} C_i \times \delta(k_i - k)}{\sum_{i=1}^{N} \delta(k_i - k)}, \tag{2.24}$$

where C_i is the clustering coefficient of node i (see Eq. 2.19).

Ravasz et al. found that the degree-dependent clustering coefficient follows a power–law function in several real-world networks:

$$C(k) \propto k^{-\alpha}, \tag{2.25}$$

where α is a constant and is empirically equal to about 1.

In a random network, the clustering coefficients of all nodes are p, which is independent of the node degree k, because an edge is drawn between a given node pair with the probability p: $C^{\text{rand}}(k) = p$. Thus, the random network cannot explain this statistical property.

This power–law degree-dependent clustering coefficient indicates that the edge density among neighbors of nodes with a small degree is high and that the edge density among the neighbors of nodes with a large degree is low. In order to explain the relation between this property and the network structure, Ravasz et al. used a simple example, as shown in Figure 2.9.

In Figure 2.9, the modules a, b, and c are integrated by the hub (black node) belonging to module O. In this case, the degree of node i (k_i) is proportional to the number of modules to which node i belongs (S_i). The number of edges among the neighbors of node i (M_i) also has a linear relationship with S_i. Thus, we obtain $M_i \propto k_i$. Substituting this relation into Equation 2.19, we find the power–law relationship of degree-dependent clustering coefficients $C(k) \propto k^{-1}$. Hence, the power–law degree-dependent clustering coefficient reflects hierarchical modularity.

2.7.1 Hierarchical Model

As mentioned above, the network model in Figure 2.9 represents hierarchical modularity. Furthermore, by recursively expanding this network as shown in Figure 2.10, the model network is found to be a scale-free network.

This model network is referred to as the "hierarchical model" [25–27] and is generated by the following procedure. (i) First, we prepare a d-node module ($t = 1$ in Fig. 2.10) and place its network in G_0. (ii) We generate $d - 1$ copies of G_0. The peripheral nodes in each copy of G_0 connect to the root. At $t = 2$ (see Fig. 2.10), for

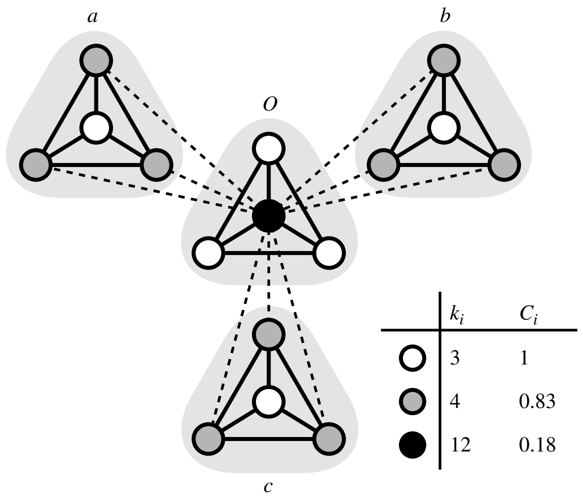


FIGURE 2.9 Schematic diagram of the hierarchical modularity. Modules are represented by the triangular-shaded regions. The character near each shaded region corresponds to the name of the module. The solid lines are the edges within a module, and the dashed lines are the edges between modules. The white nodes have 3 edges, and their clustering coefficient is 1. The gray nodes have 4 edges, and their clustering coefficient is $5/6 \approx 0.83$. The black node has 12 edges, and their clustering coefficient is $2/11 \approx 0.18$.

example, $d - 1$ nodes (excluding the central node) in each module link to the root. (iii) The network generated in Step (ii) is placed in G_0 (i.e., G_0 is updated). The model network expands by repeating Steps (ii) and (iii).

Here, we show that the degree distribution of hierarchical model follows a power law.

From the above procedure, the time evolution of the total number of nodes is d^t. When focusing on the root, the time evolution of the node degree is

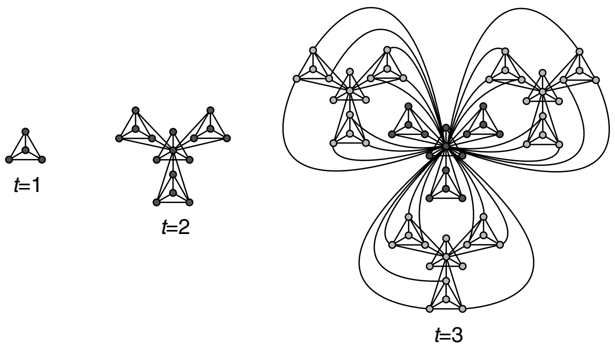


FIGURE 2.10 Schematic diagram of the hierarchical model with $d = 4$.

$\sum_{i=1}^{t}(d-1)^i = (d-1)[(d-1)^t - 1]/(d-2)$. By considering this relationship, we find that the model network has d^j nodes with degree $(d-1)[(d-1)^{t-j} - 1]/(d-2)$ or more, where j is an integer that lies in the range $0 \leq j \leq t$. Since $\ln P(\geq k) = \ln(d^j/d^t) = (j-t)\ln d$ and $\ln k \approx (t-j)\ln(d-1)$, we directly obtain $P(\geq k) = k^{-\ln d/\ln(d-1)}$. Thus, the degree distribution of the hierarchical model is

$$P(k) \propto k^{-\gamma}, \text{ where } \gamma = \frac{\ln d}{\ln(d-1)} + 1. \tag{2.26}$$

The hierarchical model network is scale-free and highly clustered. Moreover, it is also a small-world (i.e., $L \propto \ln N$) network because of its tree structure (see Ref. [28] for details).

2.7.2 Dorogovtsev–Mendes–Samukhin Model

The deterministic models such as the hierarchical model are useful because of their regularity; however, these models also have several limitations. For instance, the number of nodes cannot be closely controlled and the network is relatively artificial. In such a case, probabilistic models are more suitable. The Dorogovtsev–Mendes–Samukhin (DMS) model [29] is a simple probabilistic network model representing scale-free connectivity, hierarchical modularity, and small-world properties.

This model network is generated as follows: (i) a new node is added, and (ii) it is connected to both ends of a randomly selected edge. When an edge drawn between nodes i and j is chosen, for example, the new node k connects to nodes i and j.

This model network becomes scale-free although the preferential attachment mechanism is not directly considered. This is because the random selection of edges is essentially similar to the mechanism of preferential attachment.

We here focus on the time evolution of the degree of node i (k_i). Since the degree of node i increases when an edge leading to the node i is selected, the time evolution is expressed as $\mathrm{d}k_i/\mathrm{d}t = k_i/E$, where E is the number of edges and $E = 2t$. This equation is essentially similar to the BA model, and the degree distribution of the DMS model is $P(k) \propto k^{-3}$.

Furthermore, the number of edges among the neighbors of node i (M_i) also increases with the degree of node i because one edge leads to node i while the other edges lead to neighbors of node i. Thus, there exists a linear relationship between M_i and k_i. From this relationship, we obtain the power–law degree-dependent clustering coefficient, $C(k) \propto k^{-1}$, which indicates hierarchical modularity. Moreover, the DMS mode has a small-world property because it is almost similar to the BA model.

2.8 NETWORK MOTIF

The clustering property and hierarchical modularity indicates the organization of modules in real-world networks. However, the question is: How do we find such

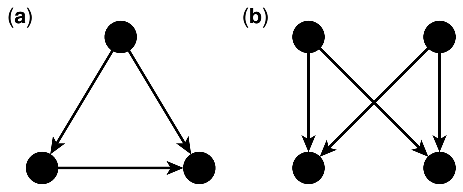


FIGURE 2.11 Example of network motifs. (a) Feedforward motif. (b) Bi-fan motif.

modules (building blocks) from real-world networks? Milo et al. [30] proposed a detection method for such modules.

It is expected that the modules and building blocks are not randomly constructed; thus, such modules are frequently observed to be more than those in random networks. For this reason, such subnetworks are referred to as "*network motifs*" [30,31]. Employing the above difference between real-world networks and random networks, we may find the modules.

This detection method focuses on the appearance frequencies of a given subnetwork (i.e., subgraph) in a real network and random networks, that is, F_{real} and F_{rand}, respectively. Random networks are generated from the real network by the randomization method, in which the terminals of two randomly selected edges are mutually exchanged at each time step: when the connected node pairs (i, j) and (m, n) are selected, we delete these edges and generate the newly connected pairs (i, n) and (m, j) (see Ref. [30] for details). We obtain the average $\langle F_{\mathrm{rand}} \rangle$ and the standard deviation SD for F_{rand} from many randomized networks generated by the above procedure. The significance of a subnetwork is evaluated by the Z-score:

$$Z = \frac{F_{\mathrm{real}} - \langle F_{\mathrm{rand}} \rangle}{SD}. \tag{2.27}$$

Thus, network motifs correspond to subnetworks with a larger Z (e.g., $Z > 2.0$, which indicates the P-value to be less than 0.05).

Several types of network motifs are found in real-world networks represented as directed networks by employing this detection method. For example, the gene regulatory networks and the neural network of nematodes possess the feedforward motif and bi-fan motif [30] (Fig. 2.11). The feedforward motif, in particular, plays an important role in gene regulatory networks [31,32].

2.9 ASSORTATIVITY

The relationship between the degrees in a connected node pair is very interesting. Since real-world networks are nonrandom, we can expect this relationship to be significant. However, degree distribution only involves the degree of each node. Thus, we need an alternative measure for characterizing such a relationship of degrees.

2.9.1 Assortative Coefficient

To characterize the relationship between node degrees, Newman proposed the assortative coefficient [33], defined as

$$r = \frac{4\langle k_i k_j \rangle - \langle k_i + k_j \rangle^2}{2\langle k_i^2 + k_j^2 \rangle - \langle k_i + k_j \rangle^2}, \tag{2.28}$$

where k_i and k_j are the degrees of two nodes at the ends of an edge, and $\langle \cdots \rangle$ denotes the average over all edges. This is simply the Pearson correlation coefficient of degrees between a connected node pair, and it lies in the range $-1 \leq r \leq 1$.

The relationship between the assortative coefficient and network structures is described as follows:

(i) In the case of $r > 0$, the network shows "*assortativity*," in which high-degree nodes tend to connect to high-degree nodes (see Fig. 2.12a).

(ii) In the case of $r = 0$, there is no correlation between the degrees in a connected node pair (see Fig. 2.12b). That is, such networks are randomly constructed, and the expected degree of neighbors of each node is constant.

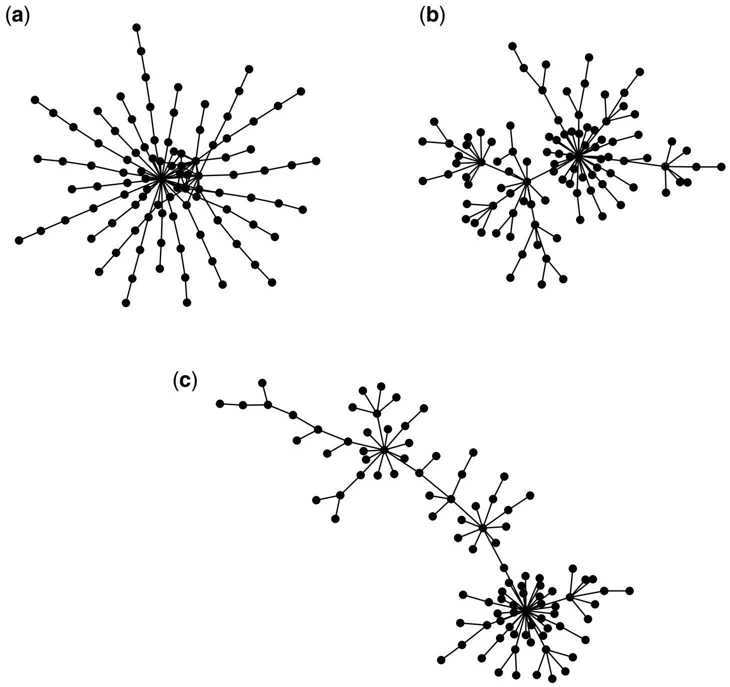


FIGURE 2.12 Schematic diagram of an assortative network (a), a random network (b), and a disassortative network (c). The networks consist of 100 nodes.

(iii) In the case of $r < 0$, the network shows "*disassortativity*," in which low-degree nodes tend to connect to high-degree nodes (see Fig. 2.12c).

It is generally known that social and technological networks exhibit assortativity and that biological and ecological networks exhibit disassortativity [33].

2.9.2 Degree Correlation

Assortativity relates to the degree correlation. The degree correlation characterizes the expected degree of the neighbors of a node with degree k and is defined as

$$\bar{k}_{nn}(k) = \frac{\sum_{i=1}^{N} \Gamma_i \times \delta(k_i - k)}{\sum_{i=1}^{N} \delta(k_i - k)}, \tag{2.29}$$

where Γ_i denotes the average nearest-neighbor degree, expressed as

$$\Gamma_i = \frac{1}{k_i} \sum_{h \in V(i)} k_h, \tag{2.30}$$

where $V(i)$ corresponds to the set of neighbors of node i.

That is, the positive and negative degree correlation indicates assortativity and disassortativity, respectively.

In many real-world networks, it is empirically known that degree correlation follows a power–law function:

$$\bar{k}_{nn}(k) \propto k^{\nu}, \tag{2.31}$$

where ν is a constant, and it also characterizes the assortativity.

The exponent ν takes values $-1 < \nu < -0.5$ in technological networks such as Internet and the World Wide Web [34,35] and biological networks such as protein–protein interaction networks [35,36] and gene regulatory networks [37]. On the other hand, a positive ν value is observed in social networks such as coauthor networks [35].

Assortativity and disassortativity is never observed in random networks. The degree correlation, Equation 2.29, is also expressed as

$$\bar{k}_{nn}(k) = \sum_{k'} k' P(k'|k), \tag{2.32}$$

where $P(k'|k)$ is the conditional probability that the edge belonging to a node with degree k points to a node with degree k' [38]. Since networks are randomly constructed, the conditional probability $P(k'|k)$ is $k' \times N_{k'}/[2E]$, where $N_{k'}$ is the number of nodes with degree k' and is equivalent to $N \times P(k')$. E denotes the number of edges. Thus, we obtain

$$\bar{k}_{nn}(k) = \sum_{k'} \frac{(k')^2 P(k')}{\langle k \rangle} = \frac{\langle k^2 \rangle}{\langle k \rangle}, \tag{2.33}$$

indicating that the degree correlation is independent from the node degree k (i.e., the network shows no assortativity). Since the random network models (i.e., the ER model and the BA model) are randomly constructed, such networks also show no assortativity although the degree distributions are different between these models.

2.9.3 Linear Preferential Attachment Model

To generate assortative and disassortative networks, the linear preferential attachment (LPA) model [39] is useful. This is an extended BA model, and the preferential attachment mechanism is modified as

$$\Pi_i^{\mathrm{LPA}} = \frac{k_i + a}{\sum_j (k_j + a)}, \tag{2.34}$$

where a is a constant and $\sum_j (k_j + a) = (2m + a)t$. This model is equivalent to the BA model when $a = 0$. When $a \to \infty$, the probability Π^{LPA} corresponds to the random attachment mechanism in which nodes are selected with the probability $1/N(t)$, where $N(t)$ is the total number of nodes at time t. High-degree nodes are more frequently chosen when $a < 0$. Since networks grow when a is larger than $-m$, which is the initial degree of a newly added node, the constant a lies in the range $-m < a < \infty$.

The degree distribution is obtained from the time evolution of the node degree, which is expressed as $\mathrm{d}k_i/\mathrm{d}t = m\Pi_i^{\mathrm{LPA}}$. Solving this equation, we have

$$k_i = (m + a)\left(\frac{t}{s}\right)^{\frac{m}{2m+a}} - a \tag{2.35}$$

where s is the time at which node i is added. Therefore, the degree distribution in the LPA model follows the power law (scale-free property):

$$P(k) \propto (k + a)^{-\gamma}, \text{ where } \gamma = 3 + \frac{a}{m}. \tag{2.36}$$

This model has a tunable degree exponent that depends on a and m, although the degree exponent of the BA model is fixed, that is, $\gamma = 3$.

The LPA model generates assortative and disassortative networks with positive a and negative $a(> -m)$, respectively.

To show this, we need to consider the time evolution of $R_i = \sum_{h \in V(i)} k_h$. In this model, there are two conditions for the increase of R_i, as follows:

(i) Node i is selected with the probability $m\Pi_i^{\mathrm{LPA}}$.
(ii) The neighbors of node i are selected with the probability $m\Pi_i^{\mathrm{LPA}}$.

In the case of (i), R_i increases by m. In the case of (ii), R_i increases by 1. Thus, the time evolution of R_i is

$$\frac{\mathrm{d}}{\mathrm{d}t} R_i = m \times m\Pi_i^{\mathrm{LPA}} + \sum_{h \in \mathcal{V}(i)} m\Pi_h^{\mathrm{LPA}} \tag{2.37}$$

The first and final terms of the above equation correspond to the increments in conditions (i) and (ii), respectively.

Substituting the solution of the above equation in Equation 2.30, the degree correlation is obtained (see Ref. 15 for derivation).The degree correlations are different with respect to the constant a as follows:

$$\bar{k}_{nn}(k) \propto \begin{cases} \ln k & (a > 0) \\ \text{Const.} & (a = 0) \\ k^{a/m} & (-m < a < 0). \end{cases} \tag{2.38}$$

As shown in Equation 2.38, the LPA model generates assortative, random, and disassortative networks depending on the parameter a. In addition, for $-m < a < 0$, it is found that the exponent of the power–law degree correlation (ν) relates to the degree exponent (γ), that is, $\nu = 3 - \gamma$.

2.9.4 Edge Rewiring Method

The rewiring based on node degrees [40] is also employed when generating assortative and disassortative networks.

The procedure is as follows. (i) Generate random networks with a given degree distribution by network models (e.g., the configuration model introduced in Section 2.4.5). (ii) Two edges are randomly selected, and the four nodes are ranked in order of their degrees (Fig. 2.13a). (iii) When generating assortative networks, the highest degree node is connected to the next highest degree nodes (Fig. 2.13b). In contrast,

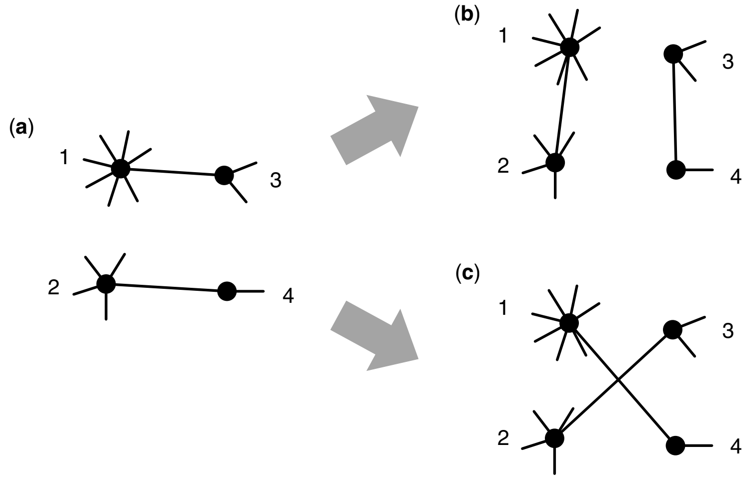


FIGURE 2.13 Schematic diagram of rewiring based on node degrees. The integer corresponds to a rank order based on node degrees. (a) Two randomly selected edges. (b) Rewiring for assortative networks. (c) Rewiring for disassortative networks.

for disassortative networks, the highest degree node is connected to the lowest degree node (Fig. 2.13c). In both cases, the other edge is drawn between the two remaining nodes.

Steps (ii) and (iii) are repeated until the rewiring of all edges is completed.

2.10 RECIPROCITY

The direction of edges is a very interesting parameter because it critically influences system dynamics, although we did not consider it in the previous sections for the sake of simplicity. In particular, the *link reciprocity* is an important measure for characterizing the significance of symmetric relationships (i.e., mutual edges) in networks.

Conventionally, link reciprocity is expressed as the ratio (e.g., [41])

$$r_d = \frac{E_d^{\leftrightarrow}}{E_d}, \tag{2.39}$$

where $E_d^{\leftrightarrow}$ and E_d correspond to the number of mutual edges and the total number of directed edges, respectively. It is clear that perfectly bidirectional and unidirectional networks show $r_d = 1$ and $r_d = 0$, respectively.

However, in order to evaluate the significance of mutual edges, the reciprocity r_d should be compared to the expected reciprocity r_d^{rand} estimated from random networks with the same number of nodes and edges. For instance, the frequent emergence of mutual edges is common in networks with the large number of edges.

To avoid this problem, Garlaschelli and Loffredo [42] proposed a novel definition of link reciprocity as the correlation coefficient between the entries of the adjacency matrix of a directed network:

$$\rho = \frac{\sum_{i \neq j}(A_{ij} - \bar{A})(A_{ji} - \bar{A})}{\sum_{i \neq j}(A_{ij} - \bar{A})}, \tag{2.40}$$

where the average value $\bar{A} = \sum_{i \neq j} A_{ij}/[N(N-1)] = E_d/[N(N-1)]$ is the ratio of observed directed edges to possible directed connections (i.e., edge density).

Since $\sum_{i \neq j} A_{ij}A_{ji} = E_d^{\leftrightarrow}$ and $\sum_{i \neq j} A_{ij}^2 = \sum_{i \neq j} A_{ij} = L$, the above equation is rewritten as

$$\rho = \frac{E_d^{\leftrightarrow}/E_d - \bar{A}}{1 - \bar{A}} = \frac{r_d - \bar{A}}{1 - \bar{A}}. \tag{2.41}$$

This equation indicates that the novel definition of link reciprocity is an extended version of the conventional definition of link reciprocity r_d.

For $\rho > 0$, the network represents link reciprocity. Mutual edges are significantly observed, indicating symmetric relationship. This property is prominently found in the world trade web (i.e., international import–export networks). This result is in agreement with the empirical knowledge that import–export relationships tend to be symmetric because international problems occur due to asymmetric relationships.

On the other hand, for $\rho < 0$, the unidirectional edges are dominant, suggesting an asymmetric relationship. Food webs, in particular, exhibit antireciprocity. This result is also consistent with our assumption. There are several examples of asymmetric relationships in food webs (e.g., wolves prey on rabbits, but rabbits never eat wolves).

2.11 WEIGHTED NETWORKS

The weight (or strength) of edges is also an important factor in complex networks. Here, we introduce statistical measures and statistical relationships in weighted networks without the direction of edges.

For a convenient explanation, we divide the adjacency matrix defined in Equation 2.1 into two matrices, b_{ij} and w_{ij}. The matrix b_{ij} corresponds to an adjacency matrix in which $b_{ij} = 1$ if there is an edge between nodes i and j, and $b_{ij} = 0$ otherwise. The weight of the edge drawn between nodes i and j is stored in w_{ij}. That is, the relationship between the original adjacency matrix A_{ij} and these matrices is $A_{ij} = b_{ij}w_{ij}$.

2.11.1 Strength

We first focus on two simple measures: the degree of node i, $k_i = \sum_{i=1}^{N} b_{ij}$ and the "*strength*" of node i [43,44], defined as

$$s_i = \sum_{i=1}^{N} b_{ij}w_{ij}. \tag{2.42}$$

In real-world weighted networks, we observe the power–law relationship between the degree k and the average strength over nodes with degree k:

$$s(k) \propto k^{\beta} \tag{2.43}$$

Assuming no correlation between the weight of edges and the node degree, the weight w_{ij} is considered as the average weight $\langle w \rangle = \sum_{ij} a_{ij}w_{ij}/[2E]$, where E denotes the total number of edges. In this case, therefore, there is a linear relationship (i.e., $s_i = \langle w \rangle k_i$), indicating $\beta = 1$.

However, the exponent β is larger than 1 in real-world weighted networks. This result is nontrivial as it indicates that the weight of edges leading to high-degree nodes (hubs) is high.

Furthermore, it was found that the average weight $\langle w_{ij} \rangle$ can be expressed as a function of the end-point degrees:

$$\langle w_{ij} \rangle \propto (k_i k_j)^{\theta} \tag{2.44}$$

Since $s_i \propto \langle w_{ij} \rangle k_i \propto k_i^{1+\theta} k_j^{\theta}$, we obtain $\beta = 1 + \theta$.

2.11.2 Weighted Clustering Coefficient

The concepts of weight and strength can be applied to the classical clustering coefficient (see also Section 2.6.1). The weighted clustering coefficient [43,44] is defined as

$$C_i^w = \frac{1}{s_i(k_i - 1)} \sum_{j,h} \frac{w_{ij} + w_{ih}}{2} b_{ij} b_{ih} b_{jh}. \tag{2.45}$$

In accordance with the classical clustering coefficient, the average weighted clustering coefficient C^w and the degree-dependent weighted clustering coefficient $C^w(k)$ are defined.

In the case where there is no correlation between weights and topology (i.e., randomized networks), $C^w = C$ and $C^w(k) = C(k)$. However, we may observe two opposite cases. If $C^w > C$, the edges with larger weights tend to form highly interconnected subnetworks such as modules. On the other hand, if $C^w < C$, such modules are likely to be formed by the edges with lower weights. Similarly, the above explanation is applicable to $C^w(k)$ for evaluating the average weighted clustering coefficient over nodes with degree k.

In real-world weighted networks [43] (e.g., international airport networks, in which nodes and weighted edges correspond to airports and flights with traffics, respectively), $C^w > C$ was observed. Furthermore, a higher weighted clustering coefficient is significant for hub nodes, that is, $C^w(k) > C(k)$ for large k. This result indicates that edges with high weights (e.g., flights with higher traffics) are densely drawn among hub nodes (e.g., airports). The scientific collaboration networks (in which the nodes and weighted edges are author and coauthor relationships, respectively, considering the number of papers) represent the same tendency.

2.11.3 Weighted Degree Correlation

Similarly, we can also consider the weighted version of degree correlation. This weighted degree correlation is obtained by modifying the original average nearest-neighbor degree Γ_i (see also Section 2.9.2). The weighted average nearest-neighbor degree [43,44] is defined as

$$\Gamma_i^w = \frac{1}{s_i} \sum_{j=1}^{N} b_{ij} w_{ij} k_j. \tag{2.46}$$

This definition implies that $\Gamma_i^w > \Gamma_i$ if the edges with larger weights are connected to the neighbors with larger degrees and that $\Gamma_i^w < \Gamma_i$ in the opposite case.

Substituting the above equation in Equation 2.29, in which Γ_i^w is referred to as Γ_i, we obtain the weighted degree correlation $\bar{k}_{nn}^w(k)$. The interpretation for magnitude relations between $\bar{k}_{nn}^w(k)$ and $\bar{k}_{nn}(k)$ is similar to that between Γ_i^w and Γ_i.

In real-world weighted networks [43] (airport networks and scientific collaboration networks), $\bar{k}_{nn}^w(k) > \bar{k}_{nn}(k)$ was observed for large k, suggesting that the edges with larger weights lead to high-degree nodes (hub nodes).

2.12 NETWORK COMPLEXITY

Real-world networks are complex and rich in variety, as explained above; a measure of the structural complexity of networks (network complexity) is therefore necessary.

The "*graph entropy*" is often used to evaluate network complexity (reviewed in Refs. [45,46]). The graph entropy of network G is based on Shannon's entropy, and it is conceptually defined as

$$\bar{I}(G) = -\sum_{i=1}^{n} \frac{|X_i|}{|X|} \log \frac{|X_i|}{|X|}, \tag{2.47}$$

where $|X|$ corresponds to a network invariant such as the total number of nodes or the total number of edges. The network is divided into n subsets, based on a given criterion, and the value $|X_i|$ denotes the cardinality of subset i. A larger $\bar{I}(G)$ indicates a higher network-variation.

Here, as a simple example, we consider the graph entropy based on the node degree [47]. Let N_k be the number of nodes with degree k; the graph entropy $\bar{I}^{\text{deg}}(G)$ is given as

$$\bar{I}^{\text{deg}}(G) = -\sum_{k=0}^{N-1} \frac{N_k}{N} \log \frac{N_k}{N}. \tag{2.48}$$

Since $N_k/N = P(k)$ (i.e., the degree distribution), this equation is rewritten as $\bar{I}^{\text{deg}}(G) = -\sum_{i=0}^{N-1} P(k) \log P(k)$. That is, the graph entropy $\bar{I}^{\text{deg}}(G)$ characterizes the degree of heterogeneity in a network. For example, random (homogeneous) networks and scale-free (heterogeneous) networks have a high $\bar{I}^{\text{deg}}(G)$ and a low $\bar{I}^{\text{deg}}(G)$, respectively.

Furthermore, we can define several graph entropies, based on different criteria, and these graph entropies are applied in a wide range of research fields (see Ref. [45] for details).

2.13 CENTRALITY

In the previous sections, we explained the global features of complex networks, such as the degree distribution and average shortest path length. However, it is also important to characterize local properties (i.e., the characteristics of each node in a complex network). For example, the clustering coefficient indicates the degree of clustering among the neighbors of a node, as explained in Section 2.6.1. In addition to the clustering coefficient, *centrality* is an important concept in network analysis because it helps in finding central (important) nodes in complex networks.

2.13.1 Definition

To date, several node centralities have been proposed, based on topological information. Well-used centralities are as follows.

Degree Centrality The degree centrality [48] is the simplest centrality measure. Assuming a correlation between the centrality (or importance) of node i and the degree of node i (k_i), the degree centrality of node i is defined as

$$C_D(i) = \frac{k_i}{N-1}, \tag{2.49}$$

where N is the network size (i.e., the total number of nodes). Since this centrality is essentially similar to the node degree, this is widely used in network analysis.

Closeness Centrality The closeness centrality [48] is based on the shortest path length between nodes i and j, $d(i, j)$. When the average path length between a node and the other nodes is relatively short, the centrality of such a node may be high. On the basis of this interpretation, the centrality of node i is expressed as

$$C_C(i) = \frac{N-1}{\sum_{j=1, j \neq i}^{N} d_{ij}}. \tag{2.50}$$

However, we cannot appropriately calculate the closeness centrality when there are isolated components resulting from unreachable node pairs. We need to assume a connected network to obtain the closeness centrality.

Betweenness Centrality As in the case of closeness centrality, the betweenness centrality [48] is based on the shortest path between nodes. However, the interpretation of centrality is a little different in these two cases.

The betweenness centrality focuses on the number of visits through the shortest paths. If a walker moves from one node to another node via the shortest path, then the nodes with a large number of visits by the walker may have high centrality. The betweenness centrality of node i is defined as

$$C_B(i) = \sum_{s \neq t \neq i} \frac{\sigma_{st}(i)}{\sigma_{st}}, \tag{2.51}$$

where $\sigma_{st}(i)$ and σ_{st} are, respectively, the number of shortest paths between nodes s and t, on which node i is located, and the number of shortest paths between nodes s and t. For normalization, the betweenness centrality is finally divided by the maximum value.

The calculation of betweenness centrality assumes a connected component, as in closeness centrality, because the betweenness centrality is also based on the shortest paths.

Eigenvector Centrality The eigenvector centrality is a higher version of degree centrality. The degree centrality is only based on the number of neighbors. However, the eigenvector centrality can consider the centralities of neighbors. Let $C_E(i)$ be the centrality of node i; $C_E(i)$ is proportional to the average of the centralities of the neighbors of node i,

$$C_E(i) = \lambda^{-1} \sum_{j=1}^{N} M_{ij} \cdot C_E(j), \tag{2.52}$$

where M_{ij} is an adjacency matrix. $M_{ij} = 1$ if node i connects to node j, and $M_{ij} = 0$ otherwise. λ is a constant. Suppose that $\boldsymbol{x} = (C_E(1), \ldots, C_E(N))$; this equation is rewritten as $\lambda \boldsymbol{x} = \boldsymbol{M} \cdot \boldsymbol{x}$. The eigenvector centrality is defined as the eigenvector with the largest eigenvalue [49].

2.13.2 Comparison of Centrality Measures

As mentioned in the previous section, the interpretation of centrality is different in these different centrality measures. To explain the differences between the centrality measures, we show an application example (Fig. 2.14).

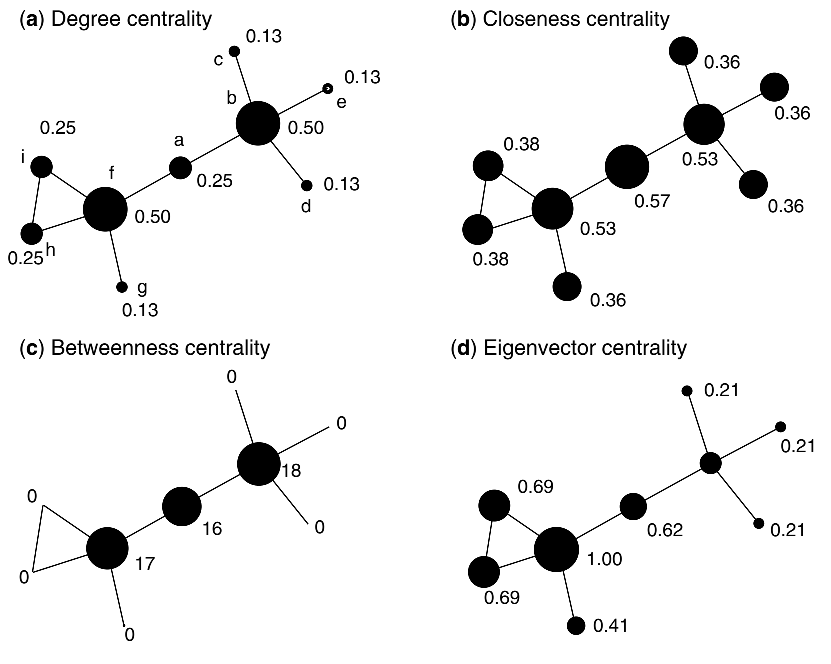


FIGURE 2.14 Comparison of centrality measures. The numerical values and lowercase letters located near nodes indicate centrality values and node identifiers, respectively. The size of a node is based on its centrality value normalized by the maximum centrality value.

Since the degree centrality is based on the node degree, high-degree nodes (e.g., nodes b and f) show high centrality. By contrast, however, node a also has high centrality in the case of both closeness centrality and betweeness centrality, although its degree is low ($k_a = 2$). Node a plays the role of an intersection between two subnetworks whose node sets are $\{b, c, d, e\}$ and $\{f, g, h, i\}$, respectively. Thus, node a may be interpreted as a central node with respect to the role of intersection. These centrality measures can find an important node because they are based on the shortest path analysis.

The results for eigenvector centrality are similar to those for degree centrality because eigenvector centrality is an extended degree centrality. In general, however, we observe high eigenvector centrality for nodes belonging to a dense subgraph (i.e., cliques) because the eigenvector centrality is determined by the centralities of its neighbors in addition to its own centrality. As shown in Figure 2.14d, the nodes belonging to the triangle consisting of nodes f, h, and i show high centralities.

In this manner, these centrality measures provide different interpretations of node centrality. In the light of these difference, we need to use centrality measures carefully.

2.14 CONCLUSION

In this chapter, we introduced several statistical properties of real-world networks, such as the scale-free property and small-world property, by comparing these networks to random networks. That is, the random network models (e.g., the ER model and configuration model) are powerful null models for network analysis. For example, we can find the network motif by comparing between real-world networks and random (null model) networks as explained in Section 2.8. Furthermore, network measures are not overestimated or underestimated by considering random network models because the models detect the bias against structural complexity.

The nonrandom structural patterns (especially scale-free property) are known to critically influence the dynamics of networks, such as epidemic spreading (e.g., [50]) and synchronization (e.g., [51]), implying the importance of network structure. Moreover, we discussed several network models and the reproduction of their structural properties. Since these models can control the tendency of structural patterns using a few parameters, they are useful to investigate the relationship between structural patterns and dynamics on networks.

The structural patterns and statistical laws help in the modeling of complex systems. For example, we may discuss the significance of unknown parameters, and simplify model representations. Furthermore, we find that network models can be used to predict missing information. The understanding of real-world networks is still not complete. For example, there may be many unknown relationships (i.e., links) in systems. Thus, the finding of missing information is the prime challenge in network science. The network models serve as a foundation for the prediction of such missing information, that is, "*link prediction*" [52].

The statistical properties and network models play important roles not only in network science but also in a wide range of research fields.

REFERENCES

1. D.J. Watts, *Small Worlds: The Dynamics of Networks Between Order and Randomness*, Princeton University Press, New Jersey, 1999.
2. S.N. Dorogovtsev, J.F.F. Mendes, *Evolution of Networks: From Biological Nets to the Internet and WWW*, Oxford University Press, Oxford, 2003.
3. R. Pastor-Satorras, A. Vespignani, *Evolution and Structure of the Internet: A Statistical Physics Approach*, Cambridge University Press, Cambridge, 2004.
4. M.E.J. Newman, *Networks: An Introduction*, Oxford University Press, Oxford, 2010.
5. B. Bollobas, *Random Graphs*, Academic Press, London, 1985.
6. A.-L. Barabási, R. Albert, Emergence of scaling in random networks, *Science* **286**, 509 (1999).
7. R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, *Rev. Mod. Phys.* **74**, 47 (2002).
8. A.-L. Barabási, Z.N. Oltvai, Network biology: understanding the cell's functional organization, *Nat. Rev. Genet.* **5**, 101 (2004).
9. R. Albert, Scale-free networks in cell biology, *J. Cell Sci.* **118**, 4947 (2005).
10. http://www.yworks.com/products/yed/
11. C. Song, S. Havlin, H.A. Makse, Self-similarity of complex networks, *Nature* **433**, 392 (2005).
12. M. Arita, Scale-freeness and biological networks, *J. Biochem.* **138**, 1 (2005).
13. L. Li, D. Alderson, J.C. Doyle, W. Willinger, Towards a theory of scale-free graphs: definition, properties, and implications, *Inter. Math.* **2**, 431 (2005).
14. R. Tanaka, Scale-rich metabolic networks, *Phys. Rev. Lett.* **94**, 168101 (2005).
15. G. Szabó, M. Alava, J. Kertész, Structural transitions in scale-free networks, *Phys. Rev. E* **67**, 056102 (2003).
16. A. Barrat, R. Pastor-Satorras, Rate equation approach for correlations in growing network models, *Phys. Rev. E* **71**, 036127 (2005).
17. M. Catanzaro, M. Boguñá, R. Pastor-Satorras, Generation of uncorrelated random scale-free networks, *Phys. Rev. E* **71**, 027103 (2005).
18. D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, *Nature* **393**, 440 (1998).
19. J. Leskovec, E. Horvitz, Planetary-scale views on a large instant-messaging network, *Proceeding of the 17th International Conference on World Wide Web*, p. 915, 2008.
20. R. Cohen, S. Havlin, Scale-free networks are ultrasmall, *Phys. Rev. Lett.* **90**, 058701 (2003).
21. A. Fronczak, P. Fronczak, J.A. Hoĺyst, Average path length in random networks, *Phys. Rev. E* **70**, 056110 (2004).
22. S.N. Dorogovtsev, Clustering of correlated networks, *Phys. Rev. E* **69**, 027104 (2004).
23. M.E.J. Newman, C. Moore, D.J. Watts, Mean-field solution of the small-world network model, *Phys. Rev. Lett.* **84**, 3201 (2000).
24. A. Barrat, M. Weigt, On the properties of small-world network models, *Eur. Phys. J. B* **13**, 547 (2000).
25. E. Ravasz, A.L. Somera, D.A. Mongru, Z.N. Oltvai, A.-L. Barabási, Hierarchical organization of modularity in metabolic networks, *Science* **297**, 1551 (2002).

26. E. Ravasz, A.-L. Barabási, Hierarchical organization in complex networks, *Phys. Rev. E* **67**, 026112 (2003).
27. A.-L. Barabási, E. Ravasza, T. Vicsek, Deterministic scale-free networks, *Physica A* **299**, 559 (2001).
28. Z. Zhang, Y. Lin, S. Gao, S. Zhou, J. Guan, Average distance in a hierarchical scale-free network: an exact solution, *J. Stat. Mech.* **2009**, P10022 (2009).
29. S.N. Dorogovtsev, J.F.F. Mendes, A.N. Samukhin, Size-dependent degree distribution of a scale-free growing network, *Phys. Rev. E* **63**, 062101 (2001).
30. R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, U. Alon, Network motifs: simple building blocks of complex networks, *Science* **298**, 824 (2002).
31. U. Alon, Network motifs: theory and experimental approaches, *Nat. Genet. Rev.* **8**, 450 (2007).
32. S. Mangan, U. Alon, Structure and function of the feed-forward loop network motif, *Proc. Natl. Acad. Sci. U.S.A.* **100**, 11980 (2003).
33. M.E.J. Newman, Assortative mixing in networks, *Phys. Rev. Lett.* **89**, 208701 (2002).
34. R. Pastor-Satorras, A. Vázquez, A. Vespignani, Dynamical and correlation properties of the internet, *Phys. Rev. Lett.* **87**, 258701 (2001).
35. A. Vázquez, Growing network with local rules: preferential attachment, clustering hierarchy, and degree correlations, *Phys. Rev. E* **67**, 056104 (2003).
36. V. Colizza, A. Flammini, A. Maritan, A. Vespignani, Characterization and modeling of protein–protein interaction networks, *Physica A* **352**, 1 (2005).
37. K. Takemoto, C. Oosawa, Modeling for evolving biological networks with scale-free connectivity, hierarchical modularity, and disassortativity, *Math. Biosci.* **208**, 454 (2007).
38. V.M. Eguíluz, K. Klemm, Epidemic threshold in structured scale-free networks, *Phys. Rev. Lett.* **89**, 108701 (2002).
39. S.N. Dorogovtsev, J.F.F. Mendes, A.N. Samukhin, Structure of growing networks with preferential linking, *Phys. Rev. Lett.* **85**, 4633 (2000).
40. A. Trusina, S. Maslov, P. Minnhagen, K. Sneppen, Hierarchy measures in complex networks, *Phys. Rev. Lett.* **92**, 178702 (2004).
41. M.Á. Serrano, M. Boguñá, Topology of the world trade web, *Phys. Rev. E* **68**, 015101(R) (2003).
42. D. Garlaschelli, M.I. Loffredo, Patterns of link reciprocity in directed networks, *Phys. Rev. Lett.* **93**, 268701 (2004).
43. A. Barrat, M. Barthélemy, R. Pastor-Satorras, A. Vespignani, The architecture of complex weighted networks, *Proc. Natl. Acad. Sci. U.S.A.* **101**, 3747 (2004).
44. A. Barrat, M. Barthélemy, A. Vespignani, Modeling the evolution of weighted networks, *Phys. Rev. E* **70**, 066149 (2004).
45. M. Dehmer, A. Mowshowitz, A history of graph entropy measures, *Inform. Sci.* **181**, 57 (2011).
46. G. Simonyi, Graph entropy: a survey, in *Combinatorial Optimization* (W. Cook, L. Lovász, P. Seymour, eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 20, p. 399, 1995.
47. N. Rashevsky, Life information theory and topology, *Bull. Math. Biophys.* **17**, 299 (1955).
48. L.C. Freeman, Centrality in social networks: conceptual clarification, *Soc. Netw.* **1**, 215 (1979).

49. P. Bonacich, Some unique properties of eigenvector centrality, *Soc. Netw.* **29**, 555 (2007).
50. R. Pastor-Satorras, A. Vespignani, Epidemic spreading in scale-free networks, *Phys. Rev. Lett.* **86**, 3200 (2001).
51. Y. Moreno, A.F. Pacheco, Synchronization of Kuramoto oscillators in scale-free networks, *Euro. Phys. Lett.* **68**, 603 (2004).
52. L. Lu, T. Zhou, Link prediction in complex networks: a survey, arXiv/1010.0725 (2010).

3

MODELING FOR EVOLVING BIOLOGICAL NETWORKS

Kazuhiro Takemoto and Chikoo Oosawa

3.1 INTRODUCTION

The biomolecules of the living organisms, such as proteins and metabolites, undergo several interactions and chemical reactions, which lead to the occurrence of various life phenomena [1–6]. These interactions can be represented in the form of networks (graphs) (see Fig. 3.1, for example), the so-called "biological networks"; the evolution of these biological networks is a long-standing question. It is believed that biological networks adaptively shape-shift with the changing environment (e.g., temperature, pressure, and radial ray), and consequently, living organisms can perform new functions. Thus, for understanding life phenomena, it is important to obtain an understanding of network structures and their formation mechanisms from a macroscopic viewpoint. In addition to this, such a network approach also plays a significant role in technological processes such as finding missing interactions and designing novel interactions.

With recent developments in biotechnology and bioinformatics, the understanding of the interactions among biomolecules is progressively becoming clearer. The data of the interactions are accumulated in several databases. For example, the Kyoto Encyclopedia of Genes and Genomes (KEGG) [7] is a famous database in which metabolic pathways of many living organisms are available. Moreover, we can obtain the large-scale data for protein–protein interactions from the Database of Interacting Proteins (DIP) [9]. Hence, we can now observe large-scale biological networks using several databases.

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

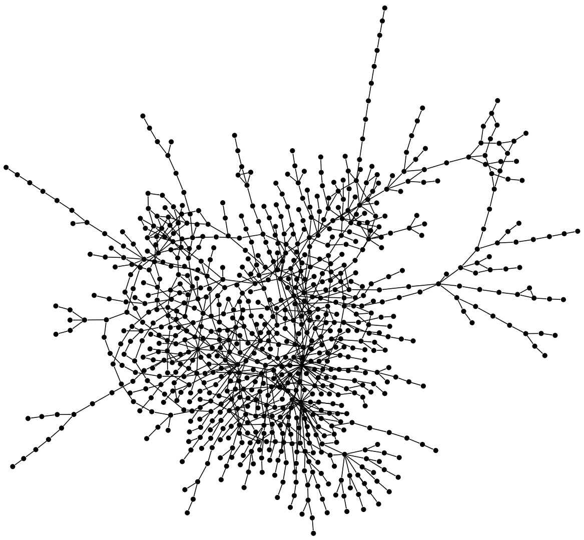


FIGURE 3.1 The partial metabolic network of *Escherichia coli* obtained from the KEGG database [7]. The nodes (filled circles) and edges (links) represent the metabolites and substrate–product relationships. This network was drawn using the yEd Graph Editor [8].

In recent years, the analyses of large-scale biological networks constructed from several databases are actively being conducted using graph-theoretical metrics. As a result, universal structural properties such as heterogeneous connectivity have been found in several types of biological networks in several organisms (e.g., reviews in Refs. [1,2,10]). This finding implies that biological networks grow according to universal design principles. However, the growth of biological networks cannot be directly observed because the databases only provide static networks. Therefore, theoretical approaches that employ models are useful for revealing formation mechanisms (rules) of biological networks.

Since the conception of the problem, the evolution of networks has been theoretically discussed using models. The most famous models are the Erdös–Rényi (ER) random network [11] and the Barabási–Albert (BA) model [12], which are used as ideal models in widely ranging fields from sociology to physics. However, the structure indicated by these models is very different from real biological networks. Thus, a new model is required to realize a more accurate representation of real networks. The theory for evolution of biological networks is still insufficient; therefore, we need to propose an alternative theory (model).

In this chapter, we introduce simple evolving network models that are in good agreement with real biological networks and their mathematical frameworks.

This chapter is divided into three main sections. In Section 3.2, we propose a unification of different models, which explains the structural properties of real biological networks. We simplify the complicated models. Furthermore, with regard to simplification, the network model without parameter tunings is considered in most existing models in Section 3.3. These models describe unipartite networks. However, the biological networks are often represented as bipartite networks such as metabolite distributions (species–metabolite networks). Thus, we also represent a bipartite network model in Section 3.4.

3.2 UNIFIED EVOLVING NETWORK MODEL: REPRODUCTION OF HETEROGENEOUS CONNECTIVITY, HIERARCHICAL MODULARITY, AND DISASSORTATIVITY

As mentioned above, several remarkable structural properties have thus far been characterized in biological networks via network analysis.

A well-known feature of networks is *heterogeneous connectivity*, which indicates that the node degree k (the number of edges that the node has) approximately follows a power–law distribution: $P(k) \propto k^{-\gamma}$. The exponent γ, called the "degree exponent," is empirically known to be between 2 and 3. This power–law degree distribution implies that a few nodes (hubs) integrate numerous nodes while most of the remaining nodes do not; this feature is clearly different from that of the Poisson (homogeneous) distribution obtained from classical (ER) random networks. To reproduce heterogeneous connectivity, the BA model was proposed. This model has two mechanisms: the growth mechanism, in which an added node connects to existing nodes, and the preferential attachment (PA) mechanism, in which the existing node i is selected with the probability $\Pi_i = k_i / \sum_j k_j$, where k_i is the degree (the number of edges) of node i. The BA model generates heterogeneous networks with the power–law degree distribution $P(k) \propto k^{-3}$ [12].

The other structural property of networks is *hierarchical modularity*. Most real-world networks are clustered; this is characterized by the clustering coefficient C. The clustering coefficient of node i (C_i) is defined as $2M_i/[k_i(k_i - 1)]$, where M_i is the number of edges among the neighboring nodes of node i. In real-world networks, there exists a power–law relationship between C_i and k_i with the exponent -1, that is, $C_i \propto k_i^{-1}$; the ER model and the BA model do not represent this relationship because the network is randomly constructed with a given degree sequence. This power–law relationship is reproduced by the hierarchical organization of modules (small dense networks) [13]; thus, it is referred to as hierarchical modularity. Furthermore, the triad formation [14] and the merging module [15] also reproduce this property. Although there are several models, the most important aspect is the proportional relationship between the degree of node i and the number of modules including node i (the number of edges among the neighbors of node i).

In addition, *disassortativity* [16] is also a well-known property. The correlation of the degree between a node and its neighboring node (hereafter referred to as "degree correlation") is useful for explaining disassortativity. Disassortativity indicates negative degree correlations. The ER model and the BA model indicate no such correlation because their networks are randomly constructed. In such a random network with a given degree sequence, the probability of transition from nodes with degree k to nodes with degree k', $P(k'|k)$, is $k'P(k')N/(2E)$ [17]. Thus, the expected degree of neighboring nodes is $\sum_{k'} k' P(k'|k) = \langle k^2 \rangle / \langle k \rangle$, where $\langle \cdots \rangle$ denotes the average over all nodes. This result indicates that the degree correlation is independent of degree k (i.e., there is no correlation).

This is particularly caused in the BA model due to the preferential attachment in which a node (node i) gets new edges with the probability Π_i that is proportional to its own degree, that is, $\Pi_i = k_i / \sum_j k_j$. Due to this preferential attachment, the transition probability is almost similar to the above equation.

We cannot neglect preferential attachment because this mechanism provides heterogeneous connectivity. To solve this problem, we need to consider information other than the degree. This information is based on studies on competition dynamics [18] and weighted networks [19,20], which discuss the effects of fitness on a network structure. Here, we also refer to this information as *fitness*. By considering this information, the networks have negative degree correlations [21,22].

As mentioned above, these different structural properties are explained by different models; thus, the theory for network evolution is complicated. For a simple understanding of this theory, we need to construct a new convenient model that includes several models. In this section, we introduce a simple unified model for reproducing heterogeneous connectivity, hierarchical modularity, and disassortativity.

3.2.1 Network Model

Our model includes growth by merging modules and the fitness-driven (FD) mechanism (see Ref. 23 for biological implications of this model).

Growth by Merging Modules. A network grows by the merging of new modules to the existing nodes of a network over time (see Fig. 3.2c). m/a is the first control parameter of the model where a and m denote the number of nodes of the complete graph to be merged in and the number of merged node(s), respectively. It should be noted that the process develops without adding extra edges [15].

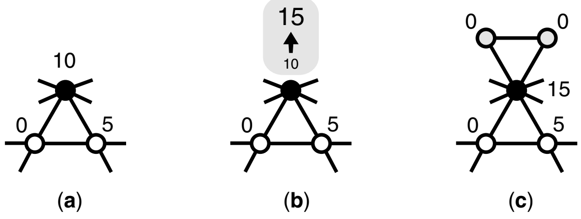


FIGURE 3.2 Schematic diagram of the growth process of our model network with $a = 3$, $m = 1$, and $\xi = 5$. (a) Selection of node(s) by the FD–PA mechanism, Equation 3.1. The filled node is selected by the FD–PA mechanism. Each number in the figure indicates the fitness of a corresponding node. (b) Updating of the fitness. The selected node's fitness increases according to the updating rule, Equation 3.2. (c) Merging new modules. As a result, $(a - m)$ new node(s) are added and filled with gray, their initial fitness is considered as zero.

Fitness-Driven Preferential Attachment. The standard PA mechanism of the BA model is the probability Π_i that node i is chosen to get an edge and is proportional to the degree of node i; hence, $\Pi_i = k_i / \sum_j k_j$, where k_i is the degree of node i. The mechanism only considers the degrees at the nodes. Here, we additionally consider the probability that node i is selected according to degree k_i and fitness f_i; we express the probability as

$$\Pi_i^* = \frac{k_i + f_i}{\sum_j (k_j + f_j)}. \tag{3.1}$$

Updating Rule of Fitness. Moreover, we consider the change in the fitness of node i. When node i is selected using the FD–PA mechanism given in Equation 3.1, the fitness of node i increases as follows:

$$f_i \leftarrow f_i + \xi, \tag{3.2}$$

where ξ takes a constant value and is the second control parameter in the model, indicating the strength of incidence of the fitness.

Taken together, our model networks are generated by the following procedures.

(i) We start from a module that is a complete graph consisting of a (≥ 3) nodes. For fitness, we assign zero to all nodes in the module.
(ii) At every time step, a new module of the same size is merged to the existing m ($< a$) nodes. The number of network nodes increases by $(a - m)$ after the merger of module into the existing network.
(iii) When merging the module, the FD–PA mechanism, Equation 3.1, is used to select m old nodes (see Fig. 3.2a). The fitnesses of the old nodes are then increased using the updating rule, Equation 3.2 (see Fig. 3.2b). Finally, zeros are assigned to the fitnesses of the new nodes (see Fig. 3.2c). When merging modules, the multiple edges between nodes selected by the FD–PA mechanism may occur. Such edges between the selected nodes are counted, and they contribute to the FD–PA mechanism in the subsequent steps.

When $a > m$ and $\xi \geq 0$, the model network evolves in time steps. In addition, our model is equivalent to the BA model with the specific condition: $a = 2$, $m = 1$, and $\xi = 0$.

3.2.2 Degree Distribution

In this section, we present the analytical and numerical solutions for the degree distribution of our model.

In order to describe the degree distribution, we employ the mean-field-based analysis [12,22]. The standard approach cannot be directly applied due to the inclusion of the fitness updating. We express the degree and fitness as

$$F_i = k_i + f_i \tag{3.3}$$

and investigate the time evolution of F_i. Since the fitness updating rule is represented as $f_i \leftarrow f_i + \xi$, F_i satisfies

$$F_i = \left(\frac{\xi}{a-1} + 1\right) k_i - \xi, \tag{3.4}$$

indicating the proportional relationship between F_i and k_i.

The time evolution of F_i is described as

$$\frac{\mathrm{d}F_i}{\mathrm{d}t} = m(a - 1 + \xi)\frac{F_i}{\sum_j F_j}, \tag{3.5}$$

where $\sum_j F_j \approx [a(a-1) + m\xi]t$. The solution of the equation with $F_i(t = s) = \binom{a}{2} + \xi = A(a, \xi)$ as an initial condition for Equation 3.5 is

$$F_i(t) = A(a, \xi) \left(\frac{t}{s}\right)^{\beta}, \tag{3.6}$$

where $\beta = [m(a - 1 + \xi)]/[a(a-1) + m\xi]$. Because s/t denotes the probability that F_i is larger than a given F, Equation 3.6 is rewritten as

$$P(\geq F) = A(a, \xi)^{1/\beta} F^{-1/\beta}, \tag{3.7}$$

which corresponds to a cumulative probability. From Equation 3.7, the probability distribution for F is given as

$$P(F) = -\frac{\mathrm{d}}{\mathrm{d}F} P(\geq F) = \frac{A(a, \xi)^{1/\beta}}{\beta} F^{-\gamma}, \tag{3.8}$$

where $\gamma = (1/\beta) + 1$. Finally, substituting Equation 3.4 into Equation 3.8, we obtain the degree distribution

$$P(k) \simeq B(a, \xi, \gamma) k^{-\gamma}, \tag{3.9}$$

where $B(a, \xi, \gamma) = A(a, \xi)^{\gamma-1}[\xi/(a-1) + 1]^{-\gamma}/(\gamma - 1)$. The degree distribution obeys the power law with the degree exponent

$$\gamma = \frac{(a + m)(a - 1) + 2m\xi}{m(a - 1 + \xi)}, \tag{3.10}$$

demonstrating that our model network contains the scale-free feature.

In order to confirm the analytical predictions, we performed numerical simulations of networks, generated using our model. Figure 3.3 shows degree distributions with $a = 5$, $N = 50,000$, and different values of $m \in [1, 4]$ and $\xi \in [0, 15]$. These degree distributions follow the power law, reflecting the scale-free feature.

Figure 3.3a shows the degree distributions with $m = 2$ and different values of ξ. The degree exponents decay with increasing ξ. Figure 3.3b shows the degree distributions with $\xi = 7$ and different values of m. The degree exponents decay with increasing m. The numerical results and theoretical predictions are in excellent agreement, demonstrating that γ is a function of m/a and ξ.

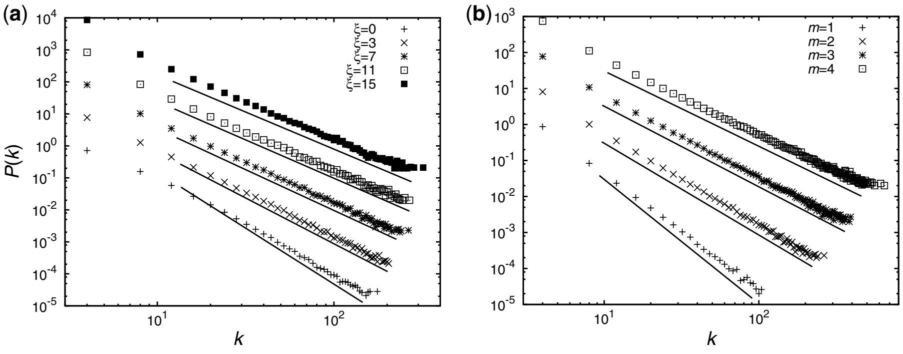


FIGURE 3.3 Degree distributions $P(k)$ with $a = 5$ and $N = 50,000$ (shifted for clarity). Different symbols correspond to different numerical results. Solid lines represent the relationship $P(k) \propto k^{-\gamma}$, where γ is predicted by Equation 3.10. (a) ξ dependency with $m = 2$. (b) m/a dependency with $\xi = 7$.

3.2.3 Degree-Dependent Clustering Coefficient

In this section, we show the analytical and numerical solutions of the degree-dependent clustering coefficient.

First, we provide the analytical solution for the degree-dependent clustering coefficient of our model. Since our model grows due to the merging of modules (see Fig. 3.2), the number of edges among neighbors of node i is approximately described [15] as

$$M_i \simeq S_i \binom{a-1}{2} = S_i(a-1)\frac{a-2}{2}, \tag{3.11}$$

where S_i corresponds to the number of selections of node i with the PA, as in Equation 3.1. In our model, because the degree of node i is expressed as $k_i = S_i(a-1)$, Equation 3.11 is rewritten as

$$M_i \simeq \frac{a-2}{2} k_i, \tag{3.12}$$

indicating the proportional relationship between M_i and k_i. Finally, substituting Equation 3.12 into the definition of the clustering coefficient (i.e., $C_i = 2M_i/[k_i(k_i-1)]$), we obtain the degree-dependent clustering coefficient

$$C(k) \simeq \frac{a-2}{k} \propto k^{-1}. \tag{3.13}$$

The degree-dependent clustering coefficient follows the power law with the exponent -1 when $a \geq 3$, reflecting the hierarchical modularity of our model network.

Next, we present the numerical results of the degree-dependent clustering coefficient of our model in order to verify the analytical solution. In Figure 3.4, we show

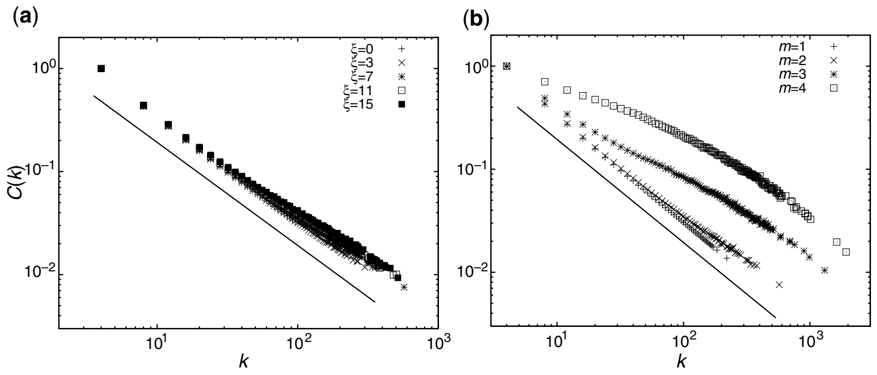


FIGURE 3.4 Degree-dependent cluster coefficient $C(k)$ with $a = 5$ and $N = 50,000$. Different symbols correspond to the different numerical results. Solid lines represent the relationship $\propto k^{-1}$. (a) ξ dependency with $m = 2$. (b) m dependency with $\xi = 7$.

the degree-dependent clustering coefficient with $a = 5$, $N = 50,000$, and different values of $m \in [1, 4]$ and $\xi \in [0, 15]$.

Figure 3.4a shows the degree-dependent clustering coefficient with $m = 2$ and different values of ξ. $C(k)$ follows the power law with the exponent -1 despite the varying ξ values. Figure 3.4b shows degree-dependent clustering with $\xi = 7$ and different m values. Moreover, $C(k)$ follows the power law with the exponent -1 for a large value of k. For a small value of k, the cut-off becomes more prominent with increasing m as it moves further away from the approximation of Equation 3.11.

The degree-dependent clustering coefficient follows the power law with the exponent more or less equal to -1, reflecting the hierarchical modularity of networks.

3.2.4 Average Clustering Coefficient

To examine the parameter-dependency of the modularity (clustering property) for the entire network, we employ an average clustering coefficient C. The coefficient is defined as $C = \frac{1}{N}\sum_{i=1}^{N} C_i$. A high value of C implies that the network has high modularity.

Figure 3.5 shows the numerical result of C in our model network with $a = 5$, $N = 12,800$, and different values of $m \in [1, 4]$ and $\xi \in [0, 15]$. C tends to increase by a greater extent for smaller m/a and larger ξ. Since the small and large m/a lead to different effects on C, Figure 3.5a shows a valley in the middle of m/a. In the case of small m/a, C can remain high because the modules combine via a few common nodes. In contrast, in the case of large m/a, C is low because the networks tend to be randomized, which is due to the fact that the modules combine via many common nodes. In this case, however, the FD mechanism helps to increase the fitness of the nodes in the modules, inducing the formation of a cluster with high-edge density. As a result, C increases with m/a. Due to the trade-off mechanism, a valley-shaped curve is obtained. In addition, C is less sensitive for large ξ.

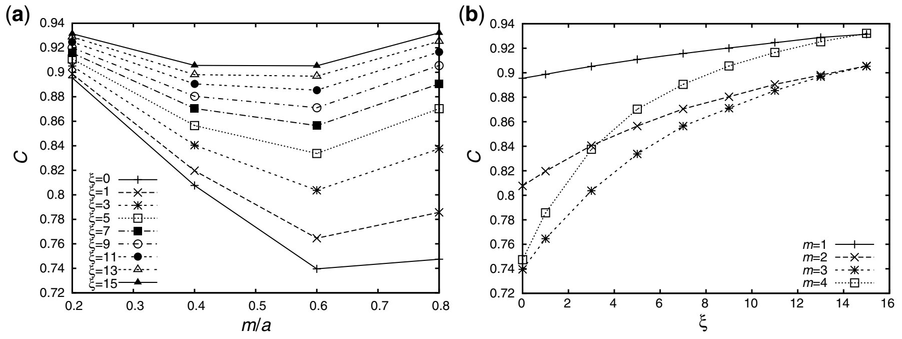


FIGURE 3.5 The dependency of the average clustering coefficient C ($a = 5$ and $N = 12,800$) on the parameter (a) m/a and (b) ξ.

3.2.5 Degree Correlation

The degree correlation is a structural property that characterizes assortativity of the networks and represents the average degree of the neighbors of nodes with degree k. This correlation is defined as

$$\bar{k}_{nn}(k) = \frac{\sum_{i=1}^{N} \Gamma_i \times \delta(k_i - k)}{\sum_{i=1}^{N} \delta(k_i - k)}, \tag{3.14}$$

where $\delta(x)$ is Kronecker's delta function. This function returns 1 when $x = 0$ and returns 0 otherwise. Γ_i denotes the average nearest-neighbor degree and is expressed as

$$\Gamma_i = \frac{1}{k_i} \sum_{h \in V(i)} k_h, \tag{3.15}$$

where $V(i)$ corresponds to the set of neighbors of node i.

Here, we provide the numerical solutions for the degree correlation of our model. Figure 3.6 shows the degree–degree correlations with $a = 5$, $N = 50,000$, and different values of m and ξ. The correlations follow the power law; $\bar{k}_{nn}(k) \propto k^{\nu}$ with $-1 < \nu < 0$ as a rough observation. Negative and larger values of ν are observed for larger m/a and ξ.

Figure 3.6a shows the degree correlations with $m = 2$ and different ξ values. The exponent ν decays with increasing ξ. Figure 3.6b shows the degree–degree correlations with $\xi = 7$ and different m values. The exponent ν decays with increasing m.

Due to the fitness updating and the PA mechanism, the degree correlations follow the power law, reflecting the disassortativity of the networks. As previously reported, disassortativity is not reproduced if we only consider the PA mechanism [15,21].

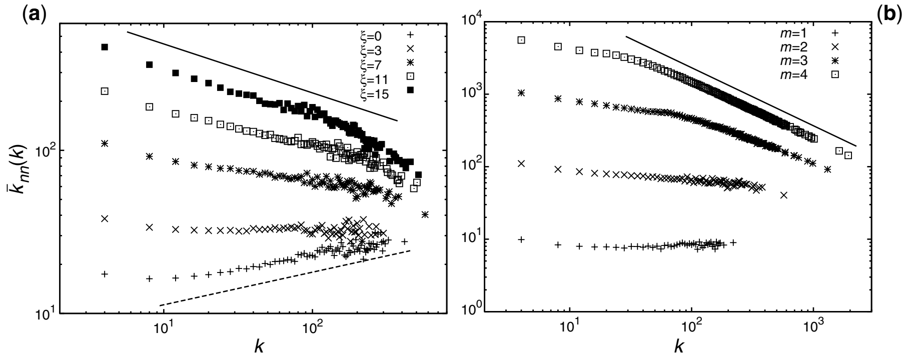


FIGURE 3.6 Degree correlations $\bar{k}_{nn}(k)$ with $a = 5$ and $N = 50,000$. (a) ξ dependency with $m = 2$. The solid and dashed lines correspond to $\propto k^{-0.3}$ and $\propto k^{0.2}$, respectively. (b) m dependency with $\xi = 7$. The solid line represents the relationship $\propto k^{-0.8}$.

3.2.6 Assortative Coefficient

The assortative coefficient (AC) [16] can be considered to be a compendium parameter of the degree correlations and is defined as

$$r = \frac{4\langle k_i k_j \rangle - \langle k_i + k_j \rangle^2}{2\langle k_i^2 + k_j^2 \rangle - \langle k_i + k_j \rangle^2}, \tag{3.16}$$

where k_i and k_j are the degrees of two nodes at the ends of an edge and $\langle \cdots \rangle$ denotes the average over all edges. In other words, the AC is the correlation coefficient for the degree correlation $\bar{k}_{nn}(k)$ and takes the values $-1 \le r \le 1$. The relationship between the AC and the network structures is described as follows:

(i) In the case of $r < 0$, low-degree nodes tend to connect to high-degree nodes, indicating disassortativity. Hence, the degree correlation $\bar{k}_{nn}(k)$ decreases with increasing k.

(ii) In the case of $r = 0$, the degree–degree correlation $\bar{k}_{nn}(k)$ is not observed.

(iii) In the case of $r > 0$, high-degree nodes tend to connect to high-degree nodes, reflecting assortativity. Hence, the degree correlation $\bar{k}_{nn}(k)$ increases with degree k.

Figure 3.7 shows the numerical solutions of the AC for our model with $a = 5$, $N = 12,800$, and different values of $m \in [1, 4]$ and $\xi \in [0, 15]$. For larger m/a and ξ values, the large negative AC is generally observed. For smaller $\xi \in [0, 1]$, on the other hand, the valley-shaped curves (once, r takes positive instead of negative values) emerge because positive degree correlations are exhibited in the case of $m/a \le 0.5$ [15]. For larger $\xi \in [3, 15]$, assortative coefficients r monotonically decrease (upward curve) with increasing m/a.

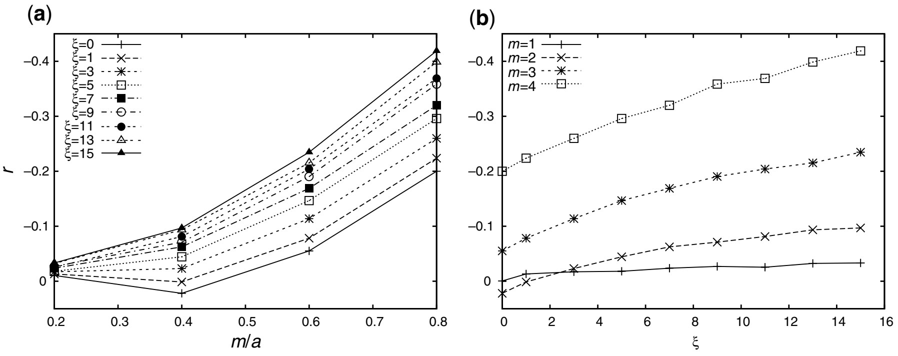


FIGURE 3.7 Assortative coefficient r with $a = 5$ and $N = 12,800$. The dependency on the parameter m/a (a) and ξ (b). The vertical axes of (a) and (b) are inverted for clarity.

3.2.7 Comparison with Real Data

In order to validate our model, we compare the structural properties of biological networks with those of our model networks. We construct two different types of networks: the gene regulatory [24] and the metabolic [25] networks of *E. coli*. The gene regulatory network is represented as a set of graphs consisting of nodes and edges, which correspond to genes and the interactions among genes, respectively. For simplicity, we extract the largest component from the networks and replace the directed and/or weighted edges with undirected and/or unweighted edges. Moreover, we remove the multiple edges and self-loops. The metabolic network is transformed by the same procedures.

In Figure 3.8, we show the structural properties of the biological networks and our model networks. Our model is found to be in good agreement with the data of the real network, demonstrating that our model reproduces the three previously mentioned remarkable structural properties that are widely shared among biological networks.

3.3 MODELING WITHOUT PARAMETER TUNING: A CASE STUDY OF METABOLIC NETWORKS

In the previous section, we introduced a simple model for reproducing several structural properties observed in real-world biological networks. Furthermore, this model network shows good qualitative and quantitative agreement with real data.

For model fitting, we need to tune model parameters. The parameters may be estimated by minimizing the difference between the model and real data using statistical methods. However, the optimal solutions may require substantial calculation, after which multiple optimal solutions may be obtained. Furthermore, the possibility of trivial agreements (e.g., over-fittings) may remain if the model has several tunable parameters.

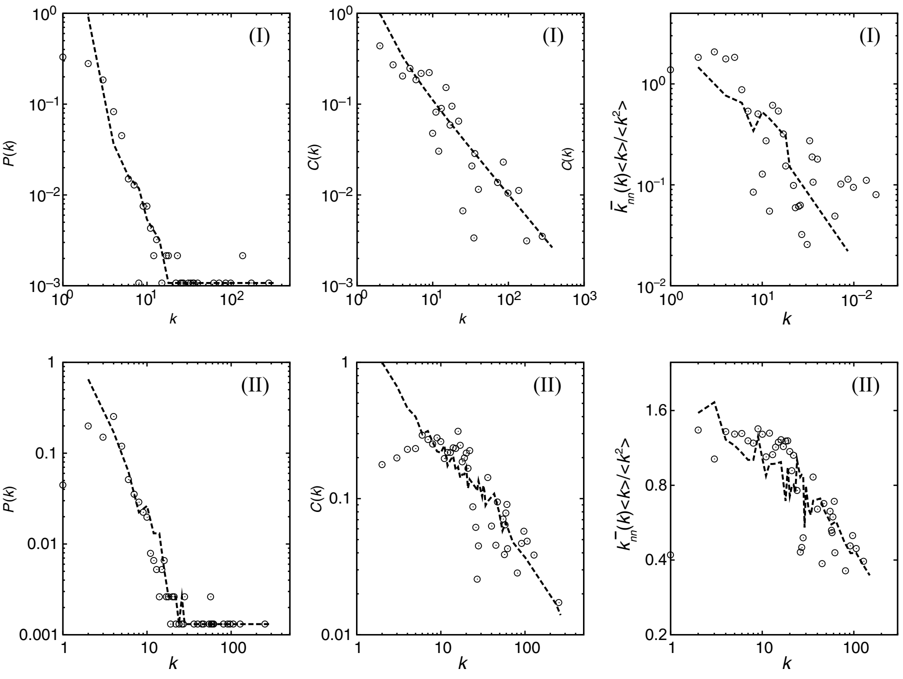


FIGURE 3.8 Comparison between structural properties of biological networks and those of model networks. The left column shows the degree distributions, $P(k)$. Degree-dependent clustering coefficients $C(k)$ are in the middle column. Degree–degree correlations $\bar{k}_{nn}(k)$ constitute the right column. These degree–degree correlations are divided by $\langle k^2 \rangle / \langle k \rangle$ for normalization. The symbols and dashed lines denote the data of biological networks and our model networks, respectively. (I) Gene regulatory network in *E. coli* [24] and our model with $a = 3$, $m = 1$, and $\xi = 20$. (II) Metabolic networks in *E. coli* [25] and our model with $a = 3$, $m = 2$, and $\xi = 1$. The size of the model network is the same as the total number of nodes of the biological network. The parameters a, m, and ξ are determined by minimizing the distributional distance, which corresponds to the Kolmogorov–Smirnov statistics (distances) for degree distributions between the predicted distributions and the empirical distributions.

In order to avoid this problem, we need to eliminate parameter tunings and construct models in which parameters are determined from observable statistics obtained from real-network data, such as the number of nodes.

In this chapter, we focus on metabolic networks and introduce a network model without parameter tuning.

3.3.1 Network Model

Here, we consider metabolic networks in which nodes and edges represent metabolites and metabolic reactions (substrate–product relationships based on atomic tracing [26]) and propose a simple model that reproduces the structural properties of metabolic

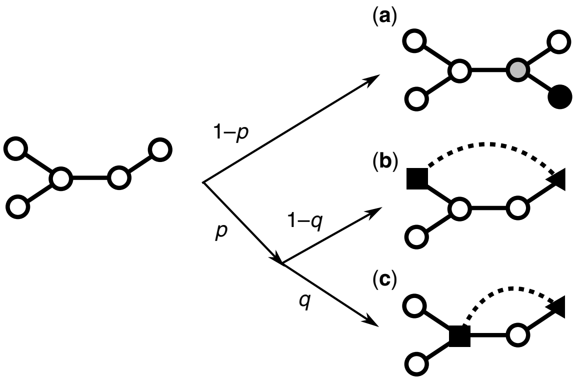


FIGURE 3.9 Schematic diagram of the growth mechanisms of the model. (a) Event I. The black and gray nodes represent a new node and a randomly selected existing node, respectively. (b) and (c) Event II. The dashed lines correspond to new edges. The triangular nodes are randomly selected existing nodes. The quadrangular nodes are existing nodes, selected by a random walk from each red node. The new edge becomes a short-cut path between the triangular node and the quadrangular node.

networks with two parameters, p and q. These parameters are determined from the statistics of real data (see Section 3.3.3 for details).

In general, it is believed that metabolic networks evolve by a reaction (enzyme) gain due to evolutionary events (see Ref. 27 for details). In this case, we can consider two situations: the case that a new reaction occurs between a new metabolite and an existing metabolite (Event I) and the case that a new reaction occurs between existing metabolites (Event II).

With the above consideration, the modeling is as follows.

(i) Event I occurs with the probability $1 - p$ (see Fig. 3.9a). In this case, a new node (the black node) is connected to a randomly selected existing node (the gray node).

(ii) Event II occurs with the probability p (see Fig. 3.9b and c). In this case, a short-cut path bypasses the path between a node and another node. The short-cut path is generated via a random walk because it may be drawn based on the existing network structure. For example, such short-cut chemical reactions may occur between related metabolic compounds, which are nearly located on the metabolic pathway (see Ref. 27 for details).Hence, we need to consider the length of the bypassed path. However, when we investigate the degree distribution and the degree-dependent clustering coefficient, it is sufficient to consider only two cases: (1) the length is equal to 2 and (2) the length is greater than or equal to 3. This assumption (of considering only two cases) is valid because the degree distribution is independent of the bypassed path length and the clustering coefficient is influenced only when a path of length 2 is bypassed (see Section 3.3.2 for the details). Therefore, we express the bypassed path length using the parameter q as follows. First, an initial node (the triangular nodes in Fig. 3.9b and c) is selected at random. Next, with the probability q, we select a path of length 2 based on a random walk from the initial node. In contrast, with the probability $1 - q$, we select a path with length greater than or equal to 3 based on a random walk from the initial node. Thus, the parameter q roughly corresponds to the bypassed path length, which is the path length between nodes connected through a short-cut path. Finally, a new edge (short-cut path) is drawn between the initial node (the triangular nodes in Fig. 3.9b and c) and the terminal node (the quadrangular nodes in Fig. 3.9b and c). Note that a triangle is accordingly generated with the probability pq.

3.3.2 Analytical Solution

Degree Distribution. First, we show the analytical solution for degree distribution of the model via mean-field-based analysis [12,22].

We consider the time evolution of k_i, which is the degree of node i. When Event I occurs, the degree of node i increases by one with the probability $1/N$, where N is the total number of nodes. Further, when Event II occurs, two existing nodes are selected and their respective degrees increase. The degree of one node increases by one with the probability $1/N$, because this node is randomly selected. The degree of another node increases by one with the probability $k_i/\sum_j k_j$, because this node is selected by a random walk from a randomly selected node. It is reported that the probability that a walker arrives at this node equals $k_i/\sum_j k_j$, irrespective of the number of steps in the random walk [28]. Note that this probability equals to that of the preferential attachment. Thus, the time evolution of k_i is

$$\frac{\mathrm{d}}{\mathrm{d}t}k_i = (1-p)\frac{1}{N} + p\left[\frac{1}{N} + \frac{k_i}{\sum_j k_j}\right], \tag{3.17}$$

where $N = (1-p)t$, because the number of nodes increases by one with the probability $1 - p$, and $\sum_j k_j = 2t$ because one edge is added each time. It should be noted that this equation is independent of the bypassed path length (the parameter q). The solution of the above equation with the initial condition $k_i(t = s) = 1$ is

$$k_i = [A(p) + 1]\left(\frac{t}{s}\right)^{p/2} - A(p), \tag{3.18}$$

where $A(p) = 2/[p(1-p)]$.

From the above equation, because $s/t = P(\geq k)$ as shown in the previous section, the cumulative distribution $P(\geq k)$ is

$$P(\geq k) = [A(p) + 1]^{2/p}[k + A(p)]^{-2/p}. \tag{3.19}$$

Since $P(k) = -\frac{\mathrm{d}}{\mathrm{d}k}P(\geq k)$, we finally obtain the degree distribution

$$P(k) = (\gamma - 1)[A(p) + 1]^{\gamma-1}[k + A(p)]^{-\gamma}, \tag{3.20}$$

where the degree exponent γ is

$$\gamma = \frac{2}{p} + 1. \tag{3.21}$$

As shown in Equation 3.20, the degree distribution follows a power law with a cutoff within a small degree.

Degree-Dependent Clustering Coefficient. Next, we show an analytical solution for the degree-dependent clustering coefficient of the model via mean-field-based analysis.

The clustering coefficient of node i is defined as $C_i = 2M_i/[k_i(k_i - 1])$, where M_i is the number of edges among the neighbors of node i. We consider the time evolution of M_i. The number of edges of M_i increases with the probability pq, because M_i increases when Event II occurs and a path of length 2 is bypassed (a triangle is generated). In other words, we do not need to consider a bypassed path with length greater than or equal to 3. Next, the M_i of each node, which belongs to the triangle, approximately increases by one. The M_i of one node increases by one with the probability $1/N$, because this node is selected at random. The M_is of the other two nodes increase by one with the probability $k_i/\sum_j k_j$, because these nodes are selected by a random walk. Therefore, the time evolution of M_i is

$$\frac{\mathrm{d}}{\mathrm{d}t}M_i \simeq pq\left[\frac{1}{N} + 2\frac{k_i}{\sum_j k_j}\right], \tag{3.22}$$

where $N = (1 - p)t$ and $\sum_j k_j = 2t$. Moreover, $k_i = [A(p) + 1](t/s)^{p/2} - A(p)$, as shown in Equation 3.18.

The solution of the above equation with the initial condition $M_i(t = s) = 0$ is

$$M_i = 2q[A(p) + 1]\left(\frac{t}{s}\right)^{p/2} + \frac{q(p-2)}{1-p}\ln\frac{t}{s} - 2q[A(p) + 1], \tag{3.23}$$

where $A(p) = 2/[p(1 - p)]$. From Equation 3.18, since $k_i = [A(p) + 1](t/s)^{p/2} - A(p)$, this equation is rewritten as

$$M_i = 2q\left[(k_i - 1) - \frac{1}{2}(2 - p)A(p)\ln\frac{k_i + A(p)}{1 + A(p)}\right]. \tag{3.24}$$

Substituting this equation into the definition of clustering coefficient described above, we finally obtain the degree-dependent clustering coefficient

$$C(k) = q\left[\frac{4}{k} - \frac{2(2-p)A(p)}{k(k-1)}\ln\frac{k + A(p)}{1 + A(p)}\right]. \tag{3.25}$$

Average Clustering Coefficient. Finally, we show an analytical solution for the average clustering coefficient of the model.

The average clustering coefficient is expressed as the summation of the product of the degree distribution and the degree-dependent clustering coefficient: $C = \sum_{k=2}^{K_m} P(k) \times C(k)$, where K_m is the maximum degree. We approximate this summation by the integral equation:

$$C = \int_2^{K_m} P(k) \times C(k)\mathrm{d}k, \tag{3.26}$$

The maximum degree indicates that the cumulative probability equals to $1/N$; thus, $P(\geq K_m) = 1/N$. From Equation 3.19, K_m can be expressed as

$$K_m = N^{p/2}[A(p) + 1] - A(p). \tag{3.27}$$

Equation 3.26 is solved via numerical integration because it is analytically unsolvable.

3.3.3 Estimation of the Parameters

As explained in Section 3.3.1, this model has two parameters p and q. In order to reproduce the structural properties of metabolic networks, we need to estimate these parameters in real-world networks. In this section, we discuss the approach for estimating these parameters.

Parameter p. Here, we consider the time evolutions of the number of nodes N and the number of edges E.

By the definition of this model, these time evolutions are described as $N = (1 - p)t$ and $E = t$. Therefore, the parameter p is estimated as

$$p = 1 - \frac{N}{E}, \tag{3.28}$$

where N and E are obtained from real metabolic networks.

Parameter q. Here, we consider the number of triangles T of this model.

In this model, when Event II occurs, the number of triangles approximately increases by one with the probability pq, because a triangle is generated with the probability q. That is,

$$T \simeq pqt. \tag{3.29}$$

Since $N = (1 - p)t$, this equation is rewritten as

$$T \simeq pq\frac{N}{1 - p}. \tag{3.30}$$

From this equation, therefore, the parameter q is estimated by

$$q \simeq \frac{T}{E - N}, \tag{3.31}$$

where T, N, and E are obtained from real metabolic networks.

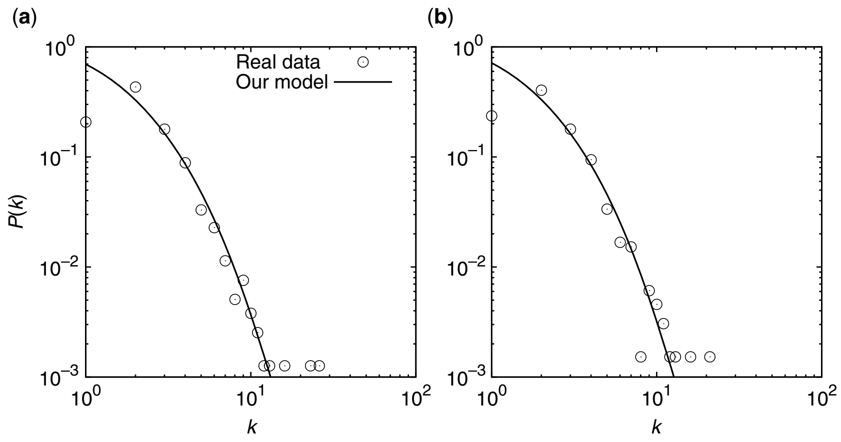


FIGURE 3.10 Comparison between the degree distribution of our model and that of real metabolic networks of *E. coli* (a) and *Bacillus subtilis* (b). The symbols represent the data for real networks. The lines are obtained from Equation 3.20.

3.3.4 Comparison with Real Data

Here, we compare structural properties of this model with those of real metabolic networks.

First, we obtain the parameters p and q from the metabolic network of each organism using Equations 3.28 and 3.31, respectively. Substituting the parameters into the equations, shown in Section 3.3.2, we obtain the structural properties predicted from this model.

Figure 3.10 shows a comparison between the degree distribution of our model and that of real metabolic networks. We provide the $P(k)$ values of two well-known organisms.

Figure 3.11 shows a comparison between the degree exponent of our model and that of real metabolic networks. For comparison, the degree exponents are obtained by the maximum likelihood method, considering a cutoff, represented by $A(p)$, as follows:

$$\gamma = 1 + N\left[\sum_{i=1}^{N} \ln \frac{k_i + A(p)}{k_{min} + A(p)}\right]^{-1}. \tag{3.32}$$

This value is different from that obtained by the original maximum likelihood method [29].

Figure 3.12 shows a comparison between the degree-dependent clustering coefficient of our model and that of real metabolic networks. We show $C(k)$ for the same two organisms.

Figure 3.13 shows a comparison between the average clustering coefficient of our model and that of real metabolic networks. The predicted average clustering coefficients are obtained with Equation 3.26 via numerical integration.

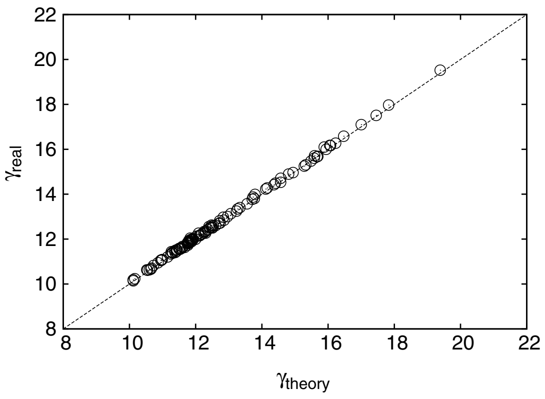


FIGURE 3.11 Comparison between the degree exponent γ of our model (theoretical) and that of real metabolic networks (real). The dashed line represents $\gamma_{\text{real}} = \gamma_{\text{theory}}$.

As shown in Figures 3.10–3.13, the theoretical predictions are in good agreement with the real data, indicating that this model can reproduce the structural properties of real metabolic networks.

As mentioned above, this model has a few parameters, which can be estimated very easily. Furthermore, the model parameters relate to the frequency of evolutionary events such as horizontal gene transfers and gene duplications, through which the metabolic networks expand (e.g., see Refs. [30,31]). This model may be useful for the estimation of missing pathways and the evolutionary origin of metabolic reactions.

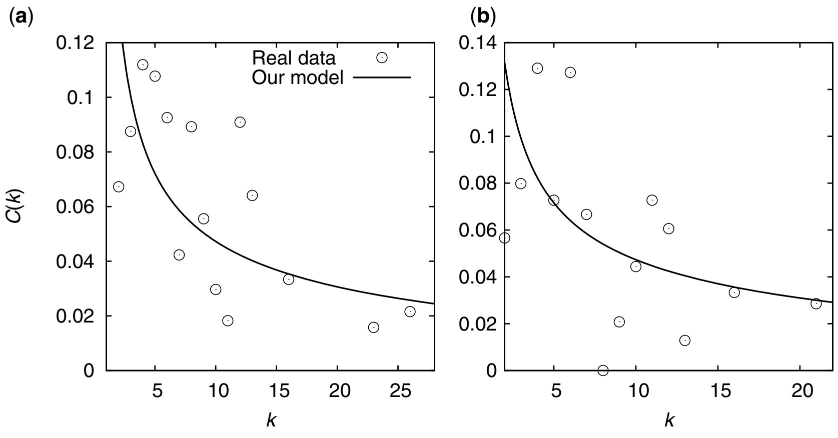


FIGURE 3.12 Comparison of the degree-dependent clustering coefficient between our model and the real metabolic networks of (a) *E. coli* and (b) *B. subtilis*. The symbols indicate the data for real networks. The lines are obtained from Equation 3.25.

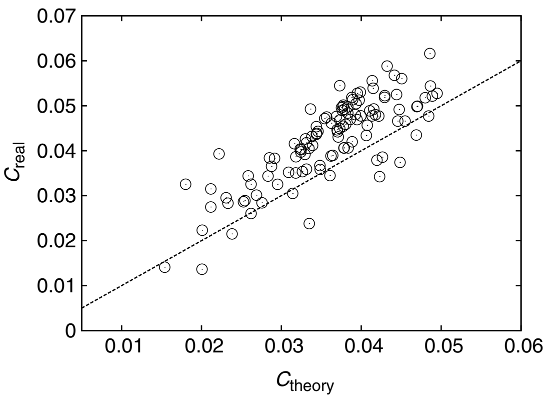


FIGURE 3.13 Comparison between the average clustering coefficient C of our model (theoretical) and that of real metabolic networks (real). The dashed line shows $C_{\text{real}} = C_{\text{theory}}$.

3.4 BIPARTITE RELATIONSHIP: A CASE STUDY OF METABOLITE DISTRIBUTION

In the previous sections, we have focused on simple (unipartite) networks consisting of one type of nodes and edges between its nodes. For instance, in the metabolic networks mentioned in the above section, the nodes are metabolites and the edges are drawn between these nodes (i.e., between the metabolites). However, we sometimes observe biological systems represented as bipartite networks.

An interesting example of this is metabolite distributions (or species–metabolite networks), which indicate how metabolites are distributed among species (i.e., the relationship between species and metabolites). Living organisms produce various types of compounds via their metabolisms, which are believed to adaptively shapeshift with changing environment over a long evolutionary history. Thus, metabolite distributions are important in order to elucidate the design principles of metabolisms such as adaptive mechanisms.

The relationship between two types of objects is represented as a bipartite network. Bipartite networks are defined as graphs having two different node sets (in this case, species and metabolites), in which edges are only drawn between one node set and the other node set (interconnectivity). Notably, there is no edge between the nodes belonging to the same node set (intraconnectivity). In the species–metabolite networks, an edge is drawn between a species node and a metabolite node when the species has the metabolite.

Another example of bipartite relationships is the relationship between proteins and the drugs targeting the proteins (i.e., drug–target networks) [32]. Further examples in other fields are the mutualistic relationship between plants and pollinators in ecological systems and manufacturer–contractor interactions in social systems [33]. As mentioned above, bipartite relationships are concepts utilized in a wide range of fields and are important to understand several systems.

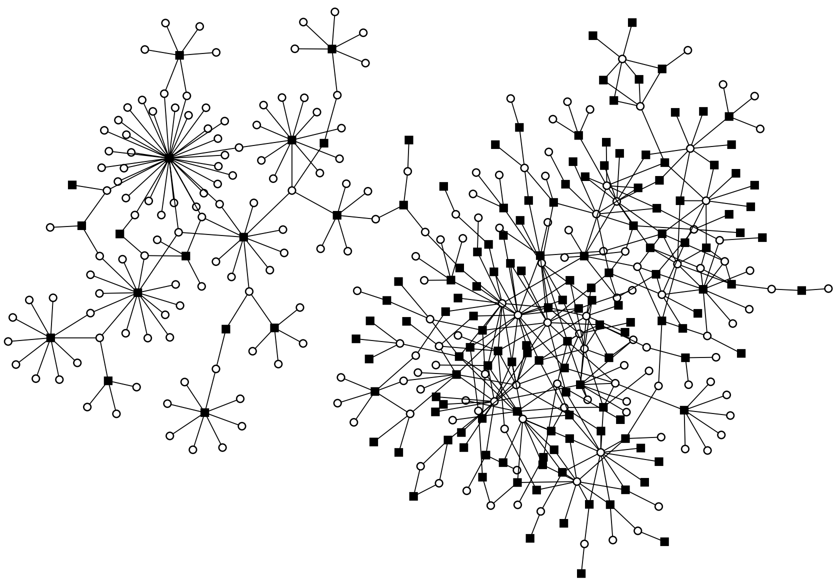


FIGURE 3.14 A partial species–flavonoid network for Lamiaceae (the Japanese basil family) drawn by the yEd Graph Editor [8]. The black squares and white circles correspond to plant species and flavonoids, respectively.

In this section, by focusing on the example of metabolite distributions, we present several structural properties observed in real bipartite relationships and introduce a simple bipartite network model.

3.4.1 Structural Properties of Metabolite Distributions

Metabolite distribution refers to the distribution of metabolites among species (i.e., species–metabolite relationship). There are several types of metabolites. Here, we focus on flavonoids, a type of metabolites. Flavonoids are particularly interesting examples when considering metabolite distributions among species. Previous studies [34–37] mention the importance and details of flavonoids.

We illustrate the example of a partial species–flavonoid network for Lamiaceae (the Japanese basil family) in Figure 3.14. The node degrees of species nodes (black squares) and flavonoid nodes (white circles) are found to be extremely varied.

Heterogeneous Connectivity. In order to characterize the tendency of connectivity, we investigated the frequency distributions of the node degree (degree distribution) in species–flavonoid networks. In bipartite networks, we can find two types of degree distributions due to the two types of nodes (i.e., species nodes and flavonoid nodes); thus, there are two types of node degrees. The node degree for a species node n_f corresponds to the number of flavonoids in its species.

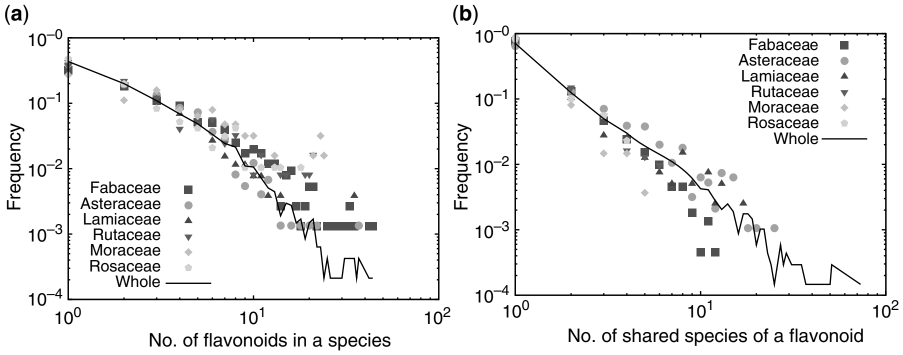


FIGURE 3.15 Degree distributions of the species–flavonoid networks. (a) Frequency distribution of the number of flavonoids in a plant species. (b) Frequency distribution of the number of shared species of a flavonoid. The symbols indicate the distributions for family-based species–flavonoid networks. The solid lines represent the distributions for all-encompassing species–flavonoid networks.

On the other hand, the node degree for a flavonoid node n_s denotes the number of species possessing its flavonoid.

As shown in Figure 3.15, the frequency distributions of n_f and n_s roughly follow the power law, implying the heterogeneous distribution of flavonoids among species. According to this power law, most flavonoids are shared by a few species; however, a few flavonoids are conserved in many species. Similarly, most species have flavonoids of a few types; however, a few species have flavonoids of many types. Furthermore, the heterogeneous distributions of flavonoids among species characterized by the power–law statistics are approximately conserved between family-based species–flavonoid networks, suggesting a scale-free feature. Regarding the number of metabolites in a species following power–law distributions, a similar result has been additionally reported in Ref. [38]. The biological meaning of heterogeneous connectivity in metabolite distributions is discussed in Ref. [36].

Nested Structure. We here consider the two types of nodes (A and B) in a bipartite network. In this case, the nested structure indicates that the set of B nodes connected to a given A node is a subset of B nodes connected to other A nodes (see Fig. 3.16). In other words, this structure in metabolite distributions implies that a plant's metabolite composition is a subset of other plants' metabolite compositions. This structural property is often considered in ecological networks such as plant–pollinator mutualism [39], because such nonrandom patterns often strongly control the dynamics of ecological systems [40].

Modular Structure. The modular structure represents that the subsets of A nodes (modules), in which nodes are strongly interconnected through B nodes, are weakly connected to each other [41] (see Fig. 3.16). In other words, species are divided into several clusters based on their metabolite compositions. This

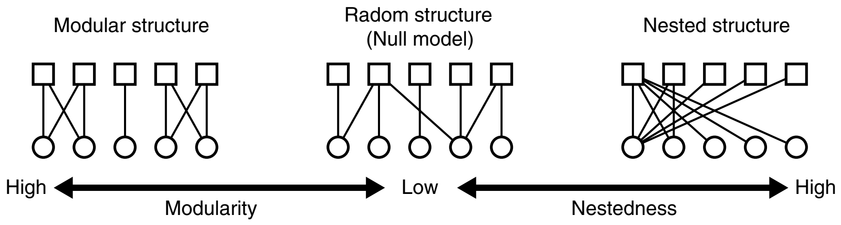


FIGURE 3.16 Schematic diagram of nested structure and modular structure in bipartite networks. This figure is based on Ref. [42].

structural property helps in understanding the coevolution of two objects. This structure is investigated in a wide range of fields such as those pertaining to social, ecological, and biological systems and are useful for clustering and functional predictions.

We investigate the nestedness and modularity of metabolite (flavonoid) distributions. To measure the degrees of nested structure and modular structure (i.e., nestedness N and modularity Q) of metabolite distributions, we employed the BINMATNEST program [43] and the optimization algorithm proposed in Ref. [41], respectively. The nestedness N ranges from perfect non-nestedness ($N = 0$) to perfect nestedness ($N = 1$), and high modularity Q indicates a strong modular structure. We also calculated N and Q from randomized networks generated by the null model 2 in Ref. [39] in order to demonstrate the statistical significance of the structural properties. The statistical significance is suitably evaluated because the null model 2 generates randomized networks without the bias of heterogeneous connectivity.

Figure 3.17 shows the comparison between the nestedness N and modularity Q of real data and those of the null model data for each family. As shown in

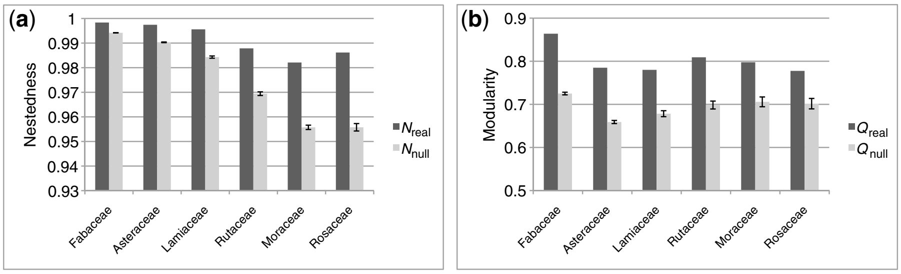


FIGURE 3.17 (a) Significantly high nestedness N and (b) modularity Q in metabolite distributions across plant species. The dark gray bars and the light gray bars correspond to real values and null model values, respectively. N and Q obtained from the null model are averaged over 100 realizations. All P-values for the difference are lower than 0.0001. The P-value is derived using the Z-score defined as $(x_{\text{real}} - \bar{x}_{\text{null}})/SE_{\text{null}}$, where x_{real} corresponds to real values (nestedness or modularity). $\bar{x}_{\text{null}}$ and SE_{null} are the average value for the null model and its standard error, respectively.

Figure 3.17, N and Q of real data are significantly larger than that of the null model, indicating that metabolite distributions also show nested structure and modular structure in addition to heterogeneous connectivity, similar to ecological networks and organizational networks. In addition, nestedness and modularity are different structural properties as there is no correlation between them (Pearson correlation coefficient, $r = 0.345$ with P-value, $p = 0.503$).

3.4.2 Bipartite Network Model

In the previous section, we showed the various structural properties of metabolite distributions. Here, we speculate on the possible origins of these structural properties using a simple model.

We consider two simple evolutionary mechanisms as follows.

(i) New flavonoids are generated by the variation of existing flavonoids. In evolutionary history, species accordingly obtain new metabolic enzymes via evolutionary events, and the metabolic enzymes synthesize new flavonoids through the modification of existing flavonoids with substituent groups and functional groups.

(ii) The flavonoid compositions of new species are inherited from those of existing (ancestral) species. New species are believed to emerge by mutation of ancestral species and are thus similar to the ancestral species. Hence, flavonoid compositions of the ancestral species might be inherited by new species. In addition, independent of our model, a bipartite network model based on the above inheritance (or copy) mechanism was proposed in Ref. [44] around the same time, with the aim of describing the evolution of protein domain networks. Note that the linking mechanism in the procedure (ii) is asymmetric with that in the procedure (i).

Considering the above two mechanisms, we propose a simple model with two parameters, p and q, reproducing the heterogeneous distributions of flavonoids among species.

Our model is defined by the following procedure:

(a) We set an initial species–flavonoid network represented as a complete bipartite graph with n_0 species and n_0 flavonoids (Fig. 3.18a).

(b) Event I, which corresponds to the emergence of a new species, occurs with probability p. An existing species is selected at random (Fig. 3.18b). A new species emerges due to the mutation of randomly selected existing species, and the flavonoids of the existing species are inherited by the new species as their candidate flavonoids (Fig. 3.2c). Due to the divergence of the flavonoid compositions, the new species finally acquires flavonoids with equal probability q for each of the candidates (Fig. 3.18d). However, if the new species has no flavonoids then it is neglected (removed) in accordance with the observation

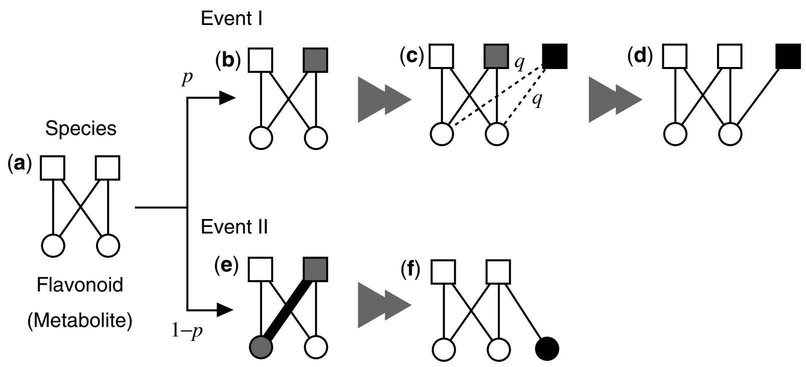


FIGURE 3.18 Schematic diagram of the model. Squares and circles represent the plant species and flavonoids, respectively. (a) An initial species–flavonoid relationship (network) with $n_0 = 2$. (b–d) Event I: the emergence of a new species. The gray square represents a randomly selected existing species. The black square represents a new species emerging due to the duplication of existing species. The dashed lines indicate possible pairs of the new species and flavonoids. (e, f) Event II: the emergence of a new flavonoid. The thick edge between gray nodes corresponds to a randomly selected existing species–flavonoid pair. The black circle represents a new flavonoid.

condition (species without flavonoids are not included in our dataset). In contrast to Event I, Event II corresponding to the emergence of a new flavonoid occurs with probability $1 - p$. A species–flavonoid pair is uniformly selected at random (Fig. 3.18e). Then, the species receives a new flavonoid (Fig. 3.18f).

(c) The procedure (b) is repeated until the number of species and the number of flavonoids are equivalent to S and M, respectively.

Our model has two parameters, p and q. The parameter p is estimated as

$$p = \frac{S}{S + M} \tag{3.33}$$

because p only controls the number of species and the number of flavonoids. The estimation of q is described in the following section. This model does not require parameter tunings.

Relation with "Rich-get-Richer" Mechanisms. The emergence of heterogeneous (power–law) distributions in evolving systems might be caused by the "rich-get-richer" or preferential mechanisms, according to which, the increase of a statistic is proportional to the statistic itself [1,12]. Here, we discuss the "rich-get-richer" mechanism of our model.

We first mention the number of flavonoids in a species n_f. When Event II occurs, n_f increases. The number of flavonoids of species i, n^i_f, increases when a randomly selected species–flavonoid pair includes species i. Thus, species with many flavonoids tend to be selected in such a case. As a result, such species acquire more flavonoids, resulting in a "rich-get-richer" mechanism. The origin of this preferential mechanism is similar to that of the mechanism in the Dorogovtsev–Mendes–Samukhin (DMS) model [45]. However, our model is essentially different from the DMS model because the DMS model does not describe bipartite relationships.

This "rich-get-richer" mechanism is mathematically described as follows. We consider the time evolution of n^i_f. Let $L(t)$ be the total number of species–flavonoid pairs at time t; the probability that species i with n^i_f flavonoids is chosen is equivalent to $n^i_f/L(t)$ because the pair is randomly selected. In addition, Event II occurs with the probability $1 - p$. Therefore, the time evolution of n^i_f is described as

$$\frac{\mathrm{d}}{\mathrm{d}t}n^i_f = (1-p)\frac{n^i_f}{L(t)}. \tag{3.34}$$

Next, we focus on the time evolution of $L(t)$. The number of pairs $L(t)$ increases in Events I and II. In the case of Event I, $L(t)$ increases by $q \times L(t)/S(t)$, where $S(t)$ is the number of species at time t, because the flavonoids of the randomly selected existing species are inherited by the new species with the probability q. It should be noted that the expected number of flavonoids of randomly selected species is $\sum_{j=1}^{S(t)} n^j_f/S(t) = L(t)/S(t)$. In the case of Event II, $L(t)$ increases by 1. The Events I and II occur with the probabilities p and $1 - p$, respectively. Therefore, the time evolution of $L(t)$ is expressed as

$$\frac{\mathrm{d}}{\mathrm{d}t}L(t) = pq\frac{L(t)}{S(t)} + (1-p). \tag{3.35}$$

Since $S(t) = pt$, the solution of this equation for the initial condition $L(1) = L_0$ is

$$L(t) = \frac{1-p}{1-q}t + \left(L_0 - \frac{1-p}{1-q}\right)t^q. \tag{3.36}$$

Assuming a small L_0 and $t \gg 1$, since $q < 1$ is satisfied, the above equation is approximated as

$$L(t) \approx \frac{1-p}{1-q}t. \tag{3.37}$$

This equation indicates that $L(t)$ is approximately proportional to time t for a large t and a relatively small q.

Using the above equation, we can estimate the parameter q as follows:

$$q = 1 - \frac{M}{L} \tag{3.38}$$

Substituting Equation 3.37 into Equation 3.34, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}n^i_f \approx (1-q)\frac{n^i_f}{t}, \tag{3.39}$$

which suggests preferential mechanism, that is, the increase of n^i_f is proportional to n^i_f.

From this equation, using the mean-field-based method [12], we immediately obtain the power–law distribution of n_f:

$$P(n_f) \sim n_f^{-(2-q)/(1-q)}. \tag{3.40}$$

We next consider the number of shared species of flavonoid n_s. When Event I occurs, n_s increases. The number of species shared by a flavonoid i, n^i_s, might increase when a randomly selected species has the flavonoid i. Thus, flavonoids shared by many species tend to be selected in such a case. As a result, such flavonoids are shared by more species, reflecting a "rich-get-richer" mechanism. The origin of this preferential mechanism is analogous to that of the mechanism in the duplication-divergence (DD) model [46]. However, our model is also different from the DD model in that the DD model does not describe bipartite relationships.

This "rich-get-richer" mechanism is mathematically explained as follows. We consider the time evolution of n^i_s. The probability that each flavonoid i is shared by a new species is equivalent because the resulting new species obtains the flavonoid i with the probability q when one of the n^i_s species with flavonoid i is randomly selected. In addition, Event I occurs with the probability p. Since $S(t) = pt$, the time evolution of n^i_s is described as

$$\frac{\mathrm{d}}{\mathrm{d}t} n^i_s = pq \frac{n^i_s}{S(t)} = q \frac{n^i_s}{t}, \tag{3.41}$$

which indicates the preferential mechanism, that is, the increase of n^i_s is proportional to n^i_s.

As in the case of n_f, we immediately obtain the power–law distribution of n_s:

$$P(n_s) \sim n_s^{-(1+q)/q}. \tag{3.42}$$

Origin of Nested and Modular Structures. We could not derive the analytical solutions of the nestedness and modularity in our model because of the complicated definitions of these structural properties. Using our model, however, the formation mechanisms of the structural properties in metabolite distributions are described as follows.

The nested structure implies that a plant's flavonoid composition is a subset of the flavonoid compositions of other plants; its origin is explained using our model as follows. In our model, metabolites of a new plant are inherited from those of an ancestral plant because the new plant tends to be similar to the ancestral plant with mutation. However, new plants obtain the metabolite part due to divergence (elimination of interactions). As a result, the metabolites of an offspring plant become a subset of those of their parent plant and hence result in a nested structure.

The modular structure implies that plant species are divided into several clusters in which they are strongly interconnected through common metabolites; these clusters interact loosely. In simple terms, a modular structure is obtained by the strong interconnections in the clusters and weak interactions among the clusters. The emergence of weak and strong interactions is also described by the inheritance and divergence of metabolite compositions. As mentioned above, metabolite compositions of the new species are inherited from ancestral species in our model. Hence, new species and ancestral species are connected because of common metabolites; the interactions of this type correspond to strong interconnections. Due to divergence, on the other hand, new species indirectly connect to other species via the metabolites of ancestral species that were not inherited by the new species; this results in weak interactions.

3.4.3 Comparison with Real Data

We estimated the parameters p and q from real data using Equations 3.33 and 3.38 and generated the corresponding model networks for comparison with real ones.

Degree Distribution. We compare the frequency distributions of the number of flavonoids of a plant species n_f and the number of species sharing a flavonoid n_s between our model and the real networks.

Figure 3.19 shows the degree distributions of metabolite distributions (species–metabolite networks) (symbols) and the models (lines). Due to space limitations, the degree distributions of only three metabolite distributions are used as representative examples. We could observe the degree distributions of

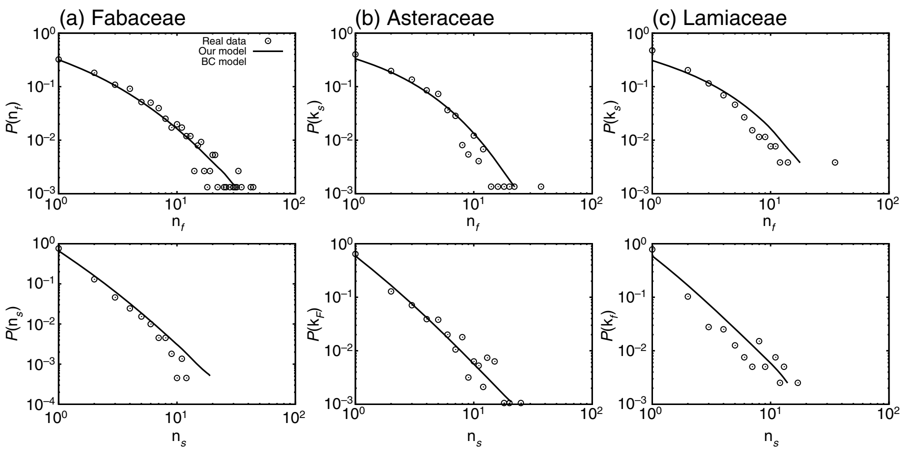


FIGURE 3.19 Heterogeneous degree distributions in metabolite distributions for plant species (top column) and metabolites (flavonoids) (bottom column). The circles and lines represent real data and models, respectively. The degree distributions of models are averaged over 100 realizations.

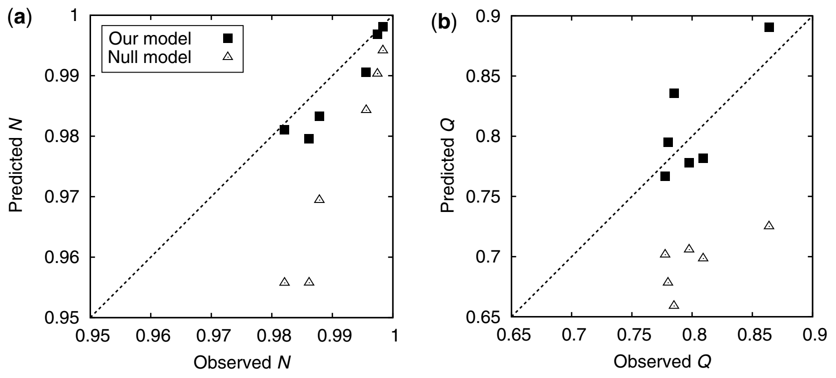


FIGURE 3.20 Comparison between the (a) nestedness N (b) and modularity Q of our model and those of real networks. The dashed line represents the perfect agreement between predicted values (N or Q) and observed ones. The nestedness and the modularity obtained from the models are averaged over 100 realizations.

two types [$P(n_f)$ and $P(n_s)$, where n_f and n_s denote the degrees of nodes corresponding to plant species and metabolites (flavonoids), respectively] because metabolite distributions are represented as bipartite graphs. In both the cases, the degree distributions follow a power law with an exponential truncation and the model-generated degree distributions are in good agreement with real ones.

Nestedness and Modularity. Next, we discuss the prediction of nestedness N and modularity Q using our model. Figure 3.20 shows the comparison between the N and Q values of the model and those of the real data. For comparison, we also computed N and Q from the null model.

As shown in this figure, the prediction accuracy of our model is clearly higher than that of the null model and the data from our model is in good agreement with real data.

3.4.4 Related Model

The model proposed by Saavedra et al. [33] is a different type of bipartite network model, called a bipartite cooperation (BC) model. The BC model is a nongrowth model, in which the number of nodes is fixed. Thus, this model is not an evolutionary model and is different from our model. In the BC model, interactions between nodes are determined based on the traits (or properties) of nodes rather than evolutionary mechanisms.

The BC model reproduces the structure of bipartite relationships in ecological and social systems. However, with regard to metabolite distributions, the BC model shows a lower prediction accuracy than our model (see Ref. [37] for details). In addition, we partially observed the high prediction accuracy of our model rather than the BC model in the case of ecological networks although the BC model is believed to be a good model for ecological networks (see Ref. [47] for details).

However, our model (evolutionary model) does not contradict the BC model (non-growth network model) because these models describe different mechanisms for the formation of nonrandom structures and therefore provides additional insights into the formation of bipartite networks. For example, the modularity of real ecological networks is in good agreement with that of the BC model rather than that of our model [48]. These two different models play an important role in the deeper understanding of formation mechanisms of bipartite networks.

3.5 CONCLUSION

In this chapter, we introduced several evolving network models by focusing on biological networks. The model networks were compared with real data, and they were found to be in good agreement with the real data in terms of structural properties. In particular, the model mentioned in Section 3.2 reproduces several types of biological networks such as gene regulatory networks and metabolic networks. The models in Sections 3.3 and 3.4 do not require parameter tunings, which are necessary in most existing models. Thus, it is easy to estimate the frequency of evolutionary events such as gene duplication and divergences. Our models are expected to provide a platform for the elucidation of formation mechanisms in biological networks. For instance, metabolic networks are believed to shape-shift in response to environmental changes [49,50], and the model in Section 3.3 explains the possible origin of structural differences with respect to growth temperatures (an environmental factor) [27].

In addition to this, our models also serve as a foundation for the prediction of interactions between biomolecules, that is, "*link prediction*" [51]. It is believed that real networks have several missing links, and the finding of such links is an important challenge in various fields including biology. The statistical and machine-learning techniques are often utilized in link prediction. Network models are particularly useful because they estimate the probability of interactions between nodes.

We use the concepts of network models in biology as well as in society and ecology. As mentioned in Section 3.4, our bipartite network model can also explain the structure of ecological networks in addition to metabolite distributions. The models mentioned in this chapter may be applicable in a wide range of fields.

REFERENCES

1. A.-L. Barabási, Z.N. Oltvai, Network biology: understanding the cell's functional organization, *Nat. Rev. Genet.* **5**, 101 (2004).
2. R. Albert, Scale-free networks in cell biology, *J. Cell Sci.* **118**, 4947 (2005).
3. U. Alon, *An Introduction to Systems Biology: Design Principles of Biological Circuits*, Chapman & Hall/CRC, Florida, 2006.
4. S.N. Dorogovtsev, J.F.F. Mendes, *Evolution of Networks: From Biological Nets to the Internet and WWW*, Oxford University Press, Oxford, 2003.
5. Z.N. Oltvai, A.-L. Barabási, Life's complexity pyramid, *Science* **298**, 763 (2002).

6. B.Ø. Palsson, *Systems Biology: Properties of Reconstructed Networks*, Cambridge University Press, New York, 2006.
7. 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**, D354 (2006).
8. http://www.yworks.com/products/yed/
9. L. Salwinski, C.S. Miller, A.J. Smith, F.K. Pettit, J.U. Bowie, D. Eisenberg, The Database of Interacting Proteins: 2004 update, *Nucleic Acids Res.* **32**, D449 (2004).
10. R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, *Rev. Mod. Phys.* **74**, 47 (2002).
11. B. Bollobas, *Random Graphs*, Academic Press, London, 1985.
12. A.-L. Barabási, R. Albert, H. Jeong, Mean-field theory for scale-free random networks, *Physica A* **272**, 173 (1999).
13. E. Ravasz, A.L. Somera, D.A. Mongru, Z.N. Oltvai, A.-L. Barabási, Hierarchical organization of modularity in metabolic networks, *Science* **297**, 1551 (2002).
14. P. Holme, B.J. Kim, Growing scale-free networks with tunable clustering, *Phys. Rev. E* **65**, 026107 (2002).
15. K. Takemoto, C. Oosawa, Evolving networks by merging cliques, *Phys. Rev. E* **72**, 046116 (2005).
16. M.E.J. Newman Assortative mixing in networks, *Phys. Rev. Lett.* **89**, 208701 (2002).
17. V.M. Eguíluz, K. Klemm, Epidemic threshold in structured scale-free networks, *Phys. Rev. Lett.* **89**, 108701 (2002).
18. G. Bianconi, A.-L. Barabási, Competition and multiscaling in evolving networks, *Europhys. Lett.* **54**, 436 (2001).
19. S.H. Yook, H. Jeong, A.-L. Barabási, Y. Tu, Weighted evolving networks, *Phys. Rev. Lett.* **86**, 5835 (2001).
20. A. Barrat, M. Barthélemy, A. Vespignani, Weighted evolving networks: coupling topology and weight dynamics, *Phys. Rev. Lett.* **92**, 228701 (2004).
21. R. Pastor-Satorras, A. Vázquez, A. Vespignani, Dynamical and correlation properties of the internet, *Phys. Rev. Lett.* **87**, 258701 (2001).
22. A. Barrat, R. Pastor-Satorras, Rate equation approach for correlations in growing network models, *Phys. Rev. E* **71**, 036127 (2005).
23. K. Takemoto, C. Oosawa, Modeling for evolving biological networks with scale-free connectivity, hierarchical modularity, and disassortativity, *Math. Biosci.* **208**, 454 (2007).
24. H. Salgado, S. Gama-Castro, A. Martínez-Antonio, E. Díaz-Peredo, F. Sénchez-Solano, M. Peralta-Gil, D. Garcia-Alonso, V. Jiménez-Jacinto, A. Santos-Zavaleta, C. Bonavides-Martńez, J. Collado-Vides, RegulonDB (version 4.0): transcriptional regulation, operon organization and growth conditions in *Escherichia coli* K-12, *Nucleic Acids Res.* **32**, D303 (2004).
25. H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, A.-L. Barabási, The large-scale organization of metabolic networks, *Nature* **407**, 651 (2000).
26. M. Arita, The metabolic world of *Escherichia coli* is not small, *Proc. Natl. Acad. Sci. U.S.A.* **101**, 1543 (2004).

27. K. Takemoto, T. Akutsu, Origin of structural difference in metabolic networks with respect to temperature, *BMC Syst. Biol.* **2**, 82 (2008).
28. J. Saramäki, K. Kaski, Scale-free networks generated by random walkers, *Physica A* **341**, 80 (2004).
29. M.E.J. Newman, Power laws, Pareto distributions and Zipf's law. *Contemporary Phys.* **46**, 323 (2005).
30. B. Papp, B. Teusink, R.A. Notebaart, A critical view of metabolic network adaptations, *HFSP J.* **3**, 24 (2008).
31. R. Fani, M. Fondia, Origin and evolution of metabolic pathways, *Phys. Life Rev.* **6**, 23 (2009).
32. M.A. Yıldırım, K.-I. Goh, M.E Cusick, A.-L. Barabási, M, Vidal, Drug-target network, *Nat. Biotech.* **25**, 1119 (2007).
33. S. Saavedra, F. Reed-Tsochas, B. Uzzi, A simple model of bipartite cooperation for ecological and organizational networks, *Nature* **457**, 463 (2009).
34. J. Gershenzon, T.J. Mabry, Secondary metabolites and the higher classification of angiosperms, *Nord. J. Bot.* **3**, 5 (1983).
35. B.A. Bohm, *Occurrence and Distribution of Flavonoids*, Harwood Academic Publishers, Amsterdam, 1998.
36. K. Takemoto, M. Arita, Heterogeneous distribution of metabolites across plant species, *Physica A* **388**, 2771 (2009).
37. K. Takemoto, Global architecture of metabolite distributions across species and its formation mechanisms, *Biosystems* **100**, 8 (2010).
38. Y. Shinbo, Y. Nakamura, M. Altaf-Ul-Amin, H. Asahi, K. Kurokawa, K. Arita, K. Saito, D. Ohta, D. Shibata, S. Kanaya, *KNApSAcK: A Comprehensive Species–Metabolite Relationship Database*, Biotechnology in Agriculture and Forestry vol. 57, Springer-Verlag, Berlin, 2006.
39. J. Bascompte, P. Jordano, C.J. Melián, J.M. Olesen, The nested assembly of plant-animal mutualistic networks, *Proc. Natl. Acad. Sci. U.S.A.* **100**, 9383 (2003).
40. U. Bastolla, M.A. Fortuna, A. Pascual-García, A. Ferrera, B. Luque, J. Bascompte, The architecture of mutualistic networks minimizes competition and increases biodiversity, *Nature* **458**, 1018 (2009).
41. R. Guimerá, L.A.N. Amaral, Functional cartography of complex metabolic networks, *Nature* **433**, 895 (2005).
42. E. Thébault, C. Fontaine, Stability of ecological communities and the architecture of mutualistic and trophic networks, *Science* **329**, 853 (2010).
43. M.A. Rodríguez-Gironés, L. Santamaría, A new algorithm to calculate the nestedness temperature of presence-absence matrices, *J. Biogeogr.* **33**, 924 (2006).
44. J.C. Nacher, T. Ochiai, M. Hayashida, T. Akutsu, A bipartite graph based model of protein domain networks, *Complex Sci.* **4**, 525 (2009).
45. S.N. Dorogovtsev, J.F.F. Mendes, A.N. Samukhin, Size-dependent degree distribution of a scale-free growing network, *Phys. Rev. E* **63**, 062101 (2001).
46. A. Vázquez, Growing network with local rules: Preferential attachment, clustering hierarchy, and degree correlations, *Phys. Rev. E* **67**, 056104 (2003).
47. K. Takemoto, M. Arita, Nested structure acquired through simple evolutionary process, *J. Theor. Biol.* **264**, 782 (2010).

48. K. Takemoto, unpublished data.
49. K. Takemoto, J.C. Nacher, T. Akutsu, Correlation between structure and temperature in prokaryotic metabolic networks, *BMC Bioinformatics* **8**, 303 (2007).
50. M. Parter, N. Kashtan, U. Alon, Environmental variability and modularity of bacterial metabolic networks, *BMC Evol. Biol.* **7**, 169 (2007).
51. L. Lu, T. Zhou, Link prediction in complex networks: a survey, arXiv/1010.0725 (2010).

4

MODULARITY CONFIGURATIONS IN BIOLOGICAL NETWORKS WITH EMBEDDED DYNAMICS

Enrico Capobianco, Antonella Travaglione, and Elisabetta Marras

4.1 INTRODUCTION

4.1.1 Biological Networks and Computational Challenges

The huge interest in complex networks across many research areas has also found application in biological studies, where associations between genes, proteins, and metabolites deserve further investigation particularly due to the underlying regulative or interactive dynamics. Here the proposed work addresses protein interactome networks (PIN) [1] from an integrative dynamic perspective, and aims to establish a better definition of their modular configurations.

There are currently reasons of concern in relation to the computational analysis of PIN, and they mainly refer to three problems. First, there is a limited interactome coverage [2] that depends on the organism under study [3] and on the available data-generating methodologies (yeast two-hybrid, co-IP, text mining and literature mining, DB curation, orthology, etc.). Consequently, data integration is often needed to ensure a better data uncertainty control and validation quality.

Second, there is also limited measurement accuracy as a limiting factor, and refers to the uncertainty inherent to both experimentally measured and predicted interactions (due to various sources of errors, biases, etc.). For example, evidence was recently provided [4] with regard to literature-curated interactome data about the necessity

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

of careful quality control for reliable inference. Notably, scoring systems have been proposed to assign reliability to the interactions, thus leading to common classification into low- and high-confidence PIN.

Third, detecting modularity is a very complicated task that offers only approximate solutions. Various different principles and methods (see Ref. 5 for comparative evaluations) can be applied for network partitioning, but without guarantee of achieving the best possible approximation quality due to the so-called "network resolution limit" problem [6,7]. As the truly informative module sizes may not match the algorithmically retrievable ones, we can observe suboptimal configurations, either sparse (with a few dense modules) or highly redundant (with many small-to-intermediate overlapping modules) maps.

As a result, such incompleteness and inaccuracy of representation calls for both new inference methods and better use of the currently adopted ones. These are challenging tasks, further complicated by the fact that the available protein interaction map consists of a mix of real interactions and false positives, and correspondingly non-interactions and false negatives. Therefore, while such map represents static entities subject to limitations and constraints, they actually refer to underlying associations that dynamically change depending on experimental conditions, system perturbations, and so on. Accordingly, a better control of the degree of uncertainty embedded in the protein maps requires that differential network features might be considered together with a variety of modular structures.

In order to control the uncertainty level, data integration is adopted in many systems biology applications; for instance, in PIN applications the use of gene coexpression and pathway information sources can complement the observed interactions. Another common strategy involves the analysis of topological properties [8,9]. Further refinement of interactome data can rely on similarity (dissimilarity) measures to allow for comparative analysis of PIN, and for the assignment of confidence levels to each interaction depending on biological and computational aspects.

4.1.2 Outline

The structure of this paper is as follows. We describe in Section 4.2 our methodological approach in both general and particular aspects related to the PIN setting; then, we present our results in Section 4.3; and finally, concluding remarks with a discussion follows in Section 4.4.

4.2 METHODS

4.2.1 General Approach

Interactome filtering may involve a selection of interaction data in terms of biological processes; as a result, a specific stratified interactome analysis could be performed. The advantage of such data disaggregation that we call "PIN fragmentation" is that the complexity levels of usually aggregated data would be avoided in favor of a reduction of dimensionality. The loss of information inherent to the integrated data may be balanced by the fact that the specific information layers that are needed may be promptly used. Thus, signaling or cell cycle or other processes can be examined once they have been retrieved, and can always be compared or combined, if needed.

Recent works [10–12] has inspired our approach. These authors compare topological characteristics of protein and metabolic filtered PIN from *Escherichia coli* and *Saccharomyces cerevisiae* model organisms, while we look specifically at cell cycle PIN data for this work, and plan to extend the fragmentation analysis in parallel studies. We thus built a compilation of filtered PIN from the yeast reference datasets of Reguly et al. [13], and used as a benchmark the literature-curated interactome (*LIT-Int*) obtained from small-scale experiments. Based on the reference dataset (say, *rPIN*) we applied a PIN fragmentation to extract subinteractomes specialized by cell biological processes.

The focus on the cell cycle process has the following rationale. Starting from the available fragmented PIN featuring cell cycle characteristics, data integration determined a sequence of "affine" PIN in which dynamic features enter through gene expression profiling and thus contribute to further differentiate subinteractomes belonging to the same process. In particular, the yeast cell cycle study of de Lichtenberg et al. [14] represents a precious experimental data source of characterized mRNA transcript levels measured during their periodical variation by time-course expression peaks.

Notably, building fragmented PIN by mapping such gene coexpression signatures brings the advantage of exploring the cohesiveness of protein modules and assessing the modular maps designed relatively to both intramodular and intermodular structures. We thus perform a more precise analysis of the constituent factors underlying the modules, which we call "modularization drivers" associated with the method used to retrieve the modules (module resolution driver), the specific biological process involved (protein interaction driver), and the integrated gene expression dynamics (gene expression driver).

4.2.2 PIN Fragmentation

In order to generate data endowed with high specificity and functional dependence in biological terms, we turn to a methodology designed to extract *ad hoc* interactome parts. In particular, PIN fragmentation represents the method to provide us with a compilation of subinteractomes; each of them has special features related to the biological processes that are involved, and if we consider just one biological process at a time, for instance, then we can end up with a series of networks that can be comparatively evaluated in their topological structure and organization. Notably, depending on the biological process that is examined, particularly interesting relationships may be present between the fragmented interactome parts, as the example reported below shows.

Relatively to our main source or interactome dataset, *rPIN*, the application of the fragmentation method delivered the following subinteractome list in relation to the cell cycle process:

- rPIN is the *LIT-Int* reference PIN.
- cPIN is the cell cycle PIN obtained from annotation of cell cycle proteins from both MIPS [15] and SGD [16] database. We classify as “static” this subinteractome, as we just refer to protein interactions.
- cePIN is a constrained cell cycle PIN built from a two-step procedure. Initially, gene expression profiles were obtained from the experiments cited above and designed to detect during the cell cycle phases the so-called “expression peaks” (maximal expression levels achieved) in order to represent gene signatures. Then, mapping such values to *rPIN* yields the corresponding “peak-to-peak” associated proteins (i.e., only proteins with related gene expression peaks are considered).
- cePIN-1 is similar to the previous subinteractome but less constrained because the constituent interactions depends on a peak signature associated with just one interacting protein (the other interacting protein may represent any other cell cycle protein regardless of the expression level). Thus, the previous *cePIN* is nested by construction.
- cePIN-2 is another subinteractome more relaxed than *cePIN-1* since now the interacting proteins with peak signature can link to any other protein, also not involved in cell cycle ones. Thus, *cePIN-1* is included.

Each extracted subinteractome has a number of proteins, say “n,” and corresponding interactions “i” that amount respectively to:

1. *rPIN*: $n = 3289, i = 11333$;
2. *cPIN*: $n = 771, i = 2493$;
3. *cePIN*: $n = 190, i = 381$;
4. *cePIN-1*: $n = 444, i = 977$;
5. *cePIN-2*: $n = 1193, i = 2254$.

The rationale behind such construction is that there exists substantial evidence from the literature that gene coexpression and physically interacting proteins tend to be correlated. Thus, as colocalization and coexpression correlate at the transcriptional level, their integration is expected to increase the reliability of modules detected by computational algorithms and further corroborate the protein modularity maps [17]. Evidence for interacting protein pairs in a complex that show mRNA coexpression is for instance provided by Dezso et al. [18], and is also available from human interactome experimental work [19] and tissue-specific interactome analysis [20].

Our first methodological step aims to try to identify the role of the described factors. Since in the proposed PIN list different contributions from the “module drivers” may be expected, we attempt to represent both cell cycle effects and gene co-expression peak signatures within the modular maps retrieved by the chosen methods. A comparison can then be made between static maps of PIN modules (usually addressed as “snapshots”), and refined maps featuring expression data recorded during the cell cycle phases.

The second methodological step involves modularization, and the assessment of the module quality and evaluation of the map structure are based on comparisons between *cePIN-2*, *cePIN-1*, and *cePIN* maps, whose modules necessarily depend on the adopted network partitioning methods. In summary, our PIN modularization approach is constructed to reflect both static and dynamic aspects, and is conditioned on particular modular architectures.

4.2.3 PIN Topology

A first interest in topological network properties is in its distributional properties in relation to the connectivity inherent to the protein map. In general, a power law is observed in a network when for the number of nodes of degree k and for suitable chosen α, a proportionality to $k^{-\alpha}$ holds. Thus, given a probability distribution $p(k)$, we have

$$p(k) \propto k^{-\alpha} \tag{4.1}$$

In Figure 4.1, we offer a view of the fragmented PIN, then fit power laws to its extracted components, and finally report the computed exponents, which result to be $\alpha = 3.5$ for *cPIN*, $\alpha = 2.6$ for *cePIN-2*, $\alpha = 2.97$ for *cePIN-1*, and $\alpha = 2.16$ for *cePIN*.

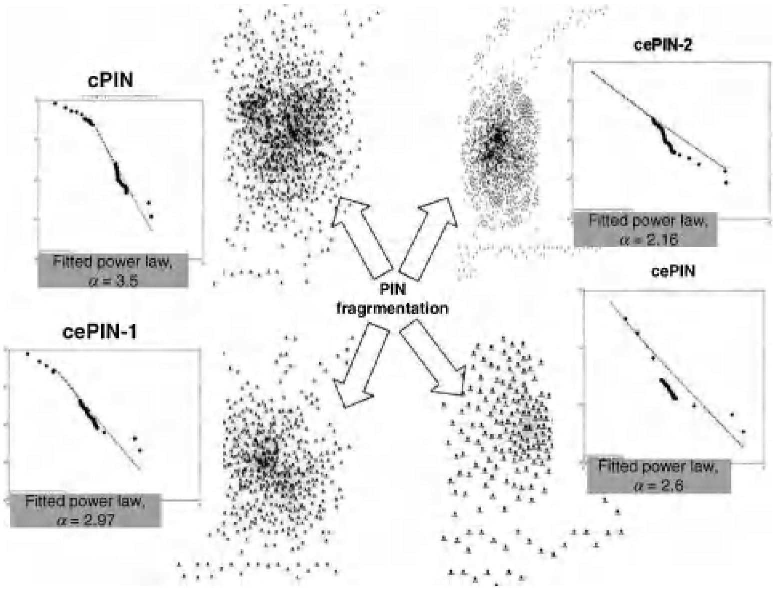


FIGURE 4.1 Scatters of cell cycle PIN and fitted power laws. Maps appear with different sparsity levels and the fitted power laws reflect the combined role of various drivers.

4.2.3.1 Modularity by Communities

Modular structures characterize PIN [21] and can be retrieved by several methods and algorithms inspired by different principles (see for a wide review and examples [22]). Accordingly, different search strategies have been employed to achieve PIN optimal modularity, such as deterministic (divisive and greedy algorithms) and stochastic (random walks) approaches. In general, the suboptimal solutions found from such algorithmic approximations receive further biological validation through multiple annotations involving protein complexes, GO categories, and pathways.

Notably, the different module structures can be also characterized by topological properties; accordingly, topological features can be used to analyze networks at various granularity (ranging from globally to locally). We applied two popular methods and retrieved two kinds of module structures, communities and cores. The community finding method works through the maximization of a *Q-modularity* function, and is based on a greedy optimization algorithm [23]. This very popular procedure iteratively merges module pairs originated by seeds and continues to expand by monitoring a modularity index that keeps increasing until a gain is detected, otherwise it stops. The resulting communities are nonoverlapping, but in our map representations we try to bypass such limitation by inserting cross-linking relationships between communities when they are found to exist.

Figure 4.2 reports the outcomes from *Q-modularity* maximization, with the typically cascading patterns from the algorithmic learning steps; in particular, the sudden pattern drop occurs after that the maximal gain has been reached and no further maximal value can be reached for the optimization function. The latter is simply defined as a difference between links belonging to modules in a network and links that are expected when a network of equivalent size but randomly placed vertices is considered,

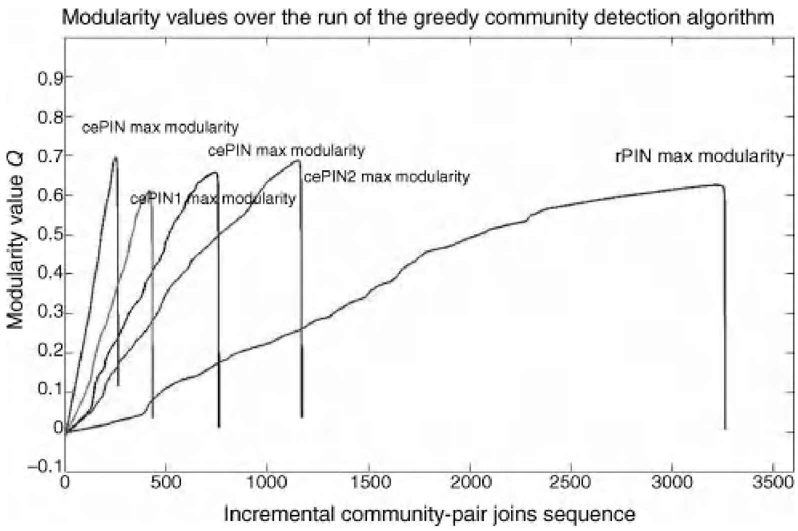


FIGURE 4.2 Comparative modularity convergence.

such that a nonmodular condition is represented. In particular, a network partition in N modules with m_i and m_j linked by e_{ij} appears in the modularity function as follows:

$$Q = \sum_i \left[e_{ii} - \left(\sum_j e_{ij} \right)^2 \right] \tag{4.2}$$

In particular, links that connect nodes within a module i are compared with all links from any other module j connected to module i. A good partition into modules leads to $Q \sim 1$, while random (i.e., poor modularity) would deliver $Q \sim 0$, thus meaning that the fraction of modular and randomized links is not significantly different. In particular, a generally accepted rule is that values $Q > 0.3$ may already suggest the presence of modular structure. Overall, modular partitions obtained by this procedure show relatively dense intramodular links and sparse intermodular links, which reflects the presence of few local maxima capturing the most relevant information about the internal network organization.

4.2.3.2 Modularity by Cores

MCODE [24] is another well-known method that exploits network local density areas to identify clusters supposed to match protein complexes. In particular, the dependence between nodes is represented by structures called "cliques," and a hierarchy of modules of different clique size is obtained at the end.

A clique is a maximally connected structure, that is, a network in which every pair of distinct node is connected by a link. MCODE starts its exploration from locally dense regions from a clustering coefficient computed a given node, that is, $CC_i = \frac{2n}{k_i(k_i-1)}$, where k_i is the size of the neighborhood of node i, and n is the number of edges in the neighborhood.

A k-core is delivered by the method based on the clique benchmark, and represents a network of minimal degree k after that all the nodes with degree less than k have been successively eliminated. As several groups are formed at each k, an internal ranking through scores based on node weighting is obtained. The output for our PIN list is reported in Table 4.2 and indicates various cores computed at different k values, The relevance of each modules can finally be biologically validated against known protein complexes.

4.3 RESULTS

4.3.1 Community Maps

The communities retrieved for each subnetwork are listed in Table 4.1, and Figure 4.2 describes the comparative algorithmic performance through the maximal values achieved by Q for each PIN. Then, Figures 4.3–4.6 represent instead

TABLE 4.1 Detected Communities (Relatively Small Ones)

rPIN	cPIN	cePIN	cePIN-1	cePIN-2
67 (48)	28 (22)	31 (13)	26 (17)	43 (30)

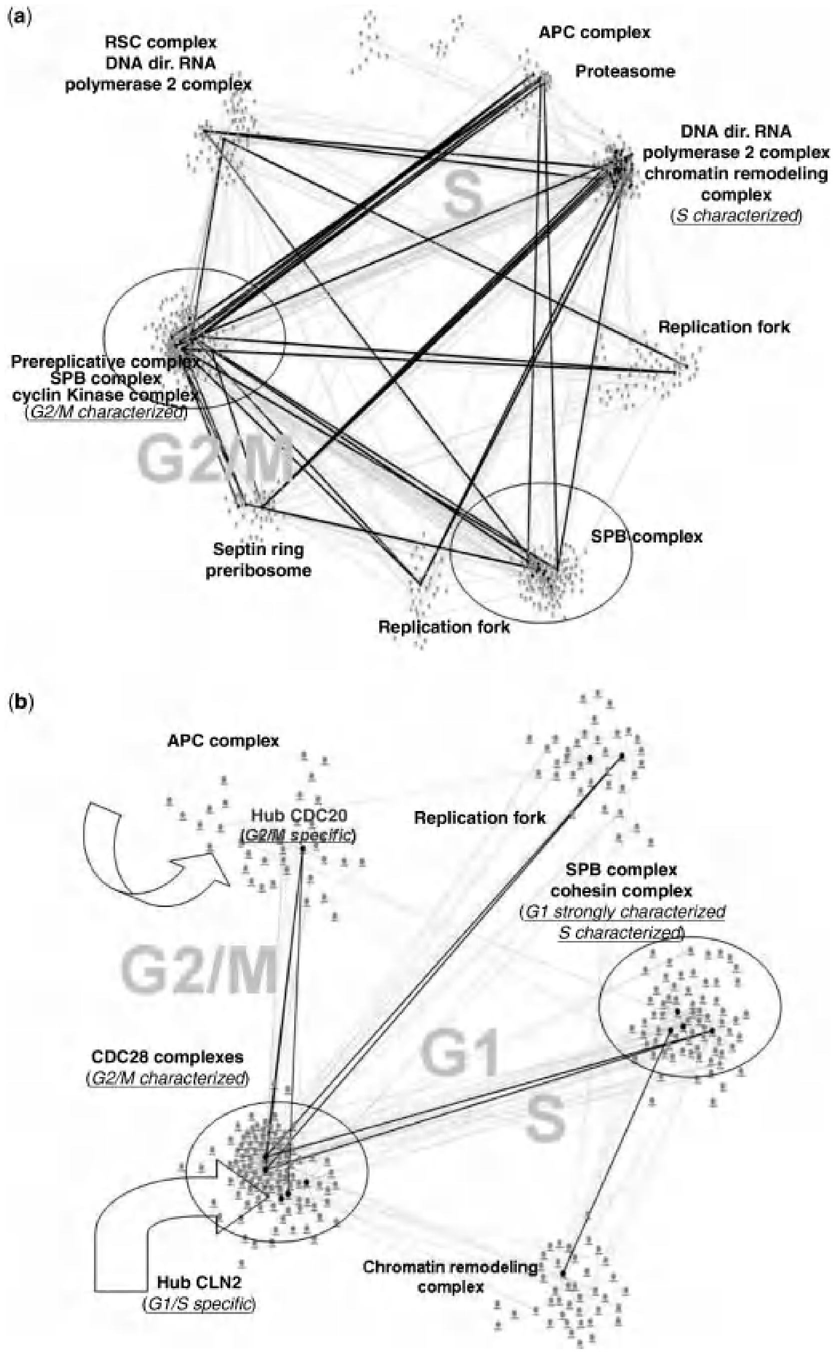


FIGURE 4.3 Community maps: (a) *cPIN* and (b) *cePIN-1*. The thick arrows point out the indicated hubs.

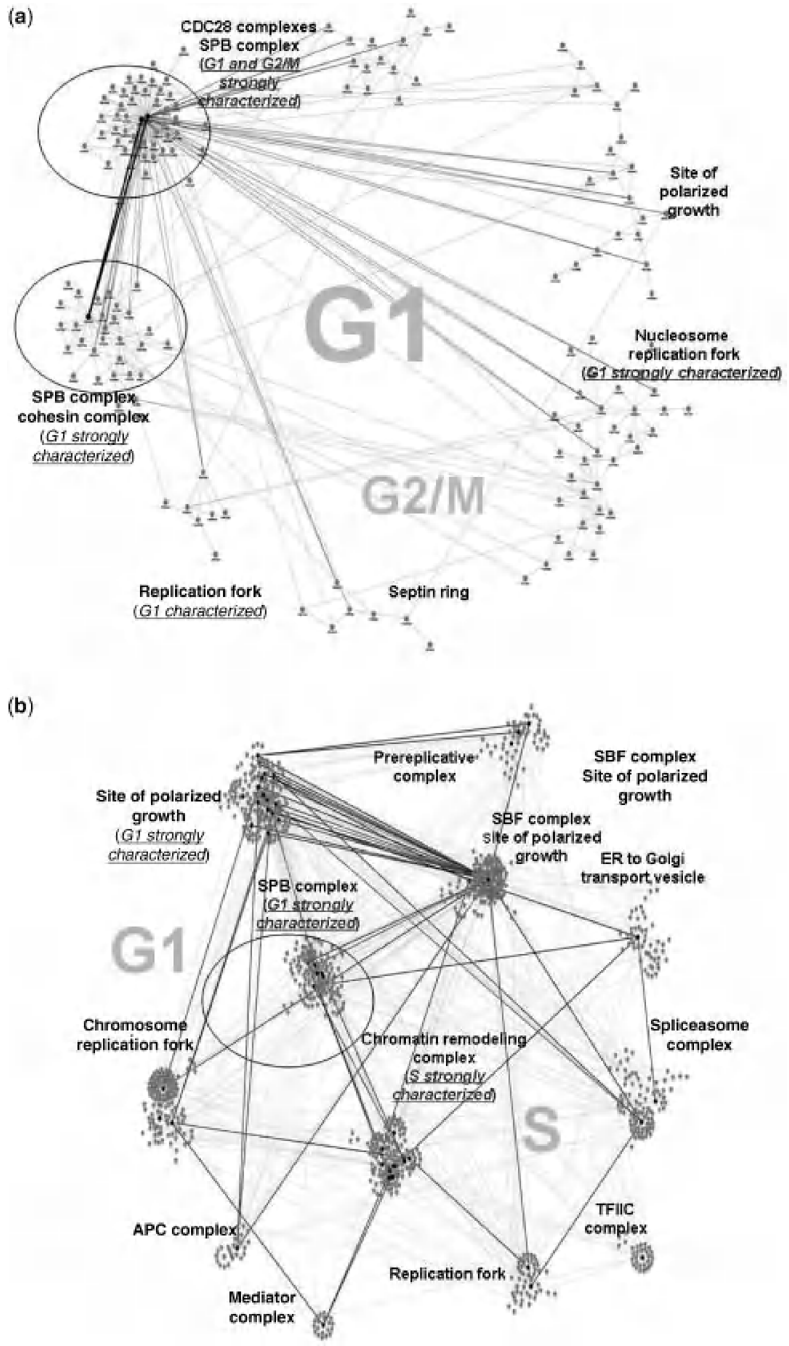

(a)
CDC28 complexes
SPB complex
(G1 and G2/M strongly characterized)
Site of polarized growth
G1
Nucleosome replication fork
(G1 strongly characterized)
SPB complex cohesin complex
(G1 strongly characterized)
G2/M
Replication fork
(G1 characterized)
Septin ring
(b)
Prereplicative complex
SBF complex Site of polarized growth
Site of polarized growth
(G1 strongly characterized)
SBF complex site of polarized growth
ER to Golgi transport vesicle
SPB complex
(G1 strongly characterized)
G1
Chromosome replication fork
Chromatin remodeling complex
(S strongly characterized)
Spliceasome complex
S
APC complex
TFIIC complex
Replication fork
Mediator complex

FIGURE 4.4 Community maps: (a) *cePIN* and (b) *cePIN-2*.

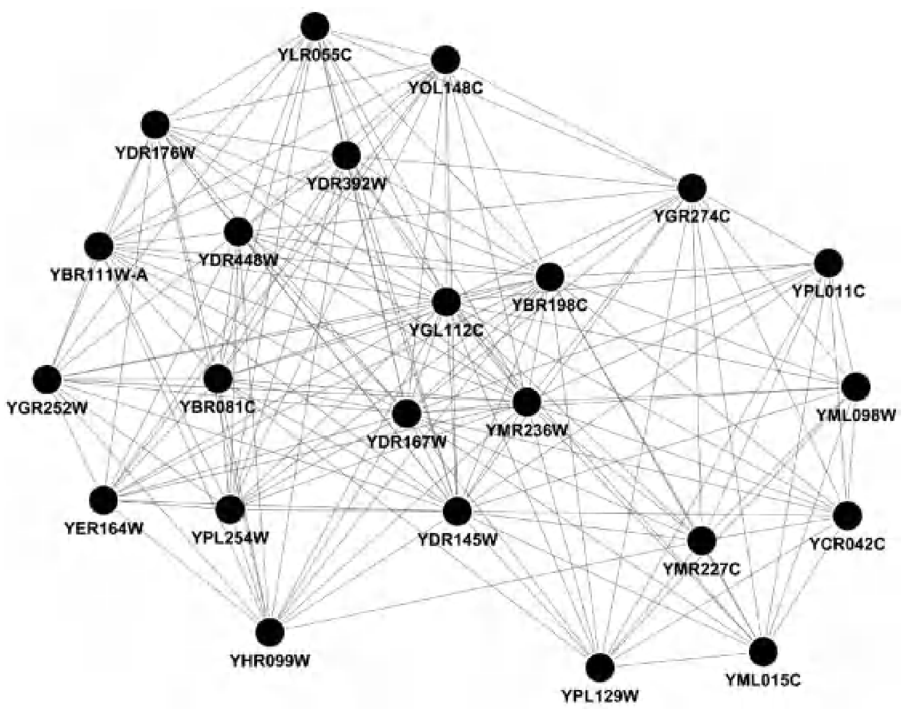


FIGURE 4.5 Best *k*-core *cPIN*.

examples of community maps computed for the relevant subnetworks, that is, *cPIN, cePIN, cePIN-1*, and *cePIN-2*, respectively. A characterization for the module maps is obtained by the cell cycle phases, that is, the S (synthesis, DNA replication) and M (mitosis, chromosome separation) phases together with the remaining G_1 and G_2 interphases. For both *cePIN-1* and *cePIN* the corresponding KEGG pathways have also been reported as supplementary data (SD).

4.3.1.1 Analysis

The analysis of the maps has been pursued according to some simple rules aimed to provide useful interpretation. As a first rule, compact maps have been formed by postprocessing the retrieved communities. In particular, hub proteins have been emphasized due to their relevant role in terms of network stability. Concerning their definition, since a connectivity threshold for distinguishing a protein hub is required but is not unique, after some tests we have retained useful hubs with at least 15 connected proteins for our purposes.

Thus, the resulting community maps are hub-centered, and allow for a coarse-grained evaluation. We compared various hubs providing community characterization with a balance between module cohesiveness and cross-links then validated through annotation. The visualized communities show darker nodes representing the hub

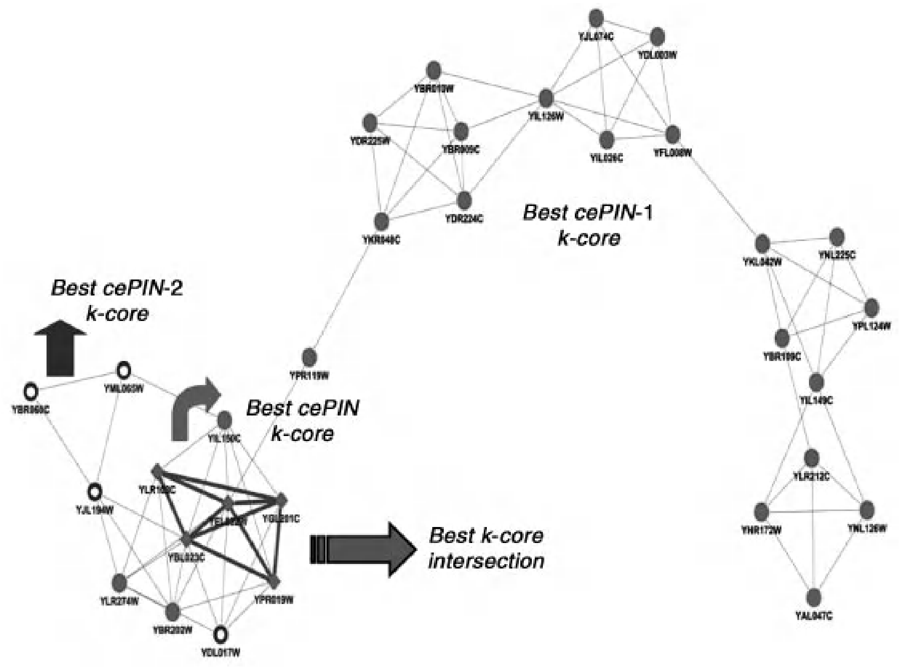


FIGURE 4.6 Best *k*-core intersection from *cePIN*, *cePIN-1*, and *cePIN-2*.

proteins, and thick links emphasizing "hub-hub" cross-connections, together with lighter links connecting hubs to nonhub proteins. All the other links have been hidden (examples with complete associations are reported in SD).

The comparative examination of the community maps suggests some remarks. In Figure 4.3a the map obtained from *cPIN* with just one driver (interactions) delivers a quite redundant map, and extra cell cycle related communities appear due to possible algorithmic errors. In the other maps, it is crucial to check the contribution from coexpression-induced dynamics. In particular, Figure 4.3b reports the *cePIN-1* map; only half of the previously shown communities in *cPIN* are left.

The *SPB* or *spindle pole body* complex is the microtubule organizing center in the yeast cells, while the *Cohesin* complex is responsible for binding the sister chromatids during synthesis through the G2 phase and into M phase. The *CDC28* complexes appear too, with the *cdk* or *cyclin-dependent kinase* that phosphorylates a variety of target substrates, together with the *APC* or *anaphase-promoting* complex (cyclosome) that promotes metaphase–anaphase transition by ubiquitinating its specific substrates (such as mitotic cyclin and anaphase inhibitor), then subsequently degraded by the proteasome.

One of the communities is characterized by both the *SPB* and the *Cohesin*, while another linked community is characterized by the *CDC28* complexes. Also, communication between the latter complexes and the *APC*-characterized community occurs in hub–hub style between *cdc28* and *clb2* with *cdc20*, and also in non hub–hub style through the hub *cln2* protein. In particular, *clb2* is a B-type cyclin that activates *cdc28* to promote the transition from G2 to M phases, and then its degradation is activated by the *APC/Cdc20* to promote mitotic exit [25].

Then, the hub *cdc28* links to *SPB* [26] and with reference to the same community pair, the hub *cdc5* that is involved in regulation of sister chromatid separation [27] interacts with the nonhub *scc1* cohesin protein. Finally, the hub *sth1* protein, which is a component of the *RSC* or *chromatin remodeling* complex associating the cohesin with centromeres and chromosome arms [28], also interacts with the nonhub *scc1* cohesin protein.

The KEGG pathway map reported in the SD (*KEGGSDf1.jpg* file) localizes (marked by circles) especially the *APC* closeness to *cdc20*, and the *Cohesin* connection to *CDC28*. In terms of phases, there appears a mix of them involving G1, G2/M, and S. In particular, *cdc20* in G2/M phase justifies the hub–hub cross-links between *APC* and *CDC28*. Then, the G1 and S specificity of both the *SPB* and *Cohesin* complexes comes also from *cln2*, which is linked to *APC* too.

By looking at Figure 4.4a, the first remark is for the G1 phase that strongly characterizes the *cePIN* community map. A community is characterized by both *CDC28* and *SPB*, with the latter also sharing a community with the *Cohesin* complex. The two main modifications that can be observed refer to the hub *cdc20* that becomes a nonhub protein, and *APC* (activated by *cdc20*) that disappears (i.e., due to inactivated *cdk* in S/M phases that allows the cell to exit from mitosis). The strong G1-characterization can justify the absence of the link between *APC* and *cdc20* as the latter acts in M and not G1 phase where it acts with another protein. As both *Cohesin* and *SPB* remain, the presence of the latter in G1 is possibly justified by the *SPB* duplication during this phase (see Refs. 29,30).

Notably, the *cePIN* modularity map appears reduced, and highly G1-specific. As the hub proteins have been found in just two communities, a very localized map results from the highest possible influence of peak coexpression signatures. In SD (*KEGGSDf2.jpg and KEGGSDf3.jpg* files), the associated KEGG pathways are reported for the two main communities.

The examination of *cePIN-2* in Figure 4.4b reveals a map significantly denser than before. Quite clearly, the dimensions of each PIN are different, but it is important to underline the following aspects. This final transition naturally allows for introducing extra communities that are not related to the cell cycle, for instance, *mediator, spliceasome, ER to Golgi transport vesicle, TFIIC* complexes. Thus, a redundancy of communities appears compared to *cePIN-1*, as it was expected from the introduction of non cell cycle proteins. In terms of phases, G1 and S are both present, but with lesser characterization than before.

Overall, the community maps reflect quite well the co-expression dynamics, and in particular identify for *cePIN* a phase-specific modularization due to peak signatures. When relaxing the constraint of peak–peak protein interaction, the distinct impact of biological process on modularization emerges, which appears from *cePIN-1*. Then, when all proteins are considered as in *cePIN-2*, the community map substantially

TABLE 4.2 Number of MCODE-Detected *k*-Cores at *k* Ranging Between Minimum of 2 and Maximum of 13 Across PIN

rPIN	cPIN	cePIN	cePIN-1	cePIN-2
2-core **85**	2-core **33**	2-core **7**	2-core **6**	2-core **13**
3-core **39**	3-core **18**	3-core **2**	3-core **2**	3-core **4**
4-core **26**	4-core **11**		4-core **1**	4-core **2**
5-core **12**	5-core **7**			
6-core **12**	6-core **5**			
7-core **8**	7-core **4**			
8-core **6**	8-core **4**			
9-core **5**	9-core **3**			
10-core **3**	10-core **2**			
11-core **1**	11–13-core **1**			

Number of MCODE-detected *k*-cores are given in boldface.

diversifies from strictly cell cycle based modules. Modularity is thus shaped by the drivers and the dynamics characterized accordingly.

4.3.2 Core Structures

The implementation of MCODE requires that some parameters are set in order to compute the network-scoring values. For instance, we set degree cutoff = 2 and node score cutoff = 0.2, and then proceeded by fixing other values, such as haircut = true, fluff = false; *k*-Core = 2; max. depth from seed = 100. The list of the retrieved *k*-cores has been reported in Table 4.2. We have then comparatively evaluated the results based on these *k*-cores, and reported them in Figures 4.4–4.6.

We looked at *k*-core modularity according to two different strategies. First, we have considered the most important structure provided by the method, and thus the best *k*-cores for each PIN. In particular, the "best *k*-core intersection" has been identified and targeted. The best *k*-core of *cPIN* has been reported apart in Figure 4.4 as it differentiates from the other cases in terms of data sources. Since the best *k*-cores (Fig. 4.5) represent modules with the highest scores, they can be considered locally optimal at the network scale. When annotation is done, the best *cePIN* *k*-core indicates a little module with highly coexpressed genes, while the proteins address the *MCM* complex involved in initiation and regulation of DNA replication. Also the best *cePIN-2* *k*-core includes the latter complex, but through a larger module annotating for the prereplicative complex. Finally, the best *cePIN-1* *k*-core results the most structured and extended one.

We observe the presence of an area of intersection that involves the whole *cePIN* core, and appears totally included by the other two structures. The most structured *k*-core with regard to cell cycle belongs to *cePIN-1*, and is also determined by co-expression. When we relax the requirement that all proteins refer to the cell cycle, as with *cePIN-2*, then the best *k*-core structure reduces drastically and converges to that

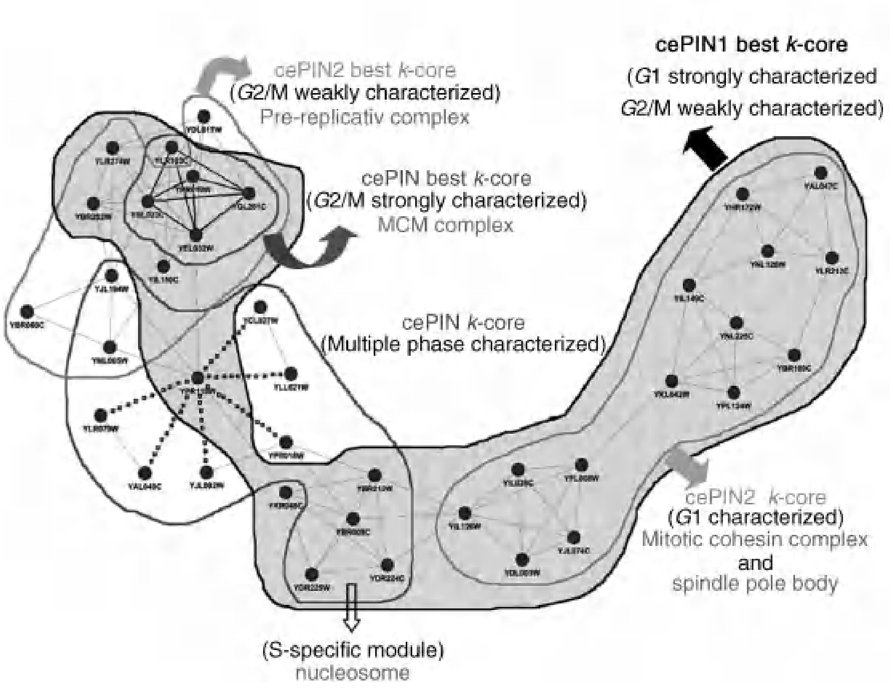


FIGURE 4.7 Best *k*-cores and innermost *k*-cores.

observed for *cePIN* by corresponding to the bottom-left region of the *cePIN-1* *k*-core (except for a few newly established associations).

Second, considering as in Figure 4.6 the whole-layered coreness structure requires that for each PIN the innermost or most densely connected core is computed, which equivalently requires the exploration of the sequence of cores at all *k*, and including therefore, the best *k*-core. In some cases, as for *cePIN-1*, we observed a convergence between the best and the innermost *k*-cores, although the same convergence was not observed in the other two cases. The annotation is reported for the marked regions defining cores, and an interesting aspect regards the presence of a module with dotted links between nodes that reports checkpoints of the cell cycle through the cyclins.

The analysis of Figure 4.7 involves envelopes to visualize different cores belonging to the various PIN, and distinguish between the best and the innermost *k*-cores at varying *k*. Then, all characterizing phases have been indicated for each *k*-core. It results that the *cePIN* is strongly G2/M characterized (particularly through the best *k*-core) and presents an S-specific module, while *cePIN-1* is strongly G1-characterized (and marginally G2/M-characterized too), and *cePIN-2* is similar to *cePIN-1* but less characterized.

We stress the fact that both strategies consider exclusively the inherent coreness structure detected by MCODE, but exclude other possible structures embedded at different resolutions [31]. In particular, we have emphasized how the cohesiveness of cores depends by the module drivers, and have evaluated the extent by which they contribute to form and differentiate cores and communities in each PIN. Overall, the clique-based modular organization is influenced by both drivers. The effect of accounting for their simultaneous presence, as in *cePIN*, instead of considering them more separately, is that more compact protein cores are found as expected.

4.3.3 On Network Entropy

The concept of network entropy is interesting for several reasons (see the review and developments presented extensively in Ref. [32] and also including characterizations in biology). Entropy as a measure of uncertainty can characterize PIN at both global and local levels. In particular, when local dynamics are investigated through modules, entropy can be used to corroborate the reliability of connectivity maps that are derived from modules. This strategy is pursued in our work in relation with the modular structure already extracted, that is, cores and communities.

Estimating entropies from finite samples remains in general a complicated task due to statistical fluctuations, and this limitation holds also for sampled networks. In the limit of sample size, when steady-state conditions are achieved, the Shannon entropy associated with the network distribution $p(k)$ is given by

$$E^s = -\sum_{k=1}^{\infty} p(k)\ lnp(k) \tag{4.3}$$

The same formula has been applied in this work to approximately investigate the uncertainty levels referred to cores and communities computed over the cell cycle related subinteractomes. As a result, Figures 4.8–4.10 have been generated. As a first observation, communities are larger modules compared to cores, and thus show in general bigger values of the estimated entropy. Protein hubs are largely responsible for the latter values, as it appears from the top plots in the first two figures (the last one has a long list of hubs that we report apart).

Notably, when the number of nodes is considered in the X axis (see bottom plots), we observe that cores shift toward the origin (i.e., relatively smaller cores for given entropy levels) the more the cell cycle effects are relaxed. This means that the cell cycle dynamics may be monitored by measuring entropy in relation to the module sizes, and what we may observe reflects the fact that random effects play a major role under relaxed conditions.

4.4 DISCUSSION AND CONCLUDING REMARKS

In the present study, we observed that the consideration of coexpression dynamics enables a strong localization power in PIN that reinforces module detection and characterizes phase-specific modules. We also found that the analysis centered on phases annotated in cores and communities reveals a certain convergence between the various PIN. Even if cores do not show the strong G1 characterization in cePIN that is

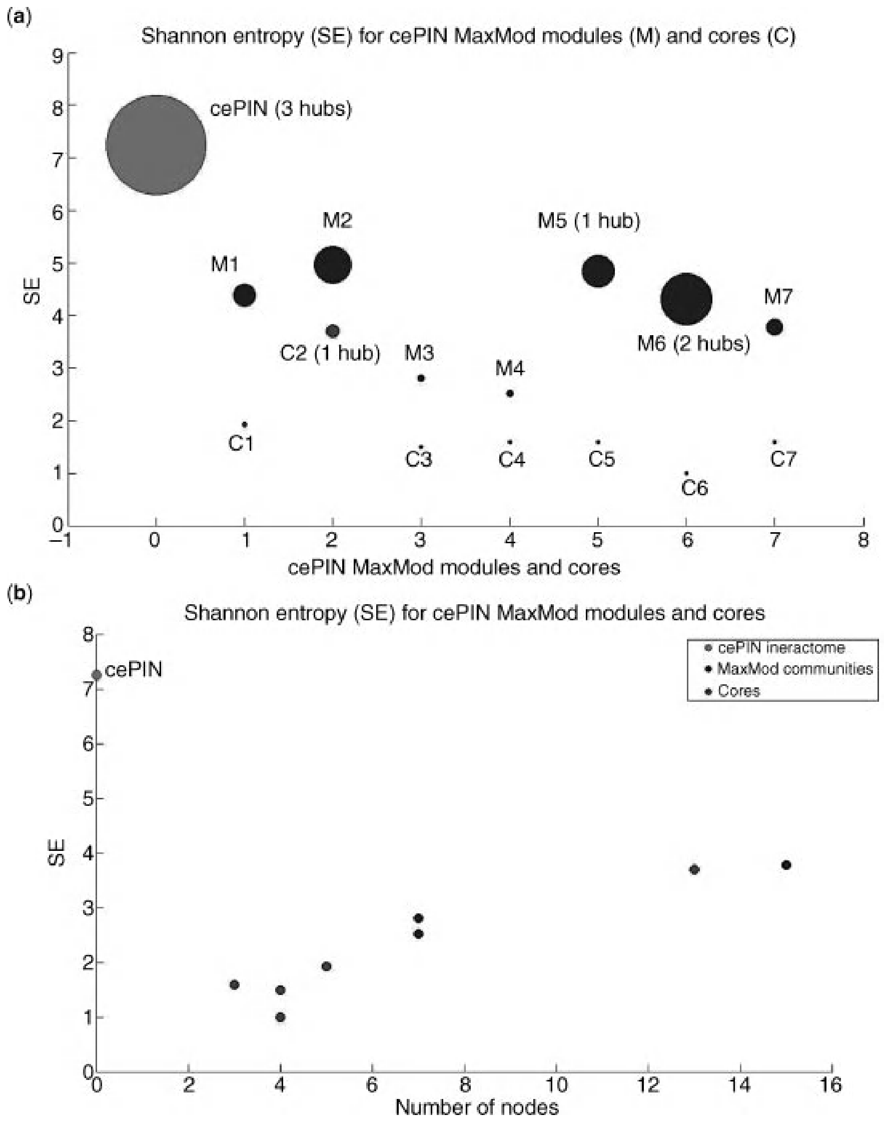


FIGURE 4.8 cePIN entropies with hubs emphasized for both (a) cores and communities and (b) sorted by size.

observed with communities, the module annotation quality relatively to the matched complexes depends necessarily on the different resolutions allowed by the methods. Thus, the main driver of modularization is resolution-dependent, as expected.

The proposed approach of PIN fragmentation offers the possibility of looking at a compilation of PIN selected according to various criteria, for instance cell cycle

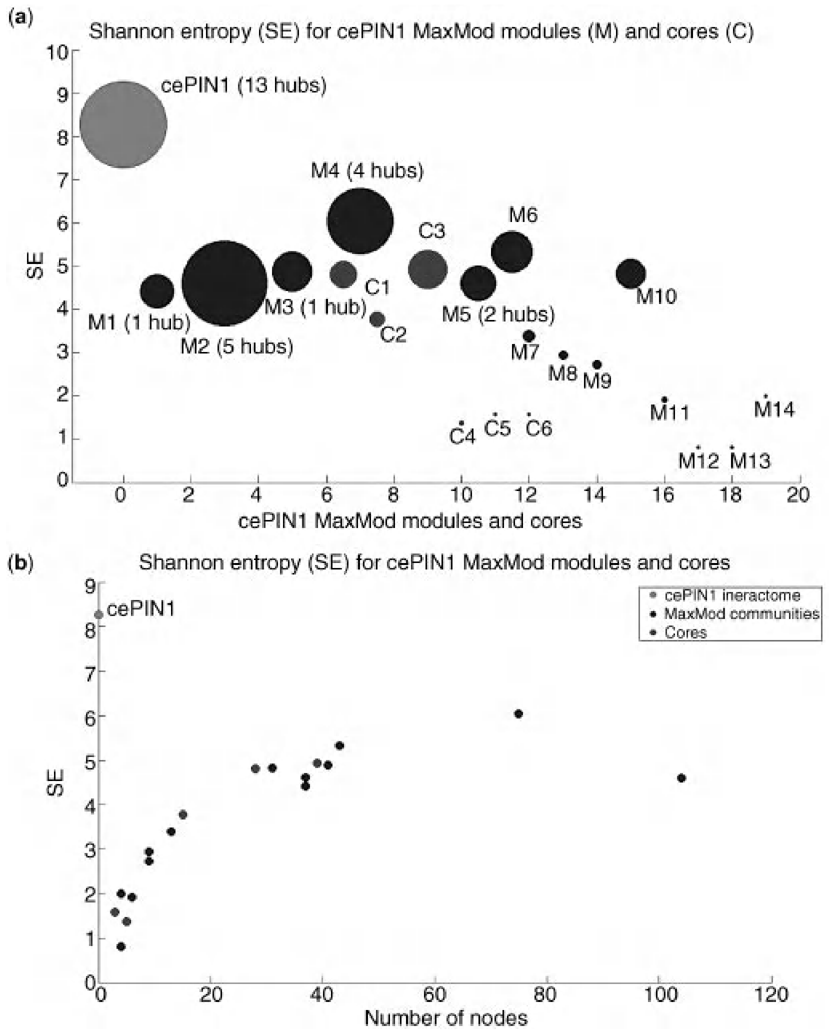


FIGURE 4.9 cePIN1 entropies with hubs emphasized for (a) both cores and communities and (b) sorted by size.

specificity. An advantage is that comparative evaluations with regard to both general topological features and modularity can refer to multiple PIN referred to a common source. Thus, our study has referred to what we defined an “affine” PIN list, which opened the possibility to explore PIN dynamical aspects. Due to the consideration of time-course experiments and their recorded gene expression peak signatures, we could center the rest of the analysis on modularity with reference to the cell cycle role in determining the “interaction driver,” and the integrated gene measurements role in determining the “expression driver.”

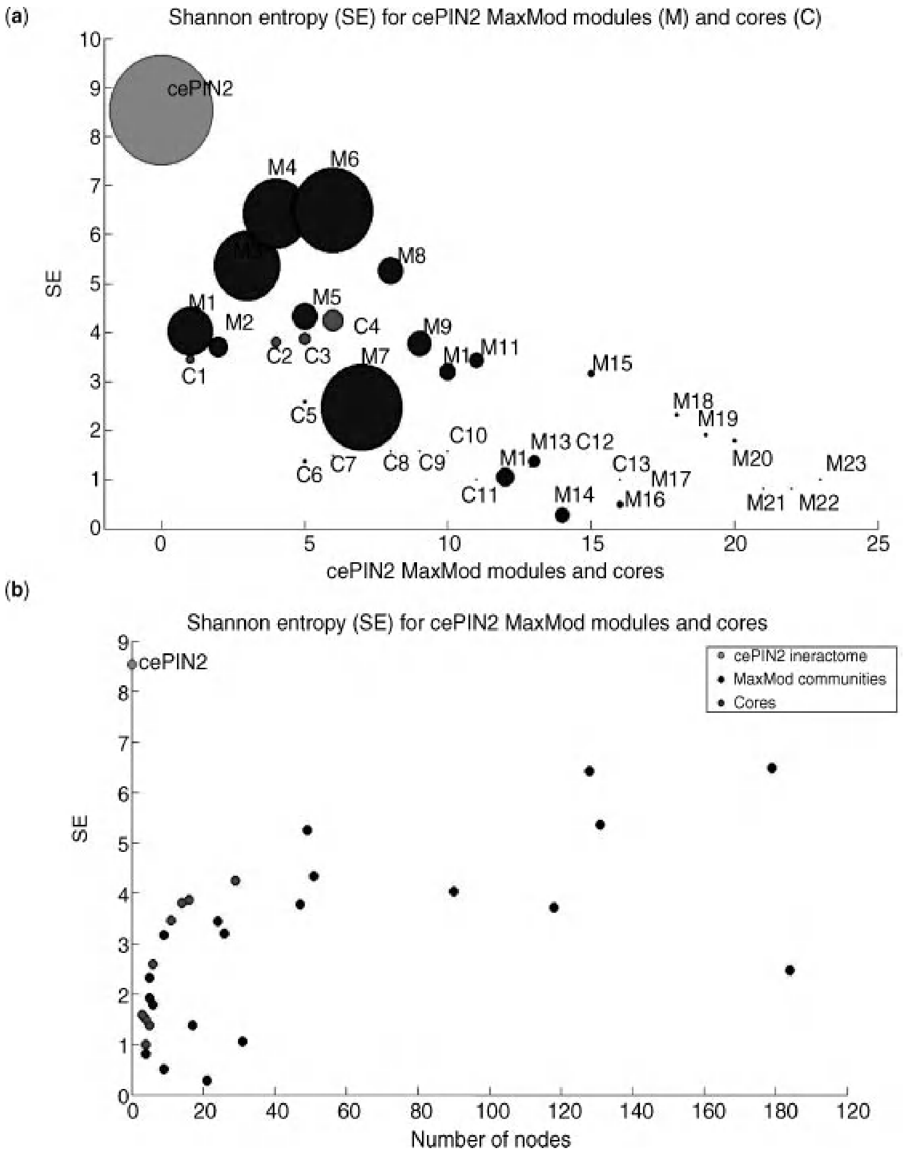


FIGURE 4.10 cePIN2 entropies with hubs emphasized for both (a) cores and communities and (b) sorted by size.

Modularization has been mainly investigated relatively to clique-based methods. The comparison of community maps offered a coarse-grained analysis useful to verify what complexes are matched by modules and up to what extent, together with the involved pathways. Furthermore, a module characterization by phases can help monitoring the retrieved communities under different dynamic conditions. Then, the hierarchical fine-grained analysis obtained by comparing best and innermost *k*-cores was useful to point out the role that both module drivers may play at intramodular resolutions.

We have emphasized the major variation in the community maps by concentrating the analysis on hub proteins, thus reducing the dimensionality and complexity of modularity maps, and then validating at the protein pathway level the established community links. An increased phase localization power within the protein maps is observed when peak signatures are considered. Especially the modularization detected through *k*-cores tends to concentrate due to peak signature influence and intersection between both best and innermost hierarchical structures.

Two final notes concern a specific comment and a more general consideration. The modularization induced by the employed methods remains conditioned by the different resolutions that they allow to uncover. Depending on the network, differentiated modularity can be observed and maps are revealed according to the combined role of process-driven interactions and coexpression dynamics. In general, we believe that the development of differential network modularization approaches for examining cellular systems may be useful for inferring the complexities that typically characterize high-dimensional and heterogeneous biological data. In particular, the proposed approach for PIN could be extended to biological contexts where a crucial goal is establishing a role for biological processes involved in disease.

ACKNOWLEDGMENT

The authors thank Sardegna Ricerche for support.

SUPPORTING INFORMATION

List of annotation tables
GO-comm-annotation.doc, *GO-core-annotation.doc*, *Core-phase-annotation.xls*, *Comm-phase-annotation.xls*.

Figures:
cePIN1-hdp.jpg, *cePIN2-hdp.jpg*, *cePIN2-hdp-redEd.jpg*, *cePIN1-hdp-redEd.jpg*, *KEGGSDf1.jpg*, *KEGGSDf2.jpg*, *KEGGSDf3.jpg*.

REFERENCES

1. M. Vidal, Interactome modeling, *FEBS Lett.* **579**, 1834–1838 (2005).
2. G.T. Hart, A.K. Ramani, E.M. Marcotte, How complete are current yeast and human protein interaction networks? *Gen. Biol.* **7**(11), 120 (2006).

3. C. von Mering et al., Comparative assessment of large-scale data sets of protein-protein interactions. *Nature* **417**, 399–401 (2002).
4. M.E. Cusick, *et al.*, Literature-curated protein interaction datasets, *Nat. Meth.* **6**, 39–46 (2009), doi:10.1038/nmeth.1284.
5. S. Brohee, J. van Helden, Evaluation of clustering algorithms for protein–protein interaction networks, *BMC Bioinform.* **7**(488), 1–19 (2006).
6. S. Fortunato, M. Barthelemy, Resolution limit in community detection. *PNAS* **104**, 36–41 (2007).
7. M. Roswall, C.T. Bergstrom, An information-theoretic framework for resolving community structure in complex networks. *PNAS* **104**(18), 7327–7331 (2007).
8. A. Clauset, C.R. Shalizi, M.E.J. Newman, Power–law distributions in empirical data. *SIAM Rev.* **51**(4), 661–703 (2009).
9. L.dF. Costa, F.A. Rodrigues, G. Travieso, P.R. Villas Boas, Characterization of complex networks: a survey of measurements. *Adv. Phys.* **56**, 167–242 (2007).
10. P. Durek, D. Walther, The integrated analysis of metabolic and protein interaction networks reveals novel molecular organization principles. *BMC Syst. Biol.* **2**, 100 (2008).
11. C. Huthmacher, C. Gille, H.G. Holzhütter, A computational analysis of protein interactions in metabolic networks reveals novel enzyme pairs potentially involved in metabolic channeling. *J. Theor. Biol.* **252**(3), 456–464 (2008).
12. C. Huthmacher, C. Gille, H.G. Holzhütter, Computational analysis of protein–protein interactions in metabolic networks of E. coli and yeast. *Genome Inform.* **18**, 162–172 (2007).
13. T. Reguly, et al., Comprehensive curation and analysis of global interaction networks in *Saccharomyces cerevisiae, J. Biol.* **5**, 11 (2006).
14. U. de Lichtenberg, L.J. Jensen, S. Brunak, P. Bork, Dynamic complex formation during the yeast cell cycle, *Science* **307**, 724–727 (2005).
15. http://mips.helmholtz-muenchen.de/genre/proj/yeast/
16. http://www.yeastgenome.org/
17. I.A. Maraziotis, K. Dimitrakopoulou, A. Bezerianos, An *in silico* method for detecting overlapping functional modules from composite biological networks. *BMC Syst. Biol.* **2**(93), (2008), doi:10.1186/1752-0509-2-93.
18. Z. Dezso, Z.N. Oltvai, A.L. Barabasi, Bioinformatics analysis of experimentally determined protein complexes in the yeast *Saccharomyces cerevisiae*. *Gen. Res.* **13**, 2450–2454 (2003).
19. J.F. Rual, et al., Towards a proteome-scale map of the human protein–protein interaction network. *Nature* **437**, 1173–1178 (2005).
20. P.V. Missiuro, et al., Information flow analysis of interactome networks. *PLos Computat. Biol.* **5**(4), e1000350 (2009).
21. A.C. Gavin, Proteome survey reveals modularity of the yeast cell machinery. *Nature* **440**, 631–636 (2006).
22. A. Clauset, Finding local community structure in networks. *Phys. Rev. E* **72**, 026132 (2005).
23. M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks. *Phys. Rev. E* **69**, 026113 (2004).
24. G.D. Bader, W.V. Hogue, An automated method for finding molecular complexes in large protein interaction networks, *BMC Bioinform.* **4**(2), 1–27 (2003).

25. D.O. Morgan, Regulation of the APC and the exit from mitosis. *Nat. Cell Biol.* **1**, E47–E53 (1999), doi:10.1038/10039.
26. S. Jaspersen, et al., Cdc28/Cdk1 regulates spindle pole body duplication through phosphorylation of Spc42 and Mps1. *Develop. Cell* **7**(2), 263–274 (2004).
27. G. Alexandru, et al., Phosphorylation of the cohesin subunit Scc1 by Polo/Cdc5 kinase regulates sister chromatid separation in yeast, *Cell* **105**(4), 459–472 (2001).
28. B. Wilson, et al., The RSC chromatin remodeling complex bears an essential fungal-specific protein module with broad functional roles. *Genetics* **172**, 795–809 (2006).
29. S. Uzawa, Spindle pole body duplication in fission yeast occurs at the G1/S boundary but maturation is blocked until exit from S by an event downstream of Cdc10+. *MBoC* **15**(12), 5219–5230 (2004).
30. L. Vardy, T. Toda, The fission yeast gamma-tubulin complex is required in G(1) phase and is a component of the spindle assembly checkpoint. *EMBO J.* **19**(22), 6098–6111 (2000).
31. E. Marras, E. Travaglione, E. Capobianco, Sub-modular resolution analysis by network mixture models. *Statist. Appl. Genet. Mol. Biol.* **9**(1), 19 (2010).
32. M. Dehmer, A. Mowshowitz, A history of graph entropy measures. *Inform. Sci.* **181**, 57–78 (2011).
33. A.L. Barabasi, R. Albert, Emergence of scaling in random networks, *Science* **286**, 509–512 (1999).
34. R.J. Cho, et al., A genome-wide transcriptional analysis of the mitotic cell cycle, *Mol. Cell* **2**, 65–73 (1998).
35. J.D. Han, et al., Evidence for dynamically organized modularity in the yeast protein–protein interaction network. *Nature* **430**, 88–93 (2004).
36. P.T. Spellman, et al., Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization, *Mol. Biol. Cell*, **9**(12), 3273–3297 (1998).

5

INFLUENCE OF STATISTICAL ESTIMATORS ON THE LARGE-SCALE CAUSAL INFERENCE OF REGULATORY NETWORKS

Ricardo de Matos Simoes and Frank Emmert-Streib

5.1 INTRODUCTION

The inference of gene regulatory networks aims to unveil the causal structure of the relations among genes in a cellular system from gene expression data [1–4]. Gene regulatory networks (GRNs) allow to organize genes according to their gene expression dependency structure and aim to complement the understanding of the molecular structures and processes in complex organismal cellular systems. The vast amount of gene regulatory network inference methods that are being developed are gaining more and more popularity due to the astonishing increase of high-throughput dataset generation. The challenge of the future is the development of novel statistical methods to benefit from present and new emerging mass data [5,6], for example, from microarray, Chip–Chip [7], Chip–seq, proteomics mass spectrometry, protein arrays, and RNA–seq. Due to the large amount of available samples and cost efficiency, DNA microarrays are still the state-of-the-art data source for gene regulatory network inference. For example, one of the largest data repository of such high-throughput gene expression data is the GEO database [8] that provides a large range of observational [9,10] and experimental gene expression data. Such large-scale datasets for different organisms, perturbation and disease conditions, enable system-wide studies of species, and phenotype-specific gene regulatory networks.

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

The edges in gene regulatory networks represent physical interactions between genes, intermediates, and their products. These can be genes regulated by the same transcription factor (coexpression) and physical interactions from protein complexes, metabolic and signaling pathways. For expression data, inferred interactions have been observed to preferably indicate transcription regulation in *Escherichia coli* [11] but can also correspond to other types of molecular interactions. Many current approaches for gene regulatory network inference result in networks with a high edge density. This leads to difficulties in elucidating the biological relevance and importance of the individual edges. Instead, a network inferred with C3NET is very sparse and represents the core of a gene regulatory network considering only the gene pairs with strongest expression dependencies [12]. The inference of sparse gene regulatory networks is therefore a promising approach to reduce the overall complexity by considering only causal dependencies with the highest signal [13,14].

There are many gene regulatory network inference methods that are based on estimates of mutual information values. Methods for network inference based on mutual information are called relevance-based gene regulatory network inference methods. The first method based on mutual information for GRN inference was introduced by Ref. [15]. Mutual information is a measure of the nonlinear correlation between two random variables (i.e., two genes), for example, estimated from the (individual) and joint entropy of two random variables.

A variety of different mutual information estimators were developed in order to obtain accurate estimates for different assumptions regarding the characteristics of the underlying data. In Ref. [16] mutual information-based gene regulatory network inference algorithms were evaluated for different mutual information estimators, demonstrating that the choice of the MI estimator influences the inference performance for a given GRN approach. We address the question as to what extent the inference performance of the C3NET algorithm is affected so as to determine which estimator is the most beneficial in terms of its inference performance. In addition, we also employ local network-based measures to study the inference performance for edges connected to genes with a high degree and edges from linearly connected genes.

In the first part of this chapter, we describe the C3NET algorithm [12] for the inference of gene regulatory networks. In the second part, we present four common mutual information estimators, an ensemble approach for global inference performance measure and local measures, investigating the influence of the estimators on the networks inferred by C3NET. In the third part, we present numerical results for *in silico* gene expression datasets generated for three Erdös–Rényi networks for various sample sizes.

5.2 METHODS

5.2.1 C3NET

5.2.1.1 C3NET (Conservative Causal Core)

C3NET consists of three main steps [12]. The first step is for estimating mutual information for all gene pairs. In the second step, the most significant link for each gene is

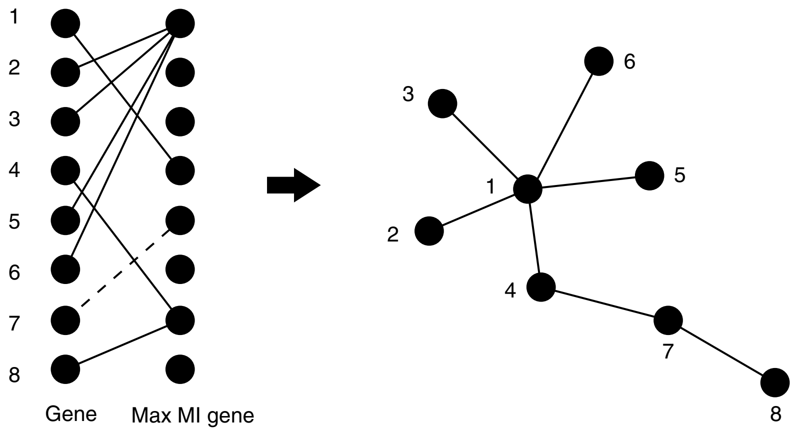


FIGURE 5.1 Principle working mechanism of C3NET. The MI value between gene 7 and 5 (dashed) is not significant.

selected. In the third step nonsignificant links, according to a chosen significance level α, between gene pairs are eliminated. The inferred link in a C3NET gene regulatory network correspond to the highest MI value among the neighbor edges for each gene. This implies that the highest possible number of edges that can be inferred by C3NET is equal to the number of genes under consideration. This number can decrease for several reasons. For example, when two genes have the same edge with maximum MI value. In this case, the same edge would be chosen by both genes to be included in the network. However, if an edge is already present another inclusion does not lead to an additional edge. Another case corresponds to the situation when a gene does not have significant edges at all. In this case, apparently, no edge can be included in the network. Since C3NET employs MI values as test statistics among genes, there is no directional information that can be inferred thereof. Hence, the resulting network is undirected and unweighted. Figure 5.1 shows the principle working mechanism of C3NET. The maximum mutual information value between gene 7 and 5 (dashed line) is not significant. All other genes have significant MI values. This results in a total of 7 edges in the inferred network. As one can see, the structure enabled from this can assume an arbitrary complexity and is not limited to simple connections among genes. The reason for this is that the selection of genes (left-hand side) is directed, whereas the final network (right-hand side) is an undirected network. For a more detailed technical explanation of C3NET, the reader is referred to Ref. [12].

In contrast, common mutual information-based gene regulatory network inference approaches RN [17], ARACNE [18], or CLR eliminate nonsignificant links for all possible gene pairs. This leads to more extensive computational effort and is often circumvented by applying arbitrarily chosen fixed significance thresholds. However, the C3NET approach allows to infer a gene regulatory network without any predefined threshold for the significance of the mutual information.

5.2.2 Estimating Mutual Information

In this chapter, we study the influence of the statistical MI estimator of mutual information values on the inference of regulatory networks. We investigate four different types of estimators that are based on the so-called histogram approach. In the first step, the expression values of two genes are discretized into defined intervals denoted as bins. Mutual information is a measure for the nonlinear dependence between two random variables. Mutual information is defined by the joint probability $P(X, Y)$ of two random variables X and Y [19]. We compute mutual information by

$$I(X, Y) = \sum \sum P(X = x_i, Y = y_i) \cdot \log \frac{P(X = x_i, Y = y_j)}{P(X = x_i) \cdot P(Y = y_i)} \tag{5.1}$$

$I(X, Y)$ is always ≥ 0 and we use base 2 for the logarithm. Mutual information can also be calculated from the marginal and joint entropy measures of the random variables X and Y:

$$I(X, Y) = H(X) + H(Y) - H(X, Y) \tag{5.2}$$

The marginal entropy for a random variable X is defined by

$$H(X) = \sum P(X = x_i) \cdot \log(P(X = x_i)) \tag{5.3}$$

The joint entropy $H(X, Y)$ is defined by

$$H(X, Y) = \sum_{i,j} P(X = x_i, Y = y_i) \cdot \log(P(X = x_i, Y = y_i)) \tag{5.4}$$

In the following, we give a numeral example how the mutual information is computed using the empirical MI estimator for an artificial expression profile for two genes. As example, we generated a linearly dependent expression profile for two genes (Fig. 5.2).

We use three different discretization methods. The first, *equal frequency* method requires the same number of values in each bin. The second, *equal width* method defines the interval for each bin with the same size and the third, *global equal width* method is similar to equal width. Here, the intervals are defined for both random variables simultaneously, rather than separately for each random variable. We use k-proportional interval discretization method [20] that defines the number of bins in dependency of the sample size by $\sqrt{N}$. In Table 5.1, we use the *global equal with* the method to discretized the expression profile for two genes. From the discretized values the count frequencies define the empirical joint probability for each value occurring in the defined intervals can be described for gene A and gene B by the joint probability matrix $P(x, y)$ (Table 5.2).

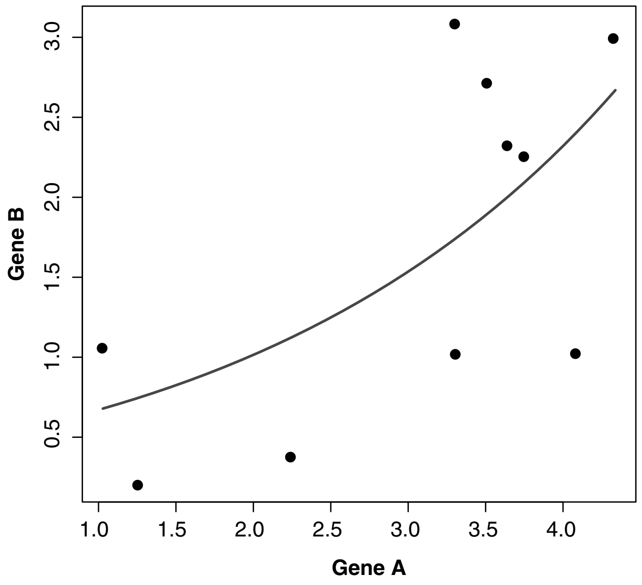


FIGURE 5.2 Example for correlated gene expression for two genes (10 samples). The expression values for gene A are sampled from a normal distribution with $\mu = 3$ and $\sigma = 1$ and the expression values for gene B are defined by gene A with an additional noise ϵ sampled from a normal distribution with $\mu = 0$ and $\sigma = 1$.

The marginal entropies for gene A and gene B are calculated using Equation 5.3:

$$H(G_A) = -\left(\left(\frac{2}{10}\cdot\log\frac{2}{10}\right) + \left(\frac{4}{10}\cdot\log\frac{4}{10}\right)\cdot 2\right) = 1.52\text{ bit}$$

$$H(G_B) = -\left(\left(\frac{2}{10}\cdot\log\frac{2}{10}\right) + \left(\frac{4}{10}\cdot\log\frac{4}{10}\right)\cdot 2\right) = 1.52\text{ bit}$$

From the discretized expression profiles, the joint probability for each defined *bin* is empirically computed from the observed frequencies for gene A and gene B:

TABLE 5.1 Continuous and Discretized Gene Expression Profile of Gene A and Gene B

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Continuous										
Gene A	3.31	1.03	3.75	3.52	2.25	3.31	3.66	4.10	1.27	4.34
Gene B	1.02	1.05	2.26	2.72	0.37	3.08	2.32	1.02	0.21	2.98
Discrete										
Gene A	3	1	3	3	2	3	3	3	1	3
Gene B	1	1	2	2	1	3	2	1	1	3

The expression values are grouped in a total of 3 bins (*global equal width*).

TABLE 5.2 Empirical Joint Probability Matrix is Obtained from the Joint Count Frequencies of the Two Expression Profiles for Each Bin

Bins	B1	B2	B3
B1	2/10	1/10	1/10
B2	1/10	0	1/10
B3	1/10	1/10	2/10

using Equation 5.4 the joint entropy is then computed by

$$H(G_A, G_B) = -\left(\left(\frac{1}{10} \cdot \log \frac{1}{10}\right) \cdot 6 + \left(\frac{2}{10} \cdot \log \frac{2}{10}\right) \cdot 2\right) = 2.92 \text{ bit}$$

Hence, we obtain the following Mutual information:

$$I(G_A, G_B) = H(G_A) + H(G_B) - H(G_A, G_B) = 0.12 \text{ bit}$$

We describe four different strategies for estimating mutual information for a discretized model. The simplest estimator is the empirical estimator that assumes normality for the values that fall in each bin. However, the main problem with this approach is that the values are often nonuniform distributed among the bins that lead to a skewed estimate of mutual information. A variety of approaches were developed to account for the induced bias that range from correcting the estimate by a constant factor or using a multivariate distribution to model the extend of missing information. In the following, we show four different mutual information estimators that are based on a discretized model. Note that the formulas for the shown estimators are for a single random variable.

The empirical MI estimator gives the maximum likelihood estimate and is based on the observed count frequencies for the values in a bin as shown in the example (see Table 5.2), where the entropy is estimated from the empirical probability distribution with n_k being the number of samples in bin k and N is the total number of samples.

$$H_{\text{emp}} = -\left(\sum_{k=1}^{B} \frac{n_k}{N} \cdot \log \frac{n_k}{N}\right)$$

The empirical estimator gives an underestimate of entropy due to the undersampling of bins when the number of bins B is large. The Miller–Madow estimator [21] accounts for this bias by adjusting the MI estimate by a constant factor proportional to the number of bins and samples.

$$H_{\text{mm}} = H_{\text{emp}} + \frac{B - 1}{2 \cdot N}$$

Here B is the number of bins and N the number of samples.

The shrinkage estimator [22] combines two models for defining cell probabilities, one model with a cell probability of $\frac{1}{B}$ and the empirical model with a cell probability $\frac{n_k}{N}$.

$$\hat{p}_\lambda(n_k) = \lambda\frac{1}{B} + (1-\lambda)\frac{n_k}{N}$$

The λ parameter is estimated by minimizing the mean squared error for the two models for each k of B bins.

$$\lambda^* = \text{argmin}_{\lambda\in[0,1]} E\left[\sum_{k\in B}(p_\lambda(n_k) - p(n_k))^2\right]$$

The entropy for the shrink estimator is computed by

$$\hat{H}^{\text{shrink}} = -\sum_{k=1}^{p} \hat{p}_\lambda(n_k)\log\hat{p}_\lambda(n_k)$$

The Schürmann–Grassberger [23] uses a Dirichlet probability distribution as conjugate prior. The Dirichlet distribution is a multinomial distribution with mean values θ_k.

$$f(\chi;\theta) = \frac{\prod_{k\in\chi}\Gamma(\theta_k)}{\Gamma(\sum_{k\in\chi}\theta_k)}\prod_{k\in\chi}x_k^{\theta_{k-1}}$$

The average θ_k are computed from the posterior using the maximum-likelihood function of the empirical estimator and the Dirichlet prior. The posterior with Schürmann–Grassberger parameters $\frac{1}{B}$ equals

$$\hat{\theta}_k = \frac{n_k + \frac{1}{B}}{N+1}$$

Note that one pseudocount is added in overall to all bins $(N+1)$. The entropy is computed by

$$\hat{H}^{\text{dir}} = -\sum_{k=1}^{p}\hat{\theta}_k\log\hat{\theta}_k$$

5.2.3 Global Measures

In the following, we describe global error measures to evaluate the performance of the inferred network. The most widely used statistical measures are obtained by comparison of an inferred (predicted) network with the true network underlying the data. From the measures listed below, three pairwise combinations thereof are frequently used to assess the performance of network inference algorithms. In the following, we show quantities that are estimated from relations between the number of true positives, true negatives, false positives, and false negatives.

The recall, also known as sensitivity, denotes the proportion of true positive inferred edges relative to all edges in the reference network.

$$\text{Recall (sensitivity)} \quad R = \frac{\text{TP}}{\text{TP+FN}} \tag{5.5}$$

The precision gives the proportion of correctly inferred edges relative to all inferred edges.

$$\text{Precision} \quad P = \frac{\text{TP}}{\text{TP+FP}} \tag{5.6}$$

The specificity is a measure for the relative proportion of edges which were correctly rejected.

$$\text{Specificity} \quad S = \frac{\text{TN}}{\text{TN+FP}} \tag{5.7}$$

The complementary sensitivity is a measure for the proportion of falsely inferred edges.

$$\text{Complementary sensitivity} \quad \text{Rc} = 1 - \text{sensitivity} = \frac{\text{FP}}{\text{TN+FP}} \tag{5.8}$$

The accuracy measures the overall proportion of true edges of the reference network to all falsely inferred and true edges.

$$\text{Accuracy} \quad A = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{5.9}$$

Mutual information-based network inference algorithms obtain a rank for each predicted edge denoted by a rank statistic or the MI estimate as for example used in C3NET. The inferred network is obtained by a defined threshold $\theta \in \Theta$ for the rank statistic of the edges. When the true network is known the threshold that gives the best inference performance can be defined by quantifying the global proportions of true and falsely predicted edges. The first global measure we describe is the area under the curve for the *receiver operator characteristics* (AUC-ROC) [24]. The ROC curve represents the sensitivity (true positive rate, TPR) as function of the complementary sensitivity (false positive rate, FPR) obtained by using various threshold values $\theta \in \Theta$. For each threshold θ, a confusion matrix is obtained that contains the number of true positive, false positives, true negatives, and false negative predictions. The ROC curve of a good classifier is shifted to the upper left side above the main diagonal that shows higher true positive than false positive predictions.

The second measure is the AUC-PR (area under the precision-recall curve) that is obtained similarly as AUC-ROC. The PR curve represents the precision (predicted true positives) as function of the sensitivity (recall, true positive rate). In contrast to the AUC-ROC curve the AUC-PR shows for a good classifier a convex curve shifted to the upper right side showing increase of precision and recall performance. The

AUC-ROC area under the curve value is computed by a numerical integration along each point of the curve.

When the rate of positive and false positives or precision and recall are equal the classifier predicts as good as a random guess. This corresponds to a diagonal line along the ROC or PR space. An AUC-ROC or AUC-PR value of 0.5 corresponds to a classifier performing as good as a random guess.

The third measure is the F-score that describes a weighted average of the precision and recall.

$$F = 2\frac{PR}{P+R}, \tag{5.10}$$

also called F_1 because it is a special form of

$$F_\beta = (1-\beta^2)\frac{PR}{\beta^2(P+R)}. \tag{5.11}$$

The performance measure is obtained by the maximal F-score. This leads to a θ-dependence of all quantities listed above and, hence, allows to obtain a functional behavior among these measures.

We want to emphasize that all measures presented above are general statistical measures used in statistics and data analysis. None of them is specific to our problem under consideration, namely, the inference of networks. Further, all of these measures can only be applied to data for which the underlying network structure is known, because the true network is needed for calculating the above measures.

5.2.4 Ensemble Data and Local Network-Based Measures

In contrast to the above measures, which were global measures, we present now local network-based measures introduced recently [25,26]. These *local network-based* measures are based on ensemble data and the availability of a reference network G that represents the "true" regulatory network. Ensemble data means that there is more than one dataset available from the biological phenomenon under investigation. This ensemble could be obtained either by bootstrapping from one large dataset, from a simulation study or by conducting multiple experiments.

After the ensemble of data $\mathcal{D} = \{D_1(G), \ldots, D_b(G)\}$ is obtained, application of an inference algorithm results in an ensemble of estimated networks $\mathcal{G}^e = \{G_i\}_{i=1}^b$ (Fig. 5.3). Here, we emphasize that each dataset depends on the underlying network structure G that governs the coupling among the genes by writing, for example, $D_i(G)$. Further, this indicates that always the same network G is used. From the ensemble of estimated networks $\mathcal{G}^e = \{G_i\}_{i=1}^b$ one obtains one probabilistic network G_P. This network is a weighted network and the weight of each edge is defined by

$$w_{ij} = \frac{\text{number of times edge } e_{ij} \text{ is present in } \mathcal{G}^e}{b}. \tag{5.12}$$

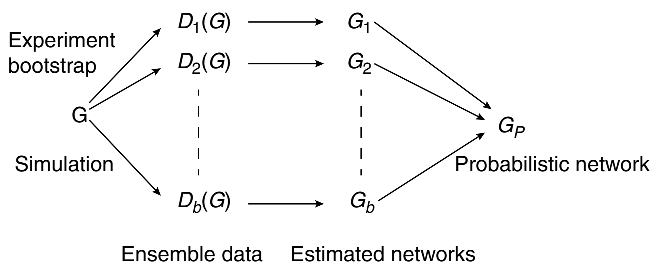


FIGURE 5.3 Schematic visualization of ensemble data of which networks are inferred and subsequently aggregated to estimate a probabilistic network.

It is easy to see that w_{ij} corresponds to the probability that edge e_{ij} is present in $\mathcal{G}^e$,

$$w_{ij} = Prob(\text{gene } i \text{ is connected with gene } j \text{ in } \mathcal{G}^e). \tag{5.13}$$

The aggregation of an ensemble of networks to obtain the probabilistic network $\mathcal{G}^e$ is visualized in Figure 5.4.

If the network structure of the underlying network G is available it is possible to obtain estimates of the TPR and TNR of edges and nonedges in G. For example, if there is an edge between gene i and j in G we obtain,

$$\text{TPR}_{ij} = \frac{\text{number of times edge } e_{ij} \text{ is present in } \mathcal{G}^e}{b}. \tag{5.14}$$

If there is no edge between gene i and j in G we obtain instead,

$$\text{TNR}_{ij} = \frac{\text{number of times edge } e_{ij} \text{ is not present in } \mathcal{G}^e}{b}. \tag{5.15}$$

This is visualized in Figure 5.5. It is important to realize that the network G is here used to *classify* edges/nonedges to different edge sets. Such a classification is not possible

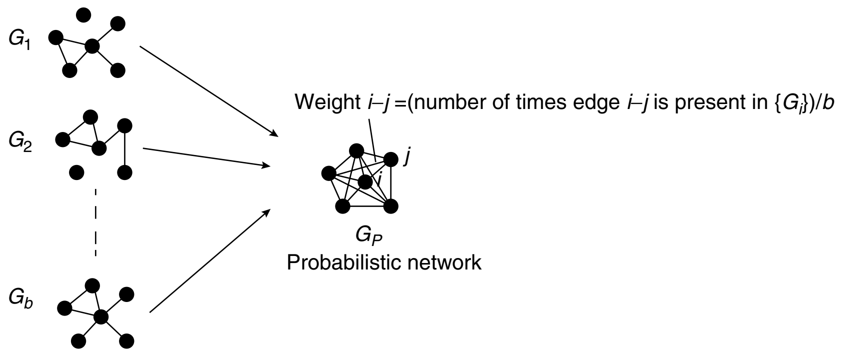
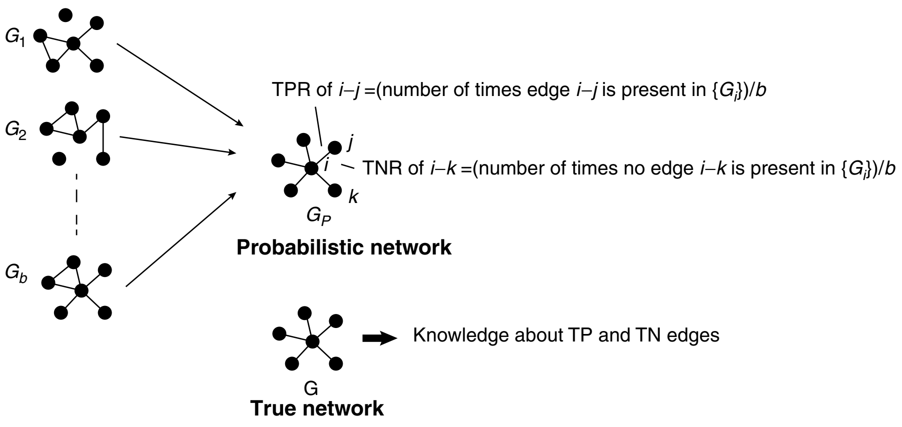

G_1
G_2
G_b
Weight i–j =(number of times edge i–j is present in $\{G_i\}$)/b
j
i
G_P
Probabilistic network

FIGURE 5.4 The ensemble of networks $\{G_i\}$ is used to obtain a weighted network G_P.



G_1
G_2
G_b
TPR of i–j =(number of times edge i–j is present in $\{G_i\}$)/b
j
i
TNR of i–k =(number of times no edge i–k is present in $\{G_i\}$)/b
k
G_P
Probabilistic network
Knowledge about TP and TN edges
G
True network

FIGURE 5.5 If the true network G is used in addition to the ensemble of networks $\{G_i\}$ one obtains estimates for the TPR and TNR of all edges.

without G. From Equations 5.14 and 5.15, follow the corresponding negative rates by

$$\text{FNR}_{ij} = 1 - \text{TPR}_{ij} \tag{5.16}$$

$$\text{FPR}_{ij} = 1 - \text{TNR}_{ij} \tag{5.17}$$

and, hence, all statistical measures described in Section 5.2.3. However, it should be emphasized that now, not the global inference performance with respect to the entire network is characterized but of individual edges. To make this clear, we are calling such measures *local network-based measures* [25,26]. The Equations 5.14–5.17 provide the finest resolution obtainable because an edge is the most basic unit in a network.

A practical consequence of the information provided by $\{\text{TPR}_{ij}\}$ and $\{\text{TNR}_{ij}\}$ is that it is possible to construct further local network-based measures representing larger subparts of a network. For example, one can construct measures to characterize the inferrability of network motifs consisting of n genes or the influence of the degree of genes. Principally, any combination of $\{\text{TPR}_{ij}\}$ and $\{\text{TNR}_{ij}\}$ values would result in a valid (statistical) measure, for example, of network motifs. Further examples of such measures can be found in Refs. [25,26].

In order to make the basic principle behind such a construction of local network-based measures more clear, we present as an example motifs consisting of three genes. Recently, it has been recognized that network motifs are important building blocks of various complex networks, including gene networks [27–30]. For this reason, biologically, it is of interest to study their inferrability. Formally, we define the true reconstruction rate of a motif by

$$p = \frac{1}{3}\sum_{1}^{3} \text{TPR}_i. \tag{5.18}$$

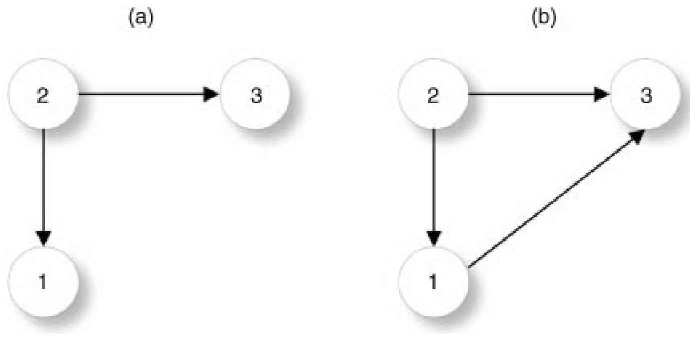


FIGURE 5.6 Two examples of three-gene network motifs.

Here, TPR corresponds either to a TPR if two genes are connected or to a TNR if these genes are unconnected and the factor results from the fact that we consider three-gene motifs only, however, extensions thereof are straightforward. Figure 5.6 illustrates two examples of three-gene motifs. From Equation 5.18, we obtain for these two motifs

$$p[\text{motif type} = A] = \frac{1}{3}\big(\text{TPR}(1 \leftarrow 2) + \text{TPR}(2 \rightarrow 3) + \text{TNR}(1 \not\leftrightarrow 3)\big), \quad (5.19)$$

$$p[\text{motif type} = B] = \frac{1}{3}\big(\text{TPR}(1 \rightarrow 2) + \text{TPR}(2 \rightarrow 3) + \text{TPR}(1 \rightarrow 3)\big). \quad (5.20)$$

Each of these measures represents the inferrability of a certain motif type. Averaging over all motifs of the same type found in network leads to the mean true reconstruction rate $\overline{p[\text{motif type}]}$ of a certain motif type.

5.2.5 Local Network-Based Measures

The most frequently studied local structural properties of biological networks are, for example, network motifs, node degree classes, and community structures. In this section, we will address the influence of different mutual information estimators on local network-based measures. The local measure is based on the estimated true positive rate (TPR) of the inferred edges from the ensemble. We demonstrate the structural influence on the true positive rate (TPR) using binary classification of the edges according to the degree of the nodes enclosing an edge. From the ensemble the TPR weights for the edges are compared between two classes defined from the degree of their nodes. We introduce a local measure for directed and undirected networks that are defined by the measure D^1 and D^2.

The measure D^1 is defined for a directed graph as the sum of the out-degree of node i plus the in-degree of node j:

$$D^1_{ij} = \deg(v_i)^{\text{in}} + \deg(v_j)^{\text{out}} \quad (5.21)$$

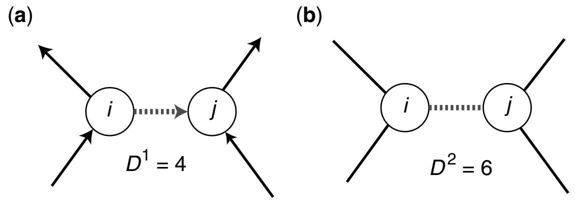


FIGURE 5.7 An edge (dashed) is scored according to the degree of the adjacent nodes for (a) directed network (sum of out degree *i* and in degree *j*) and (b) undirected network (sum of degree *i* and degree *j*).

A binary classification of the edges is obtained by assigning edges to two classes of edges, depending on the value of D^1. Specifically, for D^1 the classes are defined by

- *Class I*: Edges with $D^1 \leq 3$ (corresponds to a chain-like structure)
- *Class II*: All other edges

The measure D^2 is defined for an undirected graph by the sum of the degrees of node i plus the degrees of node j:

$$D_{ij}^2 = \deg(v_i) + \deg(v_j) \tag{5.22}$$

As shown above for D^1, a binary classification of the edges is obtained by assigning edges to two classes of edges with a high graph density score and a low graph density score. For D^2 the classes are defined by

- *Class I*: Edges with $D^2 \leq 4$ (corresponds to a chain-like structure)
- *Class II*: All other edges

An example is shown in Figure 5.7, for the directed measure D^1 and the undirected measure D^2. We measure the influence of mutual information estimators on the two defined edge classes. The ensemble of networks is inferred from bootstrap expression datasets to estimate the TPR (see Section 5.2.4). From the ensemble, we obtain the distribution of the mean TPR for the two classes of both measures in D^1 and D^2 that are compared.

5.2.6 Network Structure

The structure of networks can be described by homogeneous networks and inhomogeneous networks [31]. Homogeneous networks belong to the class of exponential networks such as the Erdös and Rènyi [32], [33] and Watts and Strogatz graph model (Small-World) [34], where all nodes in the network have approximately the same number of edges [31]. Inhomogeneous networks belong to the class of scale-free networks in which the number of edges for the nodes follow a power–law distribution

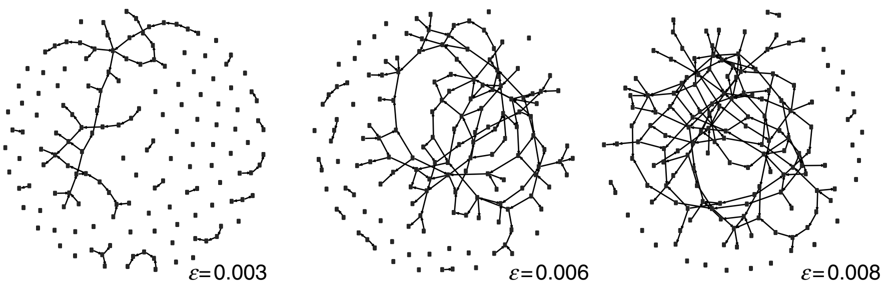


FIGURE 5.8 Three Erdös–Rènyi networks with edge density $\epsilon = \{0.003, 0.006, 0.008\}$.

[35]. This property allows the network to contain highly connected nodes such as network hubs [31].

We study the influence of the network structure on the C3NET inference using random networks with different values of p. Here, p is the probability for the presence of an edge between the two nodes. Because real gene networks, for example, the transcriptional regulatory network or the protein network, are sparse the value of p needs to be chosen to fall within a realistic interval. Typically, gene networks are sparse with an edge density of about $\sim 10^{-3}$ [36].

For our study, we are using networks with $n = 150$ genes resulting in a maximal number of $E = 22,350$ (directed) edges. We generated three Erdös–Rènyi graphs with edge density $\epsilon = \{0.003, 0.006, 0.008\}$ with $\{22, 19, 10\}$ unconnected nodes. The networks are shown in Figure 5.8. For the number of edges E_ϵ the density of a graph is defined by $\epsilon = E_\epsilon/E$ with the number of edges $E_\epsilon = \{77, 145, 170\}$.

A broad collection of procedures for *in silico* gene expression simulation are available, for example, GeneNetWeaver [37], NetSim [38], SYNTREN [39] for time-course and steady-state experiments that are used to validate the network inference accuracy. For our study, we generate simulated steady-state gene expression datasets. Gene expression datasets were simulated using SYNTREN [39] with the *bionoise* parameter set to 0.05. In SYNTREN, gene expression is modeled using nonlinear Michaelis–Menten kinetics with a superposed function that models biological noise not related to the experiment.

5.3 RESULTS

We compare the inference performance of the C3NET [12] for four mutual information estimators. We simulated gene expression datasets for random graphs for different edge densities. We studied the global inference performance and structural characteristics of the inferred networks.

5.3.1 Global Network Inference Performance

We study the influence of four different mutual information estimators on the C3NET network inference performance. We use the F-score measure to measure the

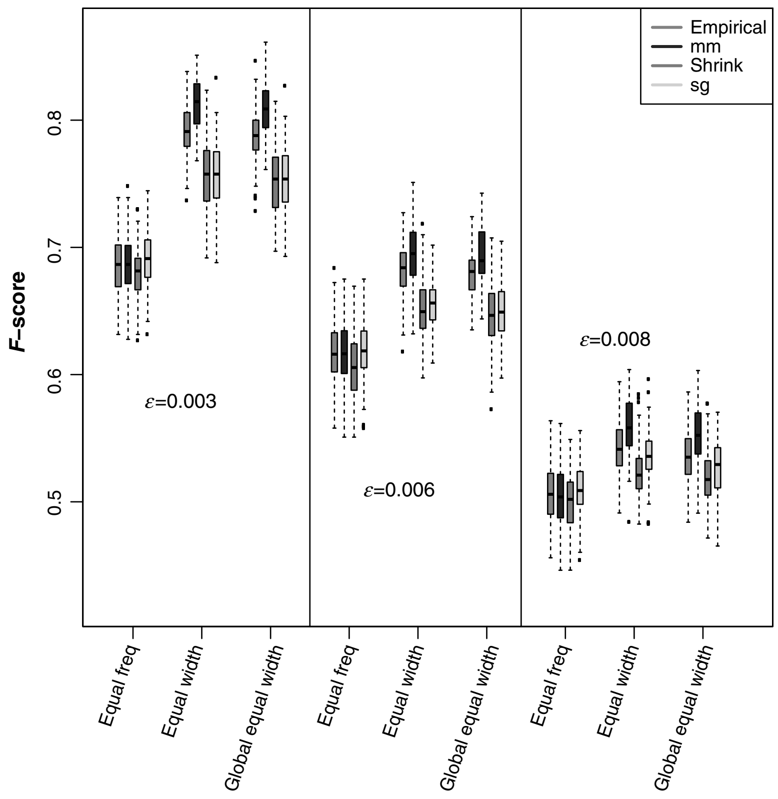


FIGURE 5.9 The influence of different discretization methods on the global network inference accuracy F-score measure for three Erdös–Rènyi networks using four MI estimators. The simulated gene expression datasets have a sample size of 200.

performance for C3NET network inference from simulated gene expression data. First, we compared the impact of three discretization methods for each estimator. For the three network types the equal width and global equal width discretization showed the highest inference accuracy for C3NET compared to the equal frequency discretization.

The *equal* and *global equal width* discretization favor the Miller–Madow estimator followed by the empirical estimator to be most beneficial for the C3NET inference performance. The Schürmann–Grassberger and Shrink estimator perform worse. However, the Schürmann–Grassberger performs better than the Shrink estimator (Fig. 5.9). For the *equal frequency* discretization, we do not observe a substantial difference of the inference performance for the empirical, Miller–Madow, Shrink, and Schürmann–Grassberger estimator (Fig. 5.9).

Further, we studied the influence of the estimators for each of the discretization methods on the sample size (*equal frequency* discretization Fig. 5.10, *equal width* discretization Fig. 5.11, *global equal width* discretization Fig. 5.12). For all network

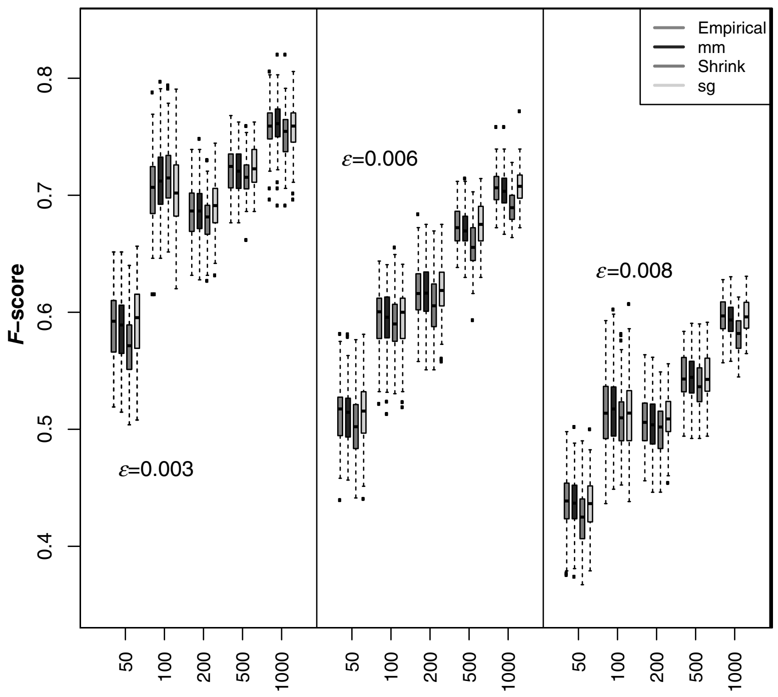


FIGURE 5.10 The influence of *equal frequency* discretization method on the global network inference accuracy F-score measure for three Erdös–Rènyi networks using four MI estimators.

types, the inference performance increases with the sample size. The Miller-Madow estimator in combination with the *equal width* and *global equal width* discretization is the most beneficial setting for the inference performance of C3NET.

The C3NET infers a sparse network with maximal one edge for each gene. Due to the limited number of edges in an inferred C3NET network, more densely connected network structure cannot be inferred. This effect can be observed in Figures 5.9–5.12, where the performance for the network inference is decreasing with an increasing edge density.

5.3.2 Local Network Inference Performance

In the previous section, we studied the impact of MI estimators on the global network inference performance of C3NET. When global measures are used, we measure the average influence of the MI estimators on the inference performance for all edges. *Local network-based* measures allow to study the local network inference performance for different edge classes. We use the D^1 measure to classify edges in the three Erdös–Rènyi reference networks into two edge classes connected to nodes with a low (Class I) and a high (Class II) edge degree. For each dataset, we obtain a measure for the true positive rate from the Bootstrap generated ensemble of networks. We compare the distribution of median true positive rates for D^1 Class I and D^1

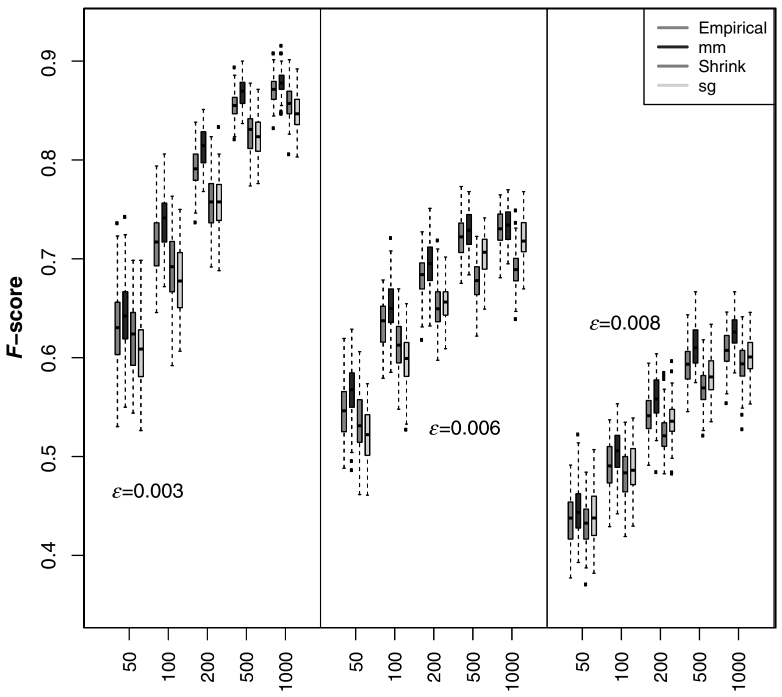


FIGURE 5.11 The influence of *equal width* discretization method on the global network inference accuracy F-score measure for three Erdös–Rènyi networks using four MI estimators.

Class II for 100 datasets for each network. In Figures 5.13 and 5.14, we show the distribution of the median true positive rate for the three simulated gene expression datasets and different sample sizes for the two classes in D^1. The Class I edges show a high inference performance for the D^1 measure, while Class II edges have a low inference performance. With increasing edge density and increasing sample size the true positive rate for Class I edges approaches 1 while for Class II the true positive rate approaches 0.

For Class I edges, the true positive rates among MI estimators do not show a substantial difference, while the Schürmann–Grassberger estimator has the tendency to perform worse than the other MI estimators. For Class II edges, the Miller–Madow estimators shows the best inference performance. We performed the same analysis using the D^2 measure (not shown). The D^1 and D^2 measure show similar results for the relative performance of the four MI estimators.

5.4 CONCLUSION AND SUMMARY

Our study shows that the choice of the discretization method and MI estimator has a crucial influence on the inference performance of C3NET. In detail, the *equal width*

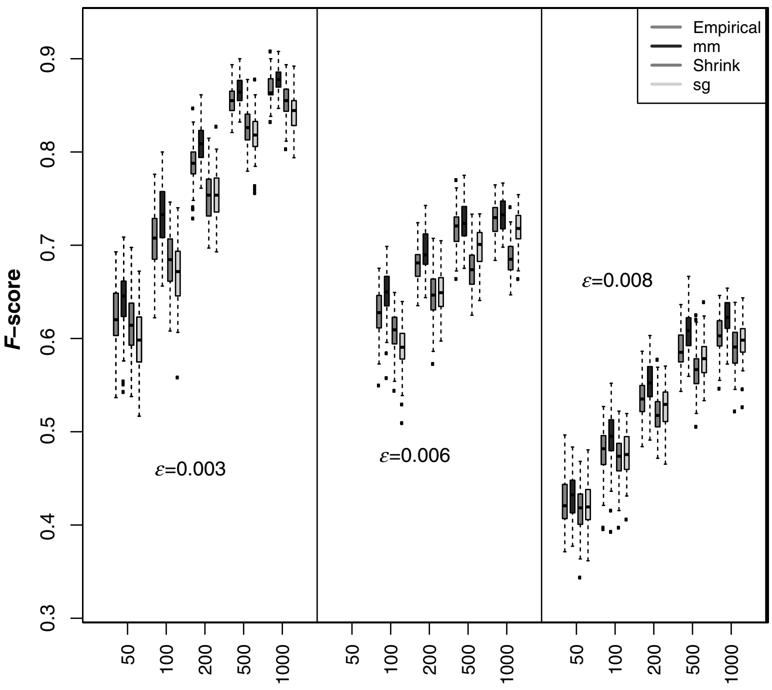


FIGURE 5.12 The influence of *global equal width* discretization method on the F-score for three Erdös–Rènyi networks using four MI estimators.

and *global equal width* showed the best performance in combination with the Miller–Madow estimator. However, the major influence on the C3NET inference performance was observed for the discretization methods, where *equal width* and *global equal width* discretization markedly outperforms the *equal frequency* discretization.

In the study conducted by Olsen et al. [16], the influence of discretization, the mutual information estimator, sample size, and network size was studied for the ARACNE, CLR, and MRNET GRN inference algorithms. In contrast to our results, the equal frequency discretization was observed to outperform the equal width discretization for the used inference algorithms. In addition, the discrete estimators did not show a large difference as seen in our study, for example, for Miller–Madow.

The results suggest that the influence of the MI estimator on the global inference performance is highly dependent on the inference algorithm used. It is, therefore, a prerequisite to test GRN inference algorithms individually for different discretization and mutual information estimators.

Global error measures quantify the average inference performance for all edges in a network. Local measures allow to zoom-in the inference performance of individual parts of the network, down to individual edges. For C3NET, edges of leaf nodes and edges of linearly connected nodes of a network are inferred with higher performance

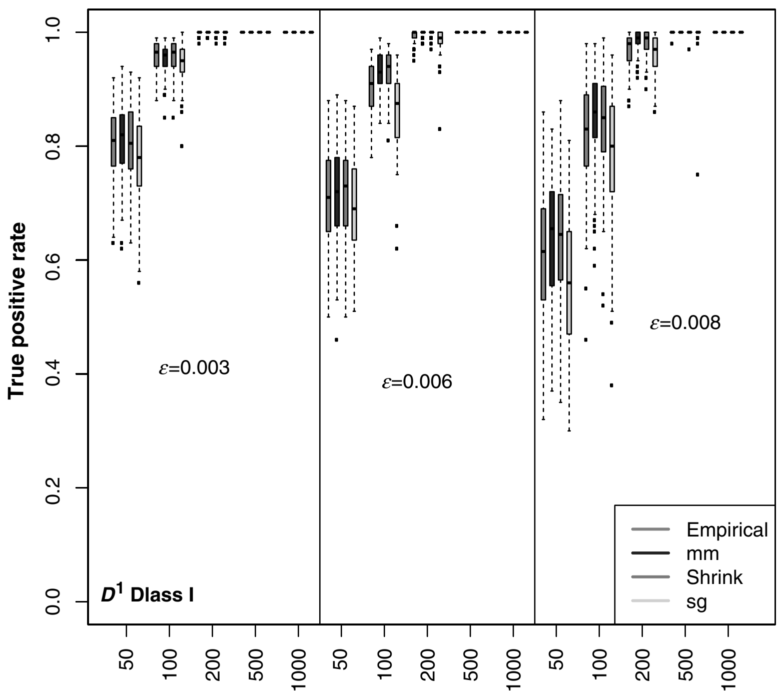


FIGURE 5.13 Class I edges (according to D^1) in the three Erdös–Rènyi networks using four different MI estimators. Simulated gene expression datasets for Erdös–Rènyi networks with edge density ϵ for sample sizes ranging from 50 to 1000 samples.

[12]. Edges from nodes with a high degree are likely underrepresented as they are more difficult to infer.

We studied the influence of the MI estimators on the performance for different edge classes that were classified by *local network-based measures*. Two edge classes were defined for edges of leaf and linearly connected nodes, and highly connected nodes in the network. The inference ability of C3NET for an edge was quantified by the true positive rate measured from an ensemble of networks inferred from Bootstrap datasets. From the set of datasets for a network the distribution of median true positive rate is obtained for Class I and Class II edges. We compared the resulting distributions of true positive rates between edges of Class I and Class II. As expected, we observed high true positive rates for Class I edges, while Class II edges show low true positive rates. As the true positive rate for Class I edges was very high the differences among the MI estimators were not so apparent as for Class II edges. For Class II edges, the Miller–Madow estimator resulted in the best inference performance for C3NET, as for the global error measure.

In this chapter, we presented a simulation study to analyze the impact of MI estimators on the inference performance of C3NET. The inference performance was studied using global and local network-based measures for simulated gene expression

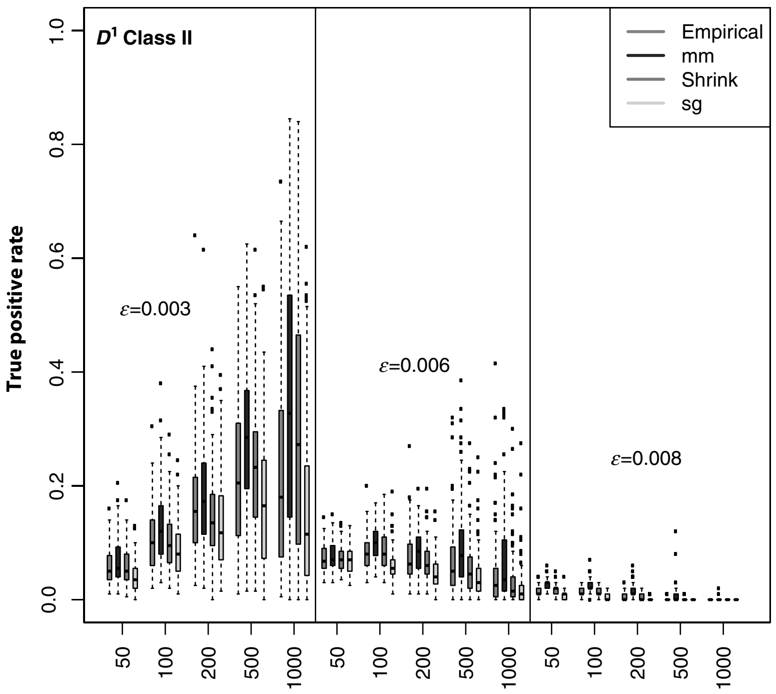


FIGURE 5.14 Class II edges (according to D^1) in the three Erdös–Rènyi networks using four different MI estimators. Simulated gene expression datasets for Erdös–Rènyi networks with edge density ϵ for sample sizes ranging from 50 to 1000 samples.

data from Erdös–Rènyi networks with different edge densities and varying sample sizes. Among the tested combinations of discretization methods and MI estimators we recommend the use of the Miller–Madow estimator with *equal width* or *global equal width* discretization for C3NET network inference.

ACKNOWLEDGMENT

We would like to thank Shailesh Tripathi for fruitful discussions. For our simulations we used R [40], minet [41], and igraph [42].

REFERENCES

1. F. Emmert-Streib, M. Dehmer. Networks for systems biology: conceptual connection of data and function. *IET Syst. Biol.* **5**, 185–207 (2011).
2. F. Emmert-Streib, G.V. Glazko. Network biology: a direct approach to study biological function. *Wiley Interdiscip. Rev. Syst. Biol. Med.* **3**, 379-3-91 (2011).

3. G. Stolovitzky, A. Califano (ed.) *Reverse Engineering Biological Networks: Opportunities and Challenges in Computational Methods for Pathway Inference*. Wiley-Blackwell, 2007.
4. G. Stolovitzky, R.J. Prill, A. Califano. Lessons from the DREAM 2 Challenges. *Ann. N. Y. Acad. Sci.* **1158**, 159–195 (2009).
5. M. Schena (ed.) *Protein Microarrays*. Jones and Bartlett Publishers, 2004.
6. S. Sechi, (ed.) *Quantitative Proteomics by Mass Spectrometry*. Humana Press, 2007.
7. P. Collas, J.A. Dahl. Chop it, ChIP it, check it: the current status of chromatin immunoprecipitation. *Front Biosci.* **13**, 929–943 (2008).
8. R. Edgar, M. Domrachev, A.E. Lash. Gene expression omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Res.* **30**, 207–210 (2002).
9. P.R. Rosenbaum. *Observational Studies*, Springer, New York, 2002.
10. R. Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *J. Educ. Psychol.* **66**, 688–701 (1974).
11. 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).
12. G. Altay, F. Emmert-Streib. Inferring the conservative causal core of gene regulatory networks. *BMC Syst. Biol.* **4**, 132 (2010).
13. A.L. Barabasi, Z.N. Oltvai. Network biology: understanding the cell's functional organization. *Nat. Rev.* **5**, 101–113 (2004).
14. E.E. Schadt. Molecular networks as sensors and drivers of common human diseases. *Nature* **461**, 218–223 (2009).
15. A.J. Butte, I.S. Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. *Pac. Sym. Biocompu.* **5**, 418–429 (2000).
16. C. Olsen, P.E. Meyer, G. Bontempi. On the impact of entropy estimation on transcriptional regulatory network inference based on mutual information. *EURASIP J. Bioinform. Syst. Biol.* **308959**, 2009.
17. A.J. Butte, I.S. Kohane. Mutual information relevance networks: Functional genomic clustering using pairwise entropy measurements. *Pac. Sym. on Biocomput.* **5**, 415–26 (2000).
18. A.A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky et al. ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinform.* **7**, S7 (2006).
19. T.M. Cover, J. Thomas. *Elements of Information Theory*. Wiley, 1991.
20. Y. Yang, G.I. Webb. Proportional k-interval discretization for naive-bayes classifiers. *Proceeding EMCL '01 Proceedings of the 12th European Conference on Machine Learning*, 2001.
21. L. Paninski. Estimation of entropy and mutual information. *Neural Comput.* **15**, 1191–1253 (2003).
22. J. Schafer, K. Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Stat. Appl. Genet. Mol. Biol.* **4** (2005).
23. T. Schürmann, P. Grassberger. Entropy estimation of symbol sequences. *Chaos* **6**, 414–427 (1996).
24. T. Fawcett. An introduction to ROC analysis. *Pattern Recogn. Lett.* **27**, 861–874 (2006).
25. F. Emmert-Streib, G. Altay. Local network-based measures to assess the inferability of different regulatory networks. *IET Syst. Biol.* **4**, 277–88 (2010).

26. G. Altay, F. Emmert-Streib. Revealing differences in gene network inference algorithms on the network level by ensemble methods. *Bioinformatics* 26, 1738–1744 (2010).
27. U. Alon. *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Chapman & Hall/CRC, Boca Raton, FL, 2006.
28. Y. Artzy-Randrup, S.J. Fleishman, N. Ben-Tal, L. Stone. Comment on "Network Motifs: Simple Building Blocks of Complex Networks" and "Superfamilies of Evolved and Designed Networks". *Science* **305**(5687), 1107c (2004).
29. G. Ciriello, C. Guerra. A review on models and algorithms for motif discovery in protein–protein interaction networks. *Brief Funct. Genomic Proteomic* **7**(2), 147–156 (2008).
30. R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, U. Alon. Network motifs: simple building blocks of complex networks. *Science* **298**(5594), 824–827 (2002).
31. R. Albert, H. Jeong, A.L. Barabasi. Error and attack tolerance of complex networks. *Nature* **406**, 378–382 (2000).
32. P. Erdös, A. Rènyi. On random graphs. *Pub. Math.* **6**, 290–297 (1959).
33. B. Bollobas. *Random Graphs*. Cambridge University Press, 2001.
34. D.J. Watts, S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature* **393**, 440–442 (1998).
35. A.L. Barabasi, R. Albert. Emergence of scaling in random networks. *Science* **286**, 509–512 (1999).
36. R.D. Leclerc. Survival of the sparsest: robust gene networks are parsimonious. *Mol. Syst. Biol.* **4**, 213 (2008).
37. T. Schaffter, D. Marbach, D. Floreano. GeneNetWeaver: *in silico* benchmark generation and performance profiling of network inference methods. *Bioinformatics* **27**(16), 2263–2270 (2011).
38. B. Di Camillo, G. Toffolo, C. Cobelli. A gene network simulator to assess reverse engineering algorithms. *Ann. N. Y. Acad. Sci.* **1158**, 125–142 (2009).
39. T. Van den Bulcke, K. Van Leemput, B. Naudts, P. van Remortel, H. Ma, A. Verschoren, B. De Moor, K. Marchal. SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC Bioinform.* **7**, 43 (2006).
40. R Development Core Team. *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2011.
41. P.E. Meyer, F. Lafitte, G. Bontempi. minet: A R/Bioconductor package for inferring large transcriptional networks using mutual information. *BMC Bioinform.* **9**, 461 (2008).
42. G. Csardi, T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems, 1695, 2006.

6

WEIGHTED SPECTRAL DISTRIBUTION: A METRIC FOR STRUCTURAL ANALYSIS OF NETWORKS

DAMIEN FAY, HAMED HADDADI, ANDREW W. MOORE, RICHARD MORTIER, ANDREW G. THOMASON, AND STEVE UHLIG

6.1 INTRODUCTION

Graph comparison is a problem that occurs in many branches of computing, from vision to speech processing to systems. Many techniques exist for graph comparison, for example, the edit distance [1] (the number of link and node additions or deletions required to turn one graph into another), or counting the number of common substructures in two graphs [2]. Unfortunately, these methods are too computationally expensive for large graphs such as the Internet topologies studied here. Moreover, they are inappropriate for dynamic graphs, resulting in varying edit distances or substructure counts. Currently used common "metrics" include the clustering coefficient, the assortativity coefficient, the node degree distribution, and the k-core decomposition. However, these are not metrics in the mathematical sense, but rather are measures. This distinction is important as *a measure cannot be used to determine unique differences between graphs*: two graphs with the same measures may not in fact be the same. For example, two graphs may have the same clustering coefficient but hugely different structures.

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

In this chapter, we present the *weighted spectral distribution* (WSD), a true metric in the mathematical sense, which compares graphs based on the distribution of a decomposition of their structure. Specifically, the WSD is based on the spectrum of the normalized Laplacian matrix and is thus strongly associated with the distribution of *random walk cycles* in a network. A random walk cycle occurs when we find that we have returned to a node having walked N steps away from it. The probability of a random walk cycle originating at a node indicates the connectivity of that node: a low probability indicates high connectivity (there are many routes, few of which return) while a high probability indicates high clustering (many of the routes lead back to the original node).

The WSD is computationally inexpensive and so can be applied to very large graphs (more than 30,000 nodes and 200,000 edges). Also, it expresses the graph structure as a simple plotted curve that can be related to two specific properties of graphs: hierarchy and local connectivity. Given that the WSD is a metric in the mathematical sense several applications become possible: assessment of synthetically generated topologies based on real measurements, where the generated graphs should share some common structure with the original measurements rather than *exactly* matching them; parameter estimation for topology generators with respect to a target dataset; direct comparison among topology generators using these optimal parameters; and quantification of change in the underlying structure of an evolving topology.

6.2 WEIGHTED SPECTRAL DISTRIBUTION

We now derive our metric, the *weighted spectral distribution*, relating it to another common structural metric, the clustering coefficient, before showing how it characterizes networks with different mixing properties.

Denote an undirected graph as $G = (V, E)$ where V is the set of vertices (nodes) and E is the set of edges (links). The adjacency matrix of G, $A(G)$, has an entry of one if two nodes, u and v, are connected and zero otherwise

$$A(G)(u, v) = \begin{cases} 1, & \text{if } u, v \text{ are connected} \\ 0, & \text{if } u, v \text{ are not connected} \end{cases} \tag{6.1}$$

Let d_v be the degree of node v and $D = \text{diag}(\text{sum}(A))$ be the diagonal matrix having the sum of degrees for each node (column of matrix) along its main diagonal. Denoting by I the identity matrix $(I)_{i,j} = 1$ if $i = j$, 0 otherwise, the normalized Laplacian L associated with graph G is constructed from A by normalizing the entries of A by the node degrees of A as

$$L(G) = I - D^{-1/2} A D^{-1/2} \tag{6.2}$$

or equivalently

$$L(G)(u,v)=\begin{cases}1, & \text{if } u=v \text{ and } d_v\neq 0\\ -\dfrac{1}{\sqrt{d_u d_v}}, & \text{if } u \text{ and } v \text{ are adjacent}\\ 0, & \text{otherwise}\end{cases} \tag{6.3}$$

As L is a real symmetric matrix there is an orthonormal basis of real eigenvectors $e_0,\ldots,e_{n-1}$ (i.e., $e_ie_j^T=0, i\neq j$, and $e_ie_i^T=1$) with associated eigenvalues $\lambda_0,\ldots,\lambda_{n-1}$. It is convenient to label these so that $\lambda_0\leq\ldots\leq\lambda_{n-1}$. The set of pairs (eigenvectors and eigenvalues of L) is called the spectrum of the graph. It can be seen that

$$L(G)=\sum_i \lambda_i e_i e_i^T \tag{6.4}$$

The eigenvalues $\lambda_0,\ldots,\lambda_{n-1}$ represent the strength of projection of the matrix onto the basis elements. This may be viewed from a statistical point of view [3] where each $\lambda_ie_ie_i^T$ may be used to approximate $A(G)$ with approximation error inversely proportional to $1-\lambda_i$. However, for a graph, those nodes which are best approximated by $\lambda_ie_ie_i^T$ in fact form a cluster of nodes. This is the basis for spectral clustering, a technique which uses the eigenvectors of L to perform clustering of a dataset or graph [4]. The first (smallest) nonzero eigenvalue and associated eigenvector are associated with the main clusters of data. Subsequent eigenvalues and eigenvectors can be associated with cluster splitting and also identification of smaller clusters [5]. Typically, there exists what is called a *spectral gap* in which for some k and j, $\lambda_k\ll\lambda_{k+1}\approx 1\approx\lambda_{j-1}\ll\lambda_j$. That is, eigenvalues $\lambda_{k+1},\ldots,\lambda_{j-1}$[1] are approximately equal to one and are likely to represent links in a graph which do not belong to any particular cluster. It is then usual to reduce the dimensionality of the data using an approximation based on the spectral decomposition. However, in this chapter we are interested in representing the global structure of a graph (e.g., we are interested in the presence or absence of many small clusters), which is essentially the spread of clustering across the graph. This information is contained in all the eigenvalues of the spectral decomposition.

Let $x=(x_0,\ldots,x_{n-1})$ be a vector. From Equation 6.3 we see that

$$xLx^T=\sum_{uv\in E}(x_u/\sqrt{d_u}-x_v/\sqrt{d_v})^2 \tag{6.5}$$

[1]That is, the eigenvalues at the center of the spectrum.

Now, the eigenvalues cannot be large because from Equation 6.5 we obtain

$$\begin{aligned} xLx^T &\le \sum_{uv\in E} (x_u/\sqrt{d_u} - x_v/\sqrt{d_v})^2 \\ &\quad + (x_u/\sqrt{d_u} + x_v/\sqrt{d_v})^2 \\ &= 2\sum_u x_u^2 = 2xx^T \end{aligned} \tag{6.6}$$

and so $\lambda_i = e_i L e_i^T \le 2$. What is more, the mean of the eigenvalues is 1 because

$$\sum_i \lambda_i = \mathrm{tr}(L) = n \tag{6.7}$$

by Equation 6.3, where $\mathrm{tr}(L)$ is the *trace* of L.

To summarize, the eigenvalues of L lie in the range 0–2 (the smallest being 0), that is, $0 = \lambda_0 \le \ldots \le \lambda_{n-1} \le 2$, and their mean is 1.

The distribution of the n numbers $\lambda_0, \ldots, \lambda_{n-1}$ contains useful information about the network, as will be seen. In turn, information about this distribution is given by its moments in the statistical sense, where the Nth moment is $1/n \sum_i (1-\lambda_i)^N$. These moments have a direct physical interpretation in terms of the network, as follows. Writing B for the matrix $D^{-1/2}AD^{-1/2}$, so that $L = I - B$, then by Equation 6.3 the entries of B are given by

$$(D^{-1/2}AD^{-1/2})_{i,j} = \frac{A_{i,j}}{\sqrt{d_i}\sqrt{d_j}} \tag{6.8}$$

Now the numbers $1 - \lambda_i$ are the eigenvalues of $B = I - L$, and so $\sum_i (1-\lambda_i)^N$ is just $\mathrm{tr}(B^N)$. Writing $b_{i,j}$ for the (i, j)th entry of B, the (i, j)th entry of B^N is the sum of all products $b_{i_0,i_1} b_{i_1,i_2}, \ldots, b_{i_{N-1} i_N}$ where $i_0 = i$ and $i_N = j$. But $b_{i,j}$, as given by Equation 6.8, is zero unless nodes i and j are adjacent. So we define an N-cycle in G to be a sequence of vertices $u_1 u_2 \ldots u_N$ with u_i adjacent to u_{i+1} for $i = 1, \ldots, N-1$ and with u_N adjacent to u_1. (Thus, for example, a triangle in G with vertices set $\{a, b, c\}$ gives rise to six 3-cycles *abc*, *acb*, *bca*, *bac*, *cab*, and *cba*. Note that, in general, an N-cycle might have repeated vertices.) We now have

$$\sum_i (1-\lambda_i)^N = \mathrm{tr}(B^N) = \sum_C \frac{1}{d_{u_1} d_{u_2} \ldots d_{u_N}} \tag{6.9}$$

the sum being over all N-cycles $C = u_1 u_2 \ldots u_N$ in G. Therefore, $\sum_i (1-\lambda_i)^N$ counts the number of N-cycles, normalized by the degree of each node in the cycle.

The number of N-cycles is related to various graph properties. The number of 2-cycles is just (twice) the number of edges and the number of 3-cycles is (six times) the number of triangles. Hence, $\sum_i (1-\lambda)^3$ is related to the clustering coefficient, as discussed below. An important graph property is the number of 4-cycles. A graph which has the minimum number of 4-cycles, for a graph of its density, is quasirandom, that is, it shares many of the properties of random graphs, including, typically, high connectivity, low diameter, having edges distributed uniformly through the graph, and so on. This statement is made precise in Refs. [6] and [7]. For regular graphs Equation (6.7) shows that the sum $\sum_i (1 - \lambda)^4$ is directly related to the number of 4-cycles. In general, the sum counts the 4-cycles with weights, for the relationship between the sum and the quasirandomness of the graph in the nonregular case, see the more detailed discussion in Ref. [8, Chapter 5]. The right-hand side of Equation 6.9 can also be seen in terms of random walks. A random walk starting at a vertex with degree d_u will choose an edge with probability $1/d_u$ and at the next vertex, say v, choose an edge with probability $1/d_v$, and so on. Thus, the probability of starting and ending randomly at a vertex after N steps is the sum of the probabilities of all N-cycles that start and end at that vertex. In other words exactly the right-hand side of Equation 6.9. As pointed out in Ref. [9], random walks are an integral part of the Internet AS structure.

The left-hand side of Equation 6.9 provides an interesting insight into graph structure. The right-hand side is the sum of normalized N-cycles whereas the left-hand side involves the spectral decomposition. We note in particular that the spectral gap is diminished because eigenvalues close to one are given a very low weighting compared to eigenvalues far from one. This is important as the eigenvalues in the spectral gap typically represent links in the network that do not belong to any specific cluster and are not therefore important parts of the larger structure of the network.

Next, we consider the well-known clustering coefficient. It should be noted that there is little connection between the clustering coefficient, and cluster identification, referred to above. The clustering coefficient, $\gamma(G)$, is defined as the average number of triangles divided by the total number of possible triangles

$$\gamma(G) = 1/n \sum_i \frac{T_i}{d_i(d_i - 1)/2}, \quad d_i \geq 2 \tag{6.10}$$

where T_i is the number of triangles for node i and d_i is the degree of node i. Now consider a specific triangle between nodes a, b, and c. For the clustering coefficient, noting that the triangle will be considered three times, once from each node, the contribution to the average is

$$\frac{1}{d_a(d_a - 1)/2} + \frac{1}{d_b(d_b - 1)/2} + \frac{1}{d_c(d_c - 1)/2} \tag{6.11}$$

However, for the weighted spectrum (with $N = 3$), this particular triangle gives rise to six 3-cycles and contributes

$$\frac{6}{d_a d_b d_c} \tag{6.12}$$

So, it can be seen that the clustering coefficient normalizes each triangle according to the total number of possible triangles while the weighted spectrum (with $N = 3$) instead normalizes using a product of the degrees. Thus, the two metrics can be considered to be similar but not equal. Indeed, it should be noted that the clustering coefficient is in fact not a metric in the strict sense. While two networks can have the same clustering coefficient they may differ significantly in structure. In contrast, the elements of $\sum_i (1-\lambda)^3$ will only agree if two networks are isomorphic.

We now formally define the *weighted spectrum* as the normalized sum of N-cycles as

$$W(G, N) = \sum_i (1-\lambda_i)^N \tag{6.13}$$

However, calculating the eigenvalues of a large (even sparse) matrix is computationally expensive. In addition, the aim here is to represent the *global* structure of a graph and so precise estimates of *all* the eigenvalue values are not required. Thus, the distribution[2] of eigenvalues is sufficient. In this chapter, the distribution of eigenvalues $f(\lambda = k)$ is estimated using pivoting and Sylvester's Law of Inertia to compute the number of eigenvalues that fall in a given interval. To estimate the distribution, we use K equally spaced bins.[3] A measure of the graph can then be constructed by considering the distribution of the eigenvalues as

$$\omega(G, N) = \sum_{k \in K} (1-k)^N f(\lambda = k) \tag{6.14}$$

where the elements of $\omega(G, N)$ form the *weighted spectral distribution*

$$WSD : G \to \Re^{|K|}\{k \in K : ((1-k)^N f(\lambda = k))\} \tag{6.15}$$

In addition, a metric can then be constructed from $\omega(G)$ for comparing two graphs, G_1 and G_2, as

$$\Im(G_1, G_2, N) = \sum_{k \in K} (1-k)^N (f_1(\lambda = k) - f_2(\lambda = k))^2 \tag{6.16}$$

where f_1 and f_2 are the eigenvalue distributions of G_1 and G_2 and the distribution of eigenvalues is estimated in the set K of bins $\in [0, 2]$. Equation 6.16 satisfies all the properties of a metric [10].

We next wish to test if the WSD for graphs generated by the same underlying process vary significantly (to show that the WSD is stable). To do this, we generate a set of graphs that have very similar structure and test to see if their WSDs are also similar. The results of an empirical test are shown in Figure 6.1. This plot was created by generating 50 topologies using the AB [11] generator with the (fixed) optimum parameters determined in Section 6.6, but with different initial conditions.[4] For each run the spectral and weighted spectral distributions are recorded yielding 50×50 bin values which are then used to estimate standard deviations. As the underlying model (i.e., the AB generator) is the same for each run, the *structure* might be expected to remain the same and so a "structural metric" should be insensitive to random initial

[2]The eigenvalues of a given graph are deterministic and so *distribution* here is not meant in a statistical sense.

[3]K can be increased depending on the granularity required.

[4]We found similar results for other parameters and topology generators.

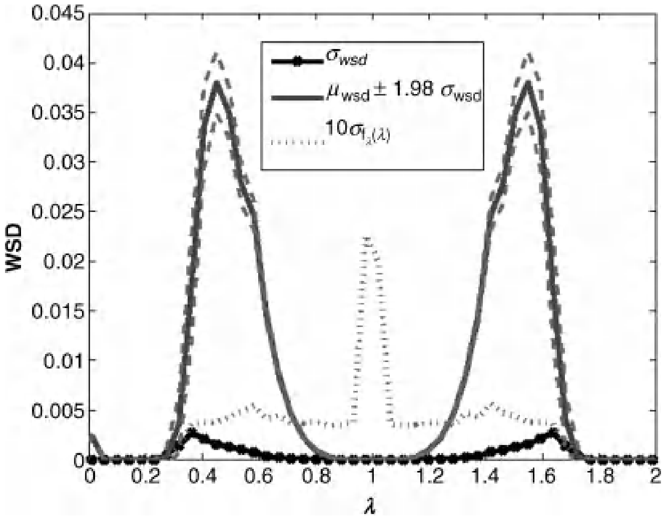


FIGURE 6.1 Mean and standard deviations for WSD and (unweighted) spectrum for the AB model over 50 simulations.

conditions. As can be seen the standard deviation[5] of the (unweighted) spectrum, $\sigma_{f_\lambda}(\lambda)$, is significantly higher at the center of the spectrum. However, for the WSD, the standard deviation, σ_{wsd}, peaks at the same point as the WSD; the noise in the spectral gap has been suppressed. The evidence suggests that the WSD successfully filters out the noise around 1 in the middle region and highlights the important parts of the signal.

6.3 A SIMPLE WORKED EXAMPLE

After the fairly theoretical previous section, we aim at giving the reader a better intuition behind the WSD with a simple example. Figure 6.2 shows a small network, called G_1, with seven nodes and eight links. As can be seen there are two cycles of length 3 in this network and one of length 4. We will take $N = 3$ in this example for convenience and without loss of generality. The random walk probabilities are labeled in Figure 6.2. For example, node 3 has a degree of 5 resulting in a probability of 1/5th for each edge. The total probability of taking a random walk around each 3-cycle is $6 \times 1/2 \times 1/3 \times 1/3 = 0.33$, also shown in Figure.[6]

Figure 6.3 shows a 3D plot of the absolute value (for clarity) of the eigenvectors of the normalized Laplacian. The corresponding eigenvalues are shown in Table 6.1.

[5]Multiplied by a factor of ten for clarity.

[6]The six comes from the fact that the random walk can start in one of the three nodes and go in one of the two directions. It can be viewed in our case as really just a nuisance scaling factor.

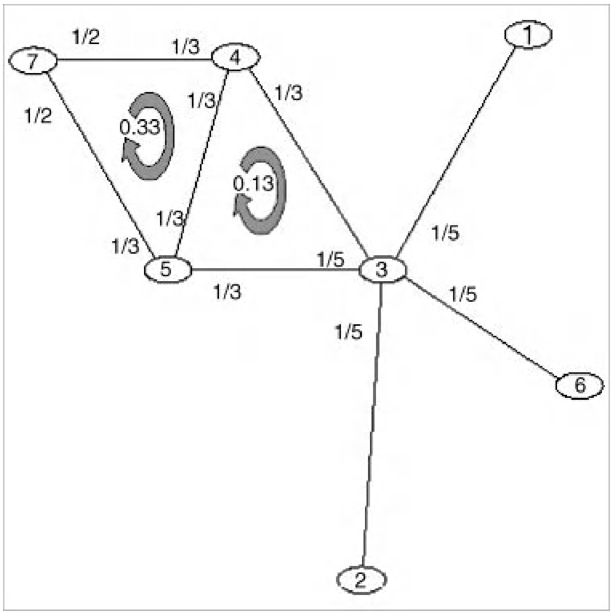


FIGURE 6.2 A simple example network G_1.

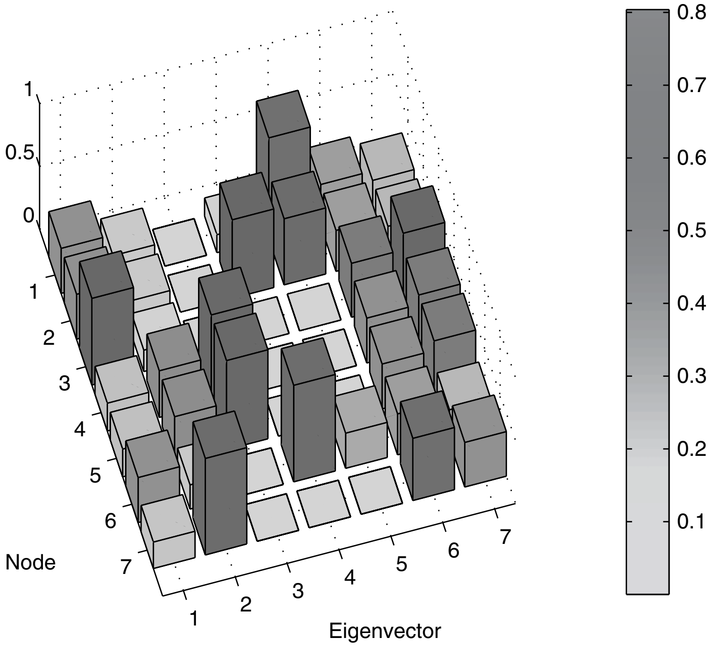


FIGURE 6.3 Eigenvectors of the simple example network.

TABLE 6.1 Eigenvalues, WSD, and Dominant Nodes of Example Network

e_7	Eigenvector	λ	$1-\lambda$	$(1-\lambda)^3$	Dominant Nodes
0.2500	1	1.8615	−0.8615	−0.6394	3, 1, 2, 6
0.2500	2	1.3942	−0.3942	−0.0612	7, 4, 5
0.5590	3	1.3333	−0.3333	−0.0370	4, 5
0.4330	4	1.0000	0.0000	0.0000	6, 2
0.4330	5	1.0000	0.0000	0.0000	1, 2, 6
0.2500	6	0.4110	0.5890	0.2043	7, 3
0.3536	7	0.0000	1.0000	1.0000	3, 4, 5, 7
	$\sum_{i=1}^{7}(1-\lambda_i)^3$			0.4667	

The eigenvectors of the normalized Laplacian can be used to form a partitioning of the nodes in a graph. In this example, nodes 4 and 5 are grouped into eigenvector 3, nodes 1, 2, and 6 into eigenvectors 4 and 5, node 7 into eigenvector 2, and node 3 into eigenvector 1 (Fig. 6.3). Note that for each partition the nodes in the partition are the same; that is, we could swap the labels between nodes 4 and 5 and the network would not change (i.e., an isomorphism). Eigenvector and eigenvalue 7, e_7 and $\lambda_7 = 0$, are special and partition all the nodes in the network with the most central nodes having the highest coefficients (see Table 6.1, Column 1). In general the number of eigenvalues that are zero is equal to the number of components, arguably the most important structural property in a graph. This graph contains one connected component and so has a single zero eigenvalue (λ_7). Note that the highest possible weighting in the WSD is given at zero (i.e., $1 = 1 - 0$); the number of components in the graph.

Note that the sum of the eigenvalues taken to the power of N is indeed the same as the sum of the probabilities of taking N random walk cycles in the graph. This is shown in Table 6.1, last row, $\sum_{i=1}^{7}(1-\lambda_i)^3 = 0.4667$ which can be easily verified by adding the cycle probabilities from Figure 6.3 ($0.3333 + 0.1333 = 0.467$). What is interesting is how this sum is constructed. In Table 6.1, the main contributions to the sum are from eigenvalues 1, 2, 3, and 6 (we ignore eigenvalue 7 as it merely reflects that the graph is connected) which are dominated by the nodes which form the cycles; 3, 4, 5, and 7.

However, this does not mean that the information provided by the WSD is confined to N-cycles in the graph. For example in Figure 6.5, we take the edge linking nodes 1 and 3 and rewire it so that 1 and 6 are now connected. Note that while the right cycle is still in place its probabilities have now changed, as the degree of node 3 is now 4. The corresponding eigenvalues have also changed as seen in Figure 6.4.[7]

In conclusion, the WSD can roughly be seen as an amalgamation of *local* views (i.e., walks of length N) taken from all the nodes. As $(1-\lambda_i) \leq 1\ \forall i$, $(1-\lambda_i)^N$ will

[7]Note that if we had used the adjacency matrix instead of the normalized Laplacian the rewiring would have no effect on the sum of the eigenvalues.

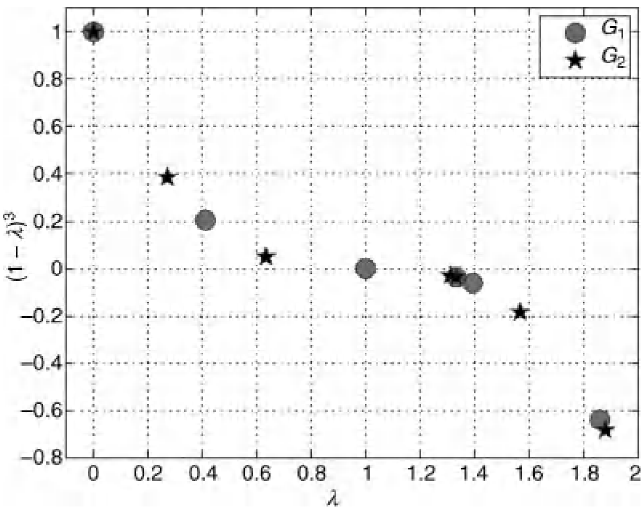


FIGURE 6.4 The second example network, G_2.

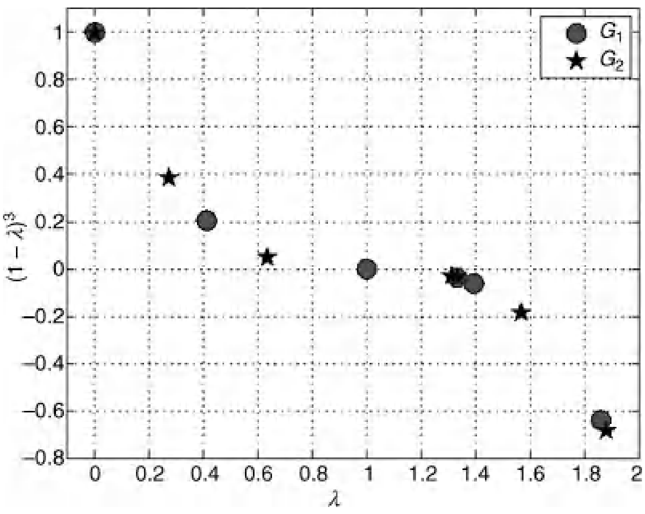


FIGURE 6.5 WSD of the example network.

suppress the smaller eigenvalues more and more as N increases.[8] We consider 3 and 4 to be suitable values of N for the current application: $N = 3$ is related to the well-known and understood clustering coefficient; and $N = 4$ as a 4-cycle represents two routes (i.e., minimal redundancy) between two nodes. For other applications, other

[8]This is closely related to the settling times in Markov chains, which are often expressed in terms of the largest nontrivial eigenvalue. It differs in that the Walk Laplacian and not the normalized Laplacian is used.

values of N may be of interest. Also note that in Section 6.2, we propose using the *distribution* of the eigenvalues for large networks; unfortunately it is not instructive to talk about a distribution for a small number of eigenvalues (7 in this example).

6.4 THE INTERNET AUTONOMOUS SYSTEM TOPOLOGY

The Internet's AS topology is a widely studied representation of the Internet at a particular scale. An AS represents a single network that can apply its own operational and peering policy. An Internet service provider (ISP) may use one or more ASes. The Internet contains over 30,000 ASes, each in a set of relationships with its neighbors, who are either its customers, providers, or peers. In the Internet core there is a full mesh formed between the ASes of the various tier-1 ISPs. However, at the edge there are a huge number of smaller ISPs and customer networks which connect through upstream providers and local public exchange points. These smaller ISPs and customer networks may have only one upstream provider, or may have many for resilience and performance reasons. In addition, the Internet constantly evolves new networks are added, old ones disappear, and existing ones grow and merge.

Links between ASes depend on business relationships which can and do change, sometimes rapidly, making any interpretation of the Internet as a *static* structure inaccurate. This rich and dynamic structure makes it difficult to provide either a single, representative topological model, or a single graph metric that captures all characteristics of any topology. However, such a metric would make it possible to generate realistic synthetic topologies improving the accuracy of Internet-wide protocol simulations, and perhaps enabling the prediction of the future evolution of the Internet's topology.

Many attempts to capture one or more characteristics have been made, resulting in several topology generators each of which synthesize Internet-like topologies using different models and parameters. Unfortunately, validating these models is an *ad hoc* matter that typically means matching several topological measures in the hope that this will ensure a matching structure. Users often select default parameters for these models based on specific datasets measured at particular times, which no longer represent the current Internet. However, as noted previously, these measures cannot be used to estimate the optimum parameters for a model given a target topology.

6.4.1 Characterization

Over the past several years many topological metrics have been proposed for quantitatively characterizing topological properties of networks. In this section, we present a large set of topological metrics that will be used to measure a *distance* in graph space,[9] that is, how topologically distant two graphs are from each other. These

[9]In Ref. [12] we present an even larger set of measures.

metrics are computed for both synthetic and measured AS topologies. When choosing our metrics we considered both those used by the topology generator designers and those used more widely in the graph theory literature. Taken individually, these metrics focus on different topological aspects, but when considered together they reveal a more complete picture of the observed AS topologies.

We specifically chose not to use the three metrics of Tangmunarunkit et al. [13] for two reasons. First, computation of resilience and distortion are both NP-complete, requiring the use of heuristics. In contrast, all our metrics are straightforward to compute directly. Second, although accurate reproduction of degree-based metrics is well supported by current topology generators, our hypothesis is that local interconnectivity has been poorly understood, and so we add several metrics that focus on exactly this, for example, assortativity, clustering, and centrality.

AS topologies are modeled as graphs $G = (V, E)$ with a collection of nodes V and a collection of links E that connect a pair of nodes. The number of nodes and links in a graph is then equal to, respectively, $N = |V|$ and $M = |E|$.

Degree. The degree k of a node is the number of links adjacent to it. The *average node degree* $\bar{k}$ is defined as $\bar{k} = 2M/N$. The *node degree distribution* $P(k)$ is the probability that a randomly selected node has a given degree k. The node degree distribution is defined as $P(k) = n(k)/N$, where $n(k)$ is the number of nodes of degree k. The *joint degree distribution* (JDD) $P(k, k')$ is the probability that a randomly selected pair of connected nodes have degrees k and k'. A summary measure of the joint degree distribution is the average neighbor degree of nodes with a given degree k, and is defined as follows $k_{nn}(k) = \sum_{k'=1}^{k_{\max}} k' P(k'|k)$. The maximum possible $k_{nn}(k)$ value is $N - 1$ for a maximally connected network, that is, a complete graph. Hence, we represent the JDD by the normalized value $k_{nn}(k)/(N - 1)$ [14] and refer to it as *average neighbor connectivity*.

Assortativity. Assortativity is a measure of the likelihood of connection of nodes of similar degrees [15]. This is usually expressed by means of the *assortativity coefficient* r: assortative networks have $r > 0$ (disassortative have $r < 0$ respectively) and tend to have nodes that are connected to nodes with similar (dissimilar respectively) degree.

Clustering. Given node i with k_i links, these links could be involved in at most $k_i(k_i - 1)/2$ triangles (e.g., nodes $a \to b \to c \to a$ form a triangle). The greater the number of triangles, the greater the clustering of this node. The clustering coefficient $\gamma(G)$ is defined as the average number of triangles divided by the total number of possible triangles: $\gamma(G) = 1/N \sum_i \frac{T_i}{k_i(k_i-1)/2}, k_i \geq 2$ where T_i is the number of triangles of node i and k_i is its degree. We use the distribution of *clustering coefficients* $C(k)$, which in fact is the distribution of the terms $\frac{T_i}{k_i(k_i-1)/2}$ in the overall summation. This definition of the clustering coefficient gives the same weight to each triangle in the network, irrespective of the distribution of the node degrees.

Rich-Club. The *rich-club coefficient* $\phi(\rho)$ is the ratio of the number of links in the component induced by the ρ largest-degree nodes to the maximum possible links $\rho(\rho - 1)/2$, where $\rho = 1, \dots, N$ are the first ρ nodes ordered by their degree ranks in a network of size N nodes and ρ is normalized by the total number of nodes N [16,17]. In this way, the node rank ρ denotes the position of a node on this ordered list.

Shortest Path. The shortest path length distribution $P(h)$ is the probability distribution of two nodes being at minimum distance h hops from each other. From the shortest path length distribution the average node distance in a connected network is derived as $\overline{h} = \sum_{h=1}^{h_{\max}} hP(h)$, where $h_{\max}$ is the longest shortest path between any pair of nodes. $h_{\max}$ is also referred to as the diameter of a network.

Centrality. Betweenness centrality is a measure of the number of shortest paths passing through a node or a link. The *node betweenness* for a node v is $B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$, where σ_{st} is the number of shortest paths from s to t and $\sigma_{st}(v)$ is the number of shortest paths from s to t that pass through a node v [18]. The average node betweenness is the average value of the node betweenness over all nodes.

Closeness. Another measure of the centrality of a node within a network is its *closeness*. The closeness of a node is the reciprocal of the sum of shortest paths from this node to all other reachable nodes in a graph.

Coreness. The l-core of a network (sometimes known as the k-core) is the maximal component in which each node has at least degree l. In other words, the l-core is the component of a network obtained by recursively removing all nodes of degree less than l. A node has coreness l if it belongs to the l-core but not to the $(l + 1)$-core. Hence, the l-core is the collection of all nodes having coreness l. The core of a network is the l-core such that the $(l + 1)$-core is empty [19].

Clique. A clique in a network is a set of pairwise adjacent nodes, that is, a component which forms a complete graph. The *top clique size*, also known as the graph clique number, is the number of nodes in the largest clique in a network [20].

Spectrum. It has recently been observed that eigenvalues are closely related to almost all critical network characteristics [8]. For example, Tangmunarunkit et al. [13] classified network resilience as a measure of network robustness subject to link failures, resulting in a minimum balanced cut size of a network. Spectral graph theory enables study of network partitioning using graph eigenvalues [8]. In this chapter, we focus on the spectrum of the *normalized Laplacian matrix*, where all eigenvalues lie between 0 and 2, allowing easy comparison of networks of different sizes. We use the normalized graph's spectrum for tuning the parameters of topology generators.

6.4.2 Generation

In this section, we present a number of topology generators, each having their own set of parameters. We also present an example of an Internet AS topology dataset which we use as a litmus test for the parameter tuning exercise.

There are many models available that claim to describe the Internet AS topology. Several of these are embodied in tools built by the community for generating simulated topologies. In this section, we describe the particular models whose output we compare in this chapter. The first are produced from the Waxman model [21], derived from the Erdös–Rényi random graphs [22], where the probability of two nodes being connected is proportional to the Euclidean distance between them. The second come from the Barabasi and Albert (BA) [23] model, following measurements of various power laws in degree distributions and rank exponents by Faloutsos et al. [24]. These incorporate common beliefs about preferential attachment and incremental growth. The third are from the generalized linear preference model [25] which additionally model clustering coefficients. Finally, Inet [26] and PFP [17] focus on alternative characteristics of AS topology the existence of a meshed core, and the phenomenon of preferential attachment, respectively. Each model focuses only on particular metrics and parameters, and has only been compared with selected AS topology observations [13,26,27].

Waxman. The Waxman model of random graphs is based on a probability model for interconnecting nodes of the topology given by

$$P(u, v) = \alpha e^{-d/(\beta L)} \tag{6.17}$$

where $0 < \alpha, \beta \leq 1$, d is the Euclidean distance between two nodes u and v, and L is the network diameter, that is, the largest distance between two nodes. Note that d and L are not parameters for the Waxman model. The Internet is known not to be a random network but we include the Waxman model as a baseline for comparison purposes.

BA. The BA [11] model was inspired by the idea of preferentially attaching new nodes to existing well-connected nodes, leading to the incremental growth of nodes and the links between them. Starting with a network of m_0 isolated nodes, $m \leq m_0$ new links are added with probability p. One end of each link is attached to a random node, while the other end is attached to a node selected by preferring the more popular, that is, well connected, nodes with probability

$$\Pi(k_i) = \frac{k_i + 1}{\sum_j k_j + 1} \tag{6.18}$$

where k_j is the degree of node j, with probability q, m links are rewired and new nodes are added with probability $1 - p - q$. A new node has m new links that, with probability $\Pi(k_i)$, are connected to nodes i already present in the system. We use the BRITE [28] implementation of this model in this chapter.

GLP. Our third model is the generalized linear preference (GLP) model [25]. It focuses on matching characteristic path length and clustering coefficients. It uses a probabilistic method for adding nodes and links recursively while preserving selected power law properties. In the GLP model, when starting with m_0 links, the probability of adding new links is defined as p where $p \in [0, 1]$. Let $\Pi(d_i)$ be the probability of choosing node i. For each end of each link, node i is chosen with probability $\Pi(d_i)$ defined as

$$\Pi(d_i) = (d_i - \beta) / \sum_j (d_j - \beta) \tag{6.19}$$

where $\beta \in (-\infty, 1)$ is a tunable parameter indicating the preference of nodes to connect to existing popular nodes. We use the BRITE implementation of this model in this chapter.

Inet. Inet [26] produces random networks using a preferential linear weight for the connection probability of nodes after modeling the core of the generated topology as a full mesh network. Inet sets the minimum number of nodes at 3037, the number of ASes on the Internet at the time of Inet's development. By default, the fraction of degree 1 nodes α is set to 0.3, based on measurements from Routeviews[10] and NLANR[11] BGP table data in 2002.

PFP. In the positive feedback preference (PFP) model [17], the AS topology of the Internet is considered to grow by interactive probabilistic addition of new nodes and links. It uses a nonlinear preferential attachment probability when choosing older nodes for the interactive growth of the network, inserting edges between existing and newly added nodes. As the PFP generator does not have any user-tunable parameters we include it only in the last part of Section 6.6 for completeness.

6.4.3 Observations

The AS topology can be inferred from two main sources of data, BGP and traceroutes, both of which suffer from measurement artifacts. BGP data is inherently incomplete no matter how many vantage points are used for collection. In particular, even if BGP updates are combined from multiple vantage points, many peering and sibling relationships are not observed [29]. Traceroute data misses alternative paths since routers may have multiple interfaces which are not easily identified, and multihop paths may be hidden by tunnelling via multiprotocol label switching (MPLS). In addition, mapping traceroute data to AS numbers is often inaccurate [30].

Chinese. The first dataset is a traceroute measurement of the Chinese AS Topology collected from servers within China in May 2005. It reports 84 ASs, representing a small subgraph of the Internet. Zhou et al. [31] claim that the Chinese AS graph exhibits all the major topology characteristics of the global AS graph. The presence of this dataset enables us to compare the AS topology models at smaller scales. Further, this dataset is believed to be nearly complete, that is, it contains very little measurement bias and accurately represents the AS topology of that region of the Internet. Thus, although it is rather small, we have included it as a valuable comparison point in our studies.

[10]http://www.routeviews.org/
[11]http://www.nlanr.net/

Skitter. The second dataset comes from the CAIDA Skitter project.[12] By running traceroutes toward a large range of IP addresses and subsequently mapping the prefixes to AS numbers using RouteViews BGP data, CAIDA computes an observation of the AS topology. For our study, we use the graphs from March 2004 to match those used by Mahadevan et al. [14]. This AS topology reports 9204 unique ASs.

RouteViews. The third dataset we use is derived from the RouteViews BGP data. This is collected both as static snapshots of the BGP routing tables and dynamic BGP data in the form of BGP update and withdrawal messages. We use the topologies provided by Mahadevan et al. [14] from both the static and dynamic BGP data from March 2004. The dataset is produced by filtering AS sets and private ASs and merging the 31 daily graphs into one. This dataset reports 17,446 unique ASs across 43 vantage points in the Internet.

UCLA. The fourth dataset comes from the Internet topology collection[13] maintained by Oliveira et al. [32]. These topologies are updated daily using BGP routing tables and updates from RouteViews, RIPE,[14] Abilene,[15] and LookingGlass servers. We use a snapshot of this dataset from November 2007, computed using a time window on the last-seen timestamps to discard ASs which have not been seen for more than 6 months. The resulting dataset reports 28,899 unique ASs.

6.5 COMPARING TOPOLOGY GENERATORS

Most past comparisons of topology generators have been limited to the average node degree, the node degree distribution and the joint degree distribution. The rationale for choosing these metrics is that if those properties are closely reproduced then the value of other metrics will also be closely reproduced [33].

In this section, we show that current topology generators are able to match first and second order properties well, that is, average node degree and node degree distribution, but fail to match many other topological metrics. We also discuss the importance of various metrics in our analysis.[16]

6.5.1 Methodology

For each generator, we specify the required number of nodes and generate 10 topologies of that size to provide confidence intervals for the metrics. We then compute the metrics introduced in Section 6.4 on both the generated and observed AS topologies. All topologies studied in this chapter are undirected, preventing us

[12]http://www.caida.org/tools/measurement/Skitter/

[13]http://irl.cs.ucla.edu/topology/

[14]http://www.ripe.net/db/irr.html

[15]http://abilene.internet2.edu/

[16]We present an extended set of metrics in Ref. [12] which further support our claims; we restrict ourselves here to only the most significant results.

from considering peering policies and provider–customer relationships. This limitation is forced upon us by the design of the generators as they do not take such policies into account.

Each topology generator uses several parameters, all of which could be tuned to best fit a particular size of topology. However, there are two problems with attempting this tuning. First, doing so requires selecting an appropriate goodness-of-fit measure, of which there are many as noted in Section 6.4. Second, in any case tuning parameters to a particular dataset is of questionable merit since, as we argued in Section 6.1, each dataset is but a sample of reality, having many biases and inaccuracies. Typically, topology generator parameters are tuned to match the number of links in the synthetic and measured networks for a given number of nodes. However, we found this to be infeasible as generating graphs with equal numbers of links from a random model and a power–law model gives completely different outputs. For space reasons, we deal with this particular issue elsewhere [34]; in this chapter, we simply use the default values embedded within each generator.

6.5.2 Topological Metrics

In this section, we discuss the results for each metric separately and analyze the reasons for differences between the observed and the generated topologies.

Table 6.2 displays the values of various metrics (columns) computed for different topologies (rows). Blocks of rows correspond to a single observed topology and the generated topologies with the same number of nodes as the observed topology. Bold numbers represent nearest match of a metric value to that for the relevant observed topology. Rows in each block are ordered with the observed topology first, followed by the generated topologies from oldest to newest generator. For synthetic topologies, the value of the metrics is averaged over the 10 generated instances. Note that Inet requires the number of nodes to be greater than 3037 and hence cannot be compared to the Chinese topology.

We observe a small but measurable improvement from older to newer generators in some metrics such as maximum degree, maximum coreness, and assortativity coefficient. This suggests that topology generators have successively improved at matching particular properties of the observed topologies. However, the number of links in the generated topologies may differ considerably from the observed topology due to the assumptions made by the generators. The Waxman and BA generators fail to capture the maximum degree, the top clique size, maximum betweenness, and coreness. Those two generators are too simplistic in the assumptions they make about the connectivity of the graphs to generate realistic AS topologies. Waxman relies on a random graph model which cannot capture the clique between tier-1 ASes or the heavy tail of the node degree distribution. BA tries to reproduce the power–law node degrees with its preferential attachment model but fails to reach the maximum node degree, as it only adds edges between new nodes and not between existing ones. Hence, neither of these two models is able to create the highly connected core of tier-1 ASes. PFP and Inet manage to come closer to the values of the metrics of the observed topologies. For Inet this is because it assumes that 30% of the nodes are

TABLE 6.2 **Comparison of AS Level Dataset with Synthetic Topologies**

Topology	Links	Avg. Degree	Max. Degree	Top Clique Size	Max. Betweenness	Max. Coreness	Assort. Coef.	Clust. Coef.	Max. Closeness
Chinese	*211*	*5.02*	*38*	2	*1,324*	*5*	*−0.32*	*0.188*	*<0.01*
Waxman	252	6	18	**2**	404	4	0.039	0.117	**0.506**
BA	165	3.93	19	3	**1,096**	2	−0.096	0.073	0.515
GLP	151	3.6	44	3	2,391	**5**	−0.257	**0.119**	0.643
PFP	**250**	**5.95**	**37**	10	849	9	**−0.38**	0.309	0.638
Skitter	*28,959*	*6.3*	*2,070*	*16*	*10,210,533*	*28*	*−0.23*	*0.026*	*<0.01*
Waxman	**27,612**	**6**	33	0	474,673	4	0.205	0.002	**0.264**
BA	18,405	4	190	0	5,918,226	2	−0.05	0.001	0.315
GLP	16,744	3.64	**2,411**	2	34,853,544	5	−0.089	0.003	0.496
INET	18,504	4.02	1,683	3	15,037,631	7	−0.195	0.004	0.514
PFP	27,611	**6**	3,000	**16**	**13,355,194**	**24**	**−0.244**	**0.012**	0.588
RouteViews	*40,805*	*4.7*	*2,498*	*9*	*30,171,051*	*28*	*−0.19*	*0.02*	*<0.01*
Waxman	52,336	6	35	0	1,185,687	4	0.205	0.001	**0.25**
BA	34,889	4	392	3	33,178,669	2	−0.04	0.001	0.33
GLP	31,391	3.6	4,226	4	127,547,256	6	−0.08	0.002	0.48
INET	**43,343**	**4.97**	**2,828**	**6**	**31,267,607**	14	−0.258	0.006	0.522
PFP	52,338	6	4,593	23	39,037,735	**30**	**−0.252**	**0.009**	0.564
UCLA	*116,275*	*8.05*	*4,393*	*10*	*76,882,795*	*73*	*−0.165*	*0.05*	*0.32*
Waxman	86,697	6	40	0	3,384,114	4	0.213	<0.001	0.246
BA	57,795	4	347	0	52,023,288	2	−003	<0.001	**0.3**
GLP	52,456	3.63	7,391	2	371,651,147	6	−0.08	<0.001	0.486
INET	**91,052**	**6.3**	**6,537**	**12**	**88,052,316**	38	−0.3	**0.01**	0.55
PFP	86,696	6	8,076	26	123,490,676	**40**	**−0.218**	**0.01**	0.57

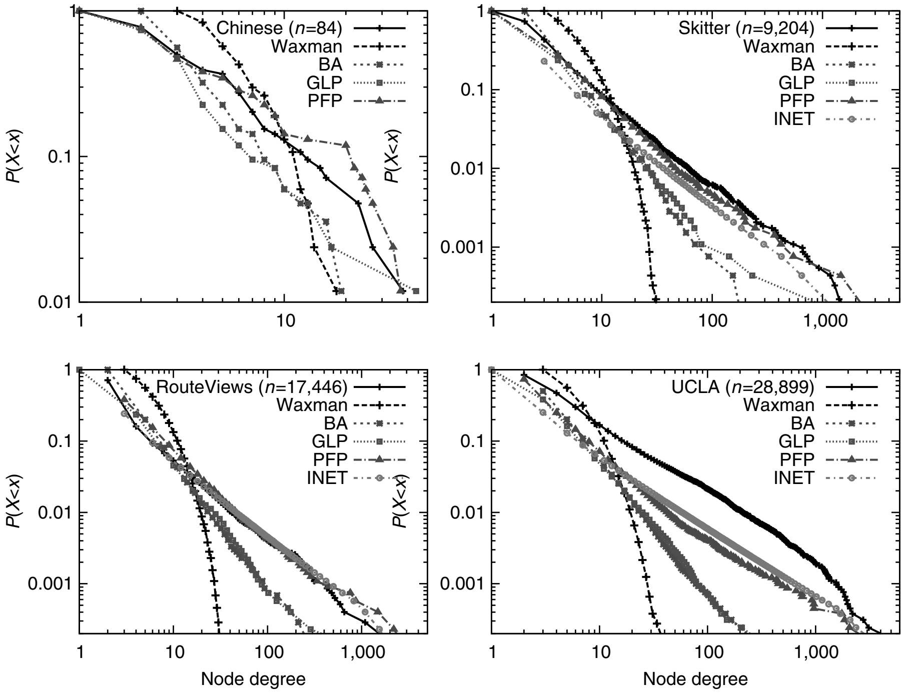


FIGURE 6.6 Comparison of node degree CCDFs.

fully meshed (at the core), whereas for PFP its rich-club connectivity model allows to add edges between existing nodes.

Node Degree Distribution. In Figure 6.6, we show the CCDF of the node degree for all topologies on a log–log scale. We observe that the Chinese topology does not exhibit power–law scaling due to its limited size, whereas all the larger AS topologies do exhibit power–law scaling of node degrees. The Waxman generator completely fails to capture this behavior as it is based on a random graph model, but recent topology generators do capture this power–law behavior of the node degrees quite well, as observed in Ref. [25]. In the case of the RouteViews and UCLA datasets, Inet and PFP outperform other topology generators. Note that the more complete UCLA dataset has a slightly concave shape in contrast to RouteViews where the degree distribution displays strict power–law scaling. In summary, more recent generation models reproduce node degree distributions well as expected since this has been a primary focus in the literature.

Average Neighbor Connectivity. Neighbor connectivity has been far less studied than node degree, although it is very important to match local interconnection among a node's neighbors when reproducing the topological structure of the Internet [14]. Figure 6.7 shows the CCDF of the average neighbor degrees

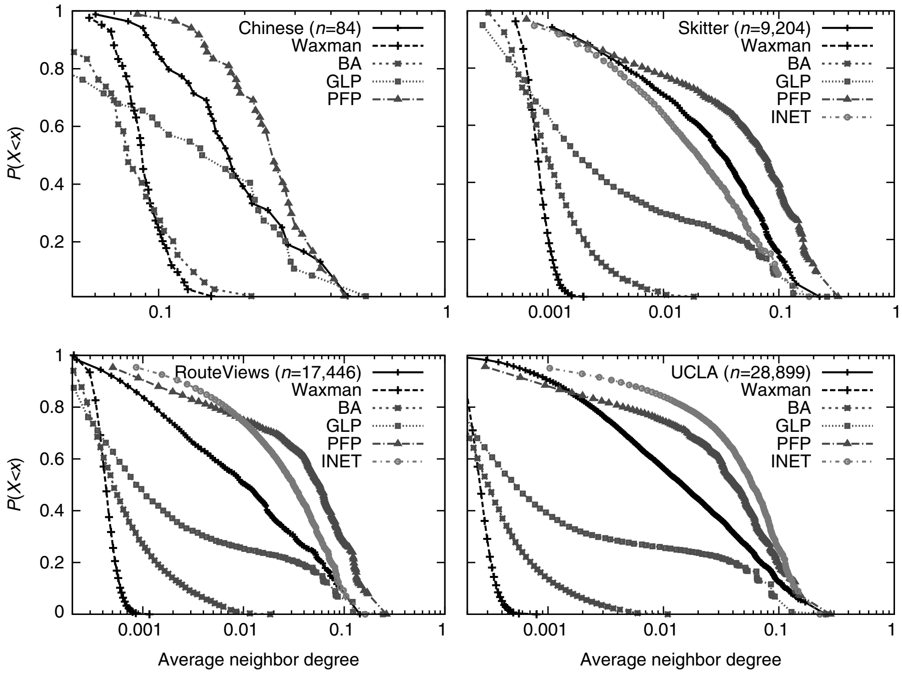


FIGURE 6.7 Comparison of average neighbor connectivity CCDFs.

for all topologies. We observe that Waxman, BA, and GLP underestimate the local interconnection structures around nodes. BA and GLP typically generate graphs with far fewer links than the observed topologies so they underestimate neighbor degrees on average.

For the larger observed topologies, that is, RouteViews and UCLA, PFP, and Inet typically overestimate the neighbor connectivity, as they both place a large number of inter-AS links in the core. In addition, the shapes of the neighbor connectivity CCDF differ for the larger topologies: Inet and PFP have two regimes, one for highly connected nodes (those with larger neighbor connectivity), and another for low-degree nodes. On the other hand, observed topologies have a smooth region for the high-degree nodes followed by a rather stable region caused by similar degree nodes. We observe that the highest degree nodes in the UCLA topology have very high values of neighbor connectivity. This is consistent with the belief that tier-1 providers are densely meshed.

Clustering Coefficients. Like the average neighbor connectivity, the clustering coefficient gives information about local connectivity of the nodes. It is important to reproduce clustering due to its impact on the local robustness in the graph: nodes with higher local clustering have increased local path diversity [14].

Figure 6.8 displays the clustering coefficients of all nodes in the topologies. Error bars indicate 95% confidence intervals around the mean values of the

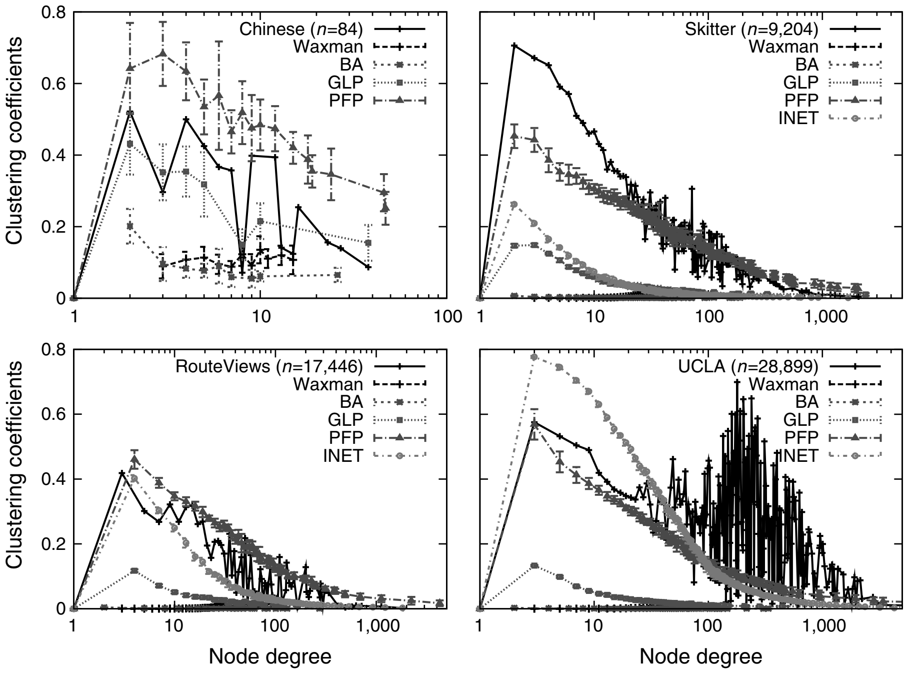


FIGURE 6.8 Comparison of clustering coefficients.

10 topologies from each generator. We observe that Waxman and BA significantly underestimate clustering, consistent with their simplistic way of connecting nodes. GLP approximates the clustering of the Chinese topology quite well but fails in the case of the larger observed topologies. PFP and Inet capture clustering reasonably well compared to the other topology generators. However, Inet does not reproduce the tail of the distribution well due to the randomness factor in its model for edge addition once the core is fully meshed.

We also observe that for medium degree nodes, clustering coefficients display rather high variability which increases with the size of the observed topologies. This behavior seems to be a property of the observed AS topology of the Internet.

In summary, all topology generators fail to properly capture clustering, typically underestimating local connectivity. Only Inet for the UCLA topology overestimates connectivity of low-degree nodes while still underestimating it for high-degree nodes. Current topology generators do not seem to adequately model local node connectivity.

Rich-Club Connectivity. Rich-club connectivity gives information about how well-connected nodes of high degree are among themselves. Figure 6.9 makes it clear that the cores of the observed topologies are very close to a full mesh, with values close to 1 on the left of the graphs. The error bars again indicate

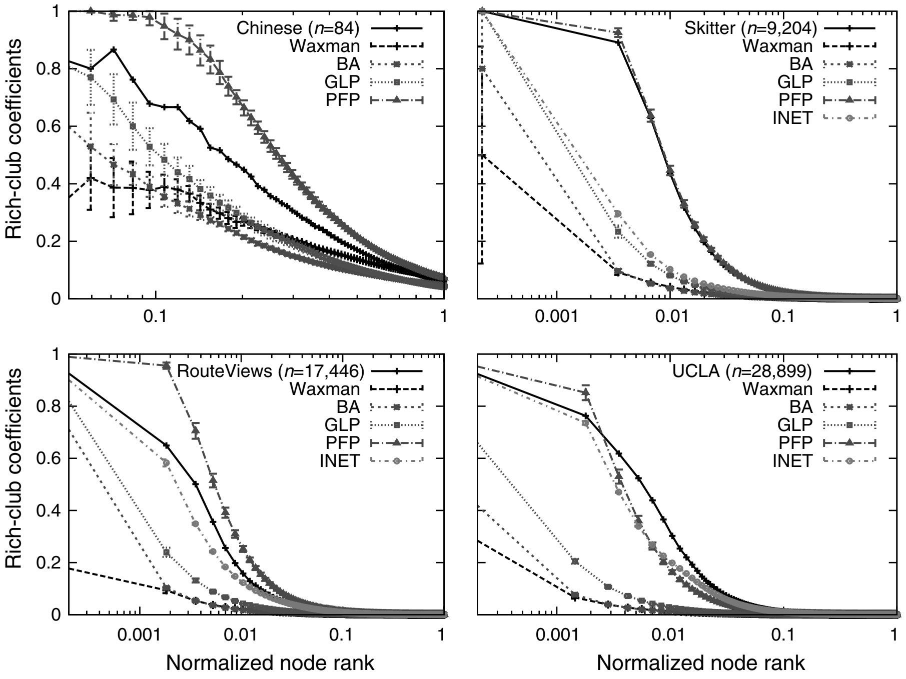


FIGURE 6.9 Comparison of rich-club connectivity coefficients

the 95% confidence intervals around the mean values of the different instances of the generated topologies. Waxman and BA perform poorly for this metric in general. Only PFP and Inet generate topologies with a dense enough core compared to the observed topologies. Given the emphasis that PFP gives to the rich-club connectivity, it overestimates it in the case of the Chinese and RouteViews topologies. Inet performs well due to its emphasis on a highly connected core, especially for larger topologies where data has been collected across multiple peering points.

In summary, most topology generators underestimate the importance of rich-club connectivity of the AS topology. PFP is the only topology generator that emphasizes the importance of the dense core of the AS topology.

Shortest Path Distributions. Figure 6.10 displays the distributions of shortest path length. Apart from BA, most topology generators approximate the shortest path length distribution of the Chinese graph quite well due to its small size. For the other topologies, PFP and Inet generally underestimate the path length distribution while Waxman and BA overestimate it. Particular generators seem to capture the path length distribution for particular topologies well, PFP matches Skitter's well and GLP is close for Routeviews. Inet and PFP both do a better job for UCLA than for RouteViews but both still underestimate the distribution.

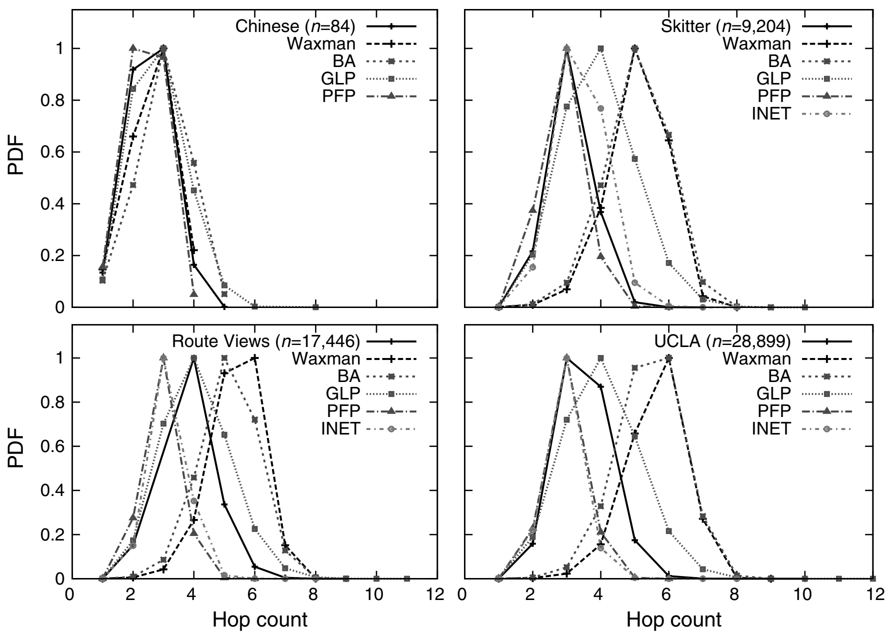


FIGURE 6.10 Comparison of shortest path distributions (number of hops).

In summary, shortest path length is not well captured by any topology generator. As shortest path length is related to local connectivity, failing to capture local connectivity is likely to lead to such a behavior.

Spectrum. The spectrum of the normalized Laplacian matrix is a powerful tool for characterizing properties of a graph. If two graphs have the same spectrum, they have the same topological structure.

Figure 6.11 displays the CDF of the eigenvalues computed from the normalized Laplacian matrix of each topology.

As with other topological metrics, Inet and PFP perform best. The difference between the topology generators is most easily observed around the eigenvalues equal to 1. These eigenvalues play a special role as they indicate repeated duplications of topological patterns within the network. By duplication, we mean different nodes having the same set of neighbors giving their induced subgraphs the same structure. Through repeated duplication, one can create networks with high multiplicity of eigenvalue 1 [35]. Further, if a network is bipartite, that is, it consists of two connected parts with no links between nodes of the same part, then its spectrum will be symmetric around 1. This phenomenon can also arise through repeated structure duplication.

We observe that the spectra have a high degree of symmetry around the eigenvalue 1, and so the observed AS topologies appear close in spectral terms to a bipartite graph. In the AS topology, many ASes share a similar set of upstream

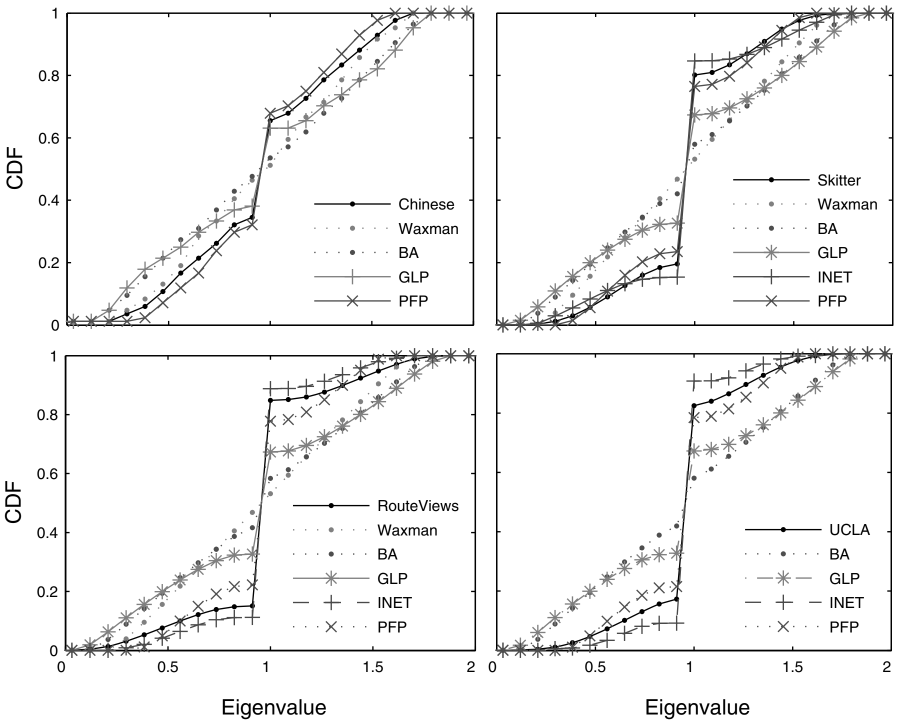


FIGURE 6.11 Comparison of cumulative distributions of eigenvalues (from normalized Laplacian).

ASes without being directly connected to each other. Inet and PFP are good examples of topology generators where this strategy is implemented. Note that the simple preferential attachment model of BA does not reproduce the eigenvalues around 1 very well. In the simple BA model, new nodes connect randomly to a given number of existing nodes, favoring connections to high degree nodes. In the Internet in contrast, although small ASes may tend to connect to large upstream providers, they might not connect preferentially to the largest ones, connecting instead to national or regional providers. In summary, these results provide further evidence that the interconnection structure of the AS topology is more complex than current models assume.

6.5.3 Discussion

Deviations between topology models and observations have been already studied in the literature. However, most works so far have focused on particular topological metrics. Concentrating on particular topological metrics has lead to underestimate the mismatch between the properties of observed AS topologies and what current models produce. When comparing several models with several observed AS topologies as we do, we see that current topology models mostly try to capture some properties of one set of observations from the AS topology. For a topology model to claim to model the Internet's AS topology, we would expect that it tries to approach the properties of observed AS topologies in many respects, which is not the case today.

6.6 TUNING TOPOLOGY GENERATOR PARAMETERS

The aim of this section is to examine how well the topology generators match the Skitter topology for different values of their parameters. To facilitate this comparison, grids are constructed over the possible values of the parameter spaces and various cost functions are evaluated as follows:

1. A cost function measuring the matching between the number of links in Skitter and the generated topologies
$$C_1(\theta) = (l_t(\theta) - l_{\text{Skitter}})^2 \tag{6.20}$$
where C_1 is the first cost function, θ are the model parameters (which differ for each topology generator), l_t is the number of links (which is a function of the parameters), and l_{Skitter} is the number of links in the Skitter dataset.
2. A cost function measuring the matching between the spectra of the Skitter network and of the generated topologies
$$C_2(\theta) = \sum_{k \in K} (f_t(\lambda = k) - f_{\text{Skitter}}(\lambda = k))^2 \tag{6.21}$$
where $f_t(\lambda = k)$ is the number of eigenvalues that fall in bin k for topology t. Note that $f_t(\lambda = k)$ is dependant on θ.
3. A cost function measuring the matching of the weighted spectral distributions
$$C_3(\theta) = \Im(G_t, G_{\text{Skitter}}, N) \tag{6.22}$$
as defined in Equation 6.16. Here, $N = 4$ is used.

In addition to examining different parameter values across a grid, the optimum parameters with respect to $C_3(\theta)$ are estimated using the Nelder Meade simplex search algorithm [36,37]. Note that the topologies generated by the topology generators are random in a statistical sense, due to differing random seeds for each run. Ten topologies are generated for each value of θ and the average spectral distribution is calculated. We found that the variance of the spectral distributions was sufficiently low to allow reasonable estimates of the minima in each case.

6.6.1 Link Densities

Figure 6.12 displays the value of the cost function $C_1(\theta)$ as a function of the topology generator parameters. On the upper and lower left graphs, the grayscale color indicates

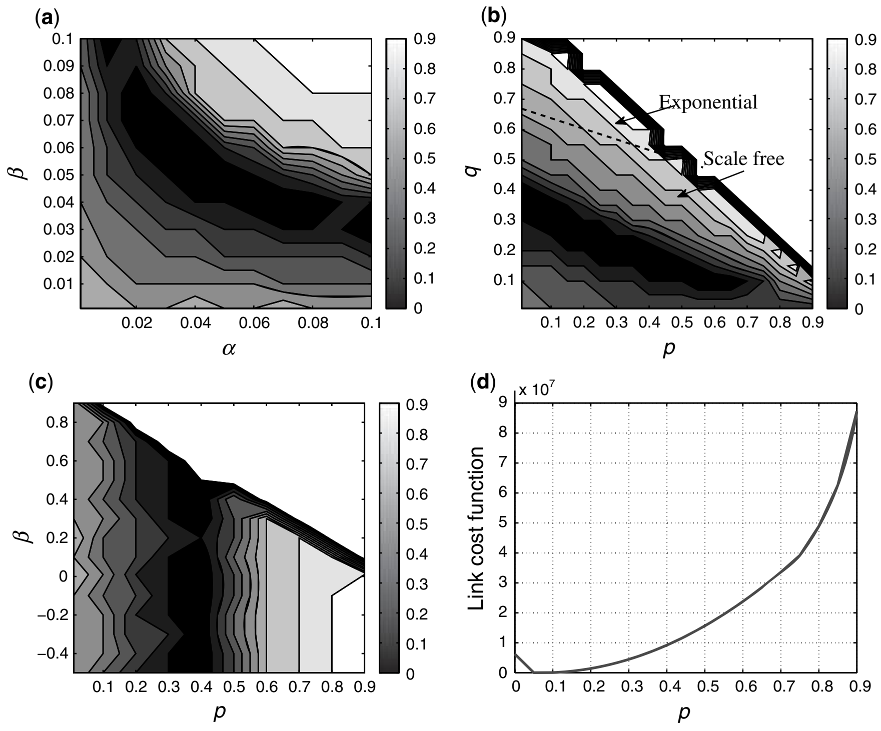


FIGURE 6.12 Topology generator parameter grid for sum-squared error from number of links. (a) Waxman. (b) BA. (c) GLP. (d) Inet.

the value of the cost function. For Inet there is only one parameter, p, so it is plotted as a curve in Figure 6.12d. Figure 6.12 shows that a minimum exists for each topology in approximately the same regions as the default values of each generator.[17] For the BA generator, it is known that for values of p and q above the line shown in Figure 6.12b, the topologies generated follow an exponential node degree distribution while those below follow a scale-free distribution. It is encouraging to note that the values of $C_1(\theta)$ are large in the exponential region and the minimum is in the scale-free region as the node degree distribution of the Internet is known to be approximately scale free [11]. Overall the results obtained by tuning the parameters based on $C_1(\theta)$ appear reasonable. For link density matching it is possible to obtain parameter values which match the link densities exactly. Indeed, there is a ridge of parameters for BA, GLP, and Waxman for which the link densities can be matched. However, as noted in the introduction, there is no control over any other characteristic of the graph using this method.

[17]Some of these default values are listed in Table 6.3.

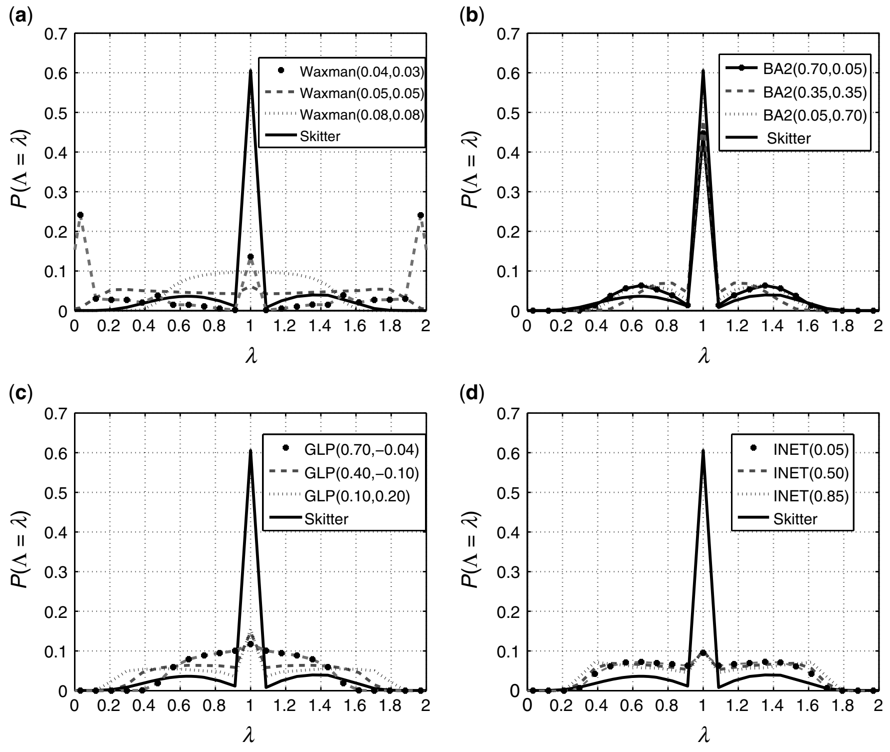


FIGURE 6.13 PDF of spectra. (a) Waxman. (b) BA. (c) GLP. (d) Inet.

6.6.2 Spectra PDF

Figure 6.13 shows the spectral PDF of the Skitter dataset and the four topology generators calculated at three parameters values in each grid (the parameter values are indicated in brackets in the legends). The aim is to illustrate how much the spectral PDFs change with the values of the parameters. The spectral PDFs of Waxman (Fig. 6.13a) vary significantly for different values of α and β. Furthermore, none of the Waxman PDFs match well with the spectral PDF of the Skitter graph. The BA PDFs vary to a lesser extent (Fig. 6.13b) and appear to give a much better match than the Waxman model, especially around eigenvalue 1 ($\lambda = 1$). This better match of BA is not surprising as the Waxman model is not a good model for the Internet as noted in Section 6.5. GLP (Fig. 6.13c) and Inet (Fig. 6.13d) give similar results to BA, with a poor match outside eigenvalue 1. The better match of the BA model around eigenvalue 1 is interesting. As noted in Section 6.2 the regions away from eigenvalue 1 are far more important than the region around $\lambda = 1$. However, what is required is a technique that reveals the differences with distance from one as these are more important. Thus, it would appear difficult to evaluate which model, or even which parameter, is better based on the PDFs alone. This point is now further explored by analysis of the grids calculated with respect to $C_2(\theta)$.

6.6.3 Limitations of Spectra PDF

Figure 6.14 shows the value of the second cost function $C_2(\theta)$ as a function of the topology generator parameters, in the same way as Figure 6.12. As can be seen in Figure 6.14, there are many islands corresponding to local minima, creating a rugged landscape. The variance in the PDFs referred to in this section is actually greater than any gradient that might exist in the grid. This means that it is not possible to estimate the minimum with respect to $C_2(\theta)$. Figure 6.14 shows that the spectrum on its own is not sufficient to identify the optimum parameters of any of the topology generators. This is because each eigenvalue in $C_2(\theta)$ is weighted equally. As noted in Section 6.2, the eigenvalues close to 1 are more likely to be affected by the random seeds for each topology generator and are the source of the noise on the grid.

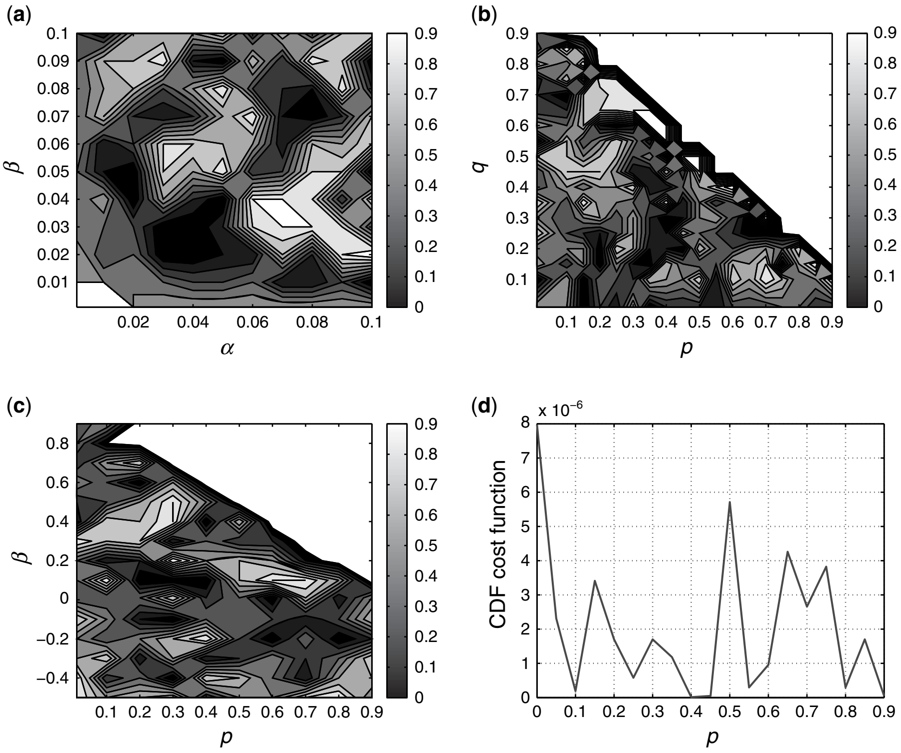


FIGURE 6.14 Parameter grid for sum of absolute differences of spectra CDFs. (a) Waxman. (b) BA. (c) GLP. (d) Inet.

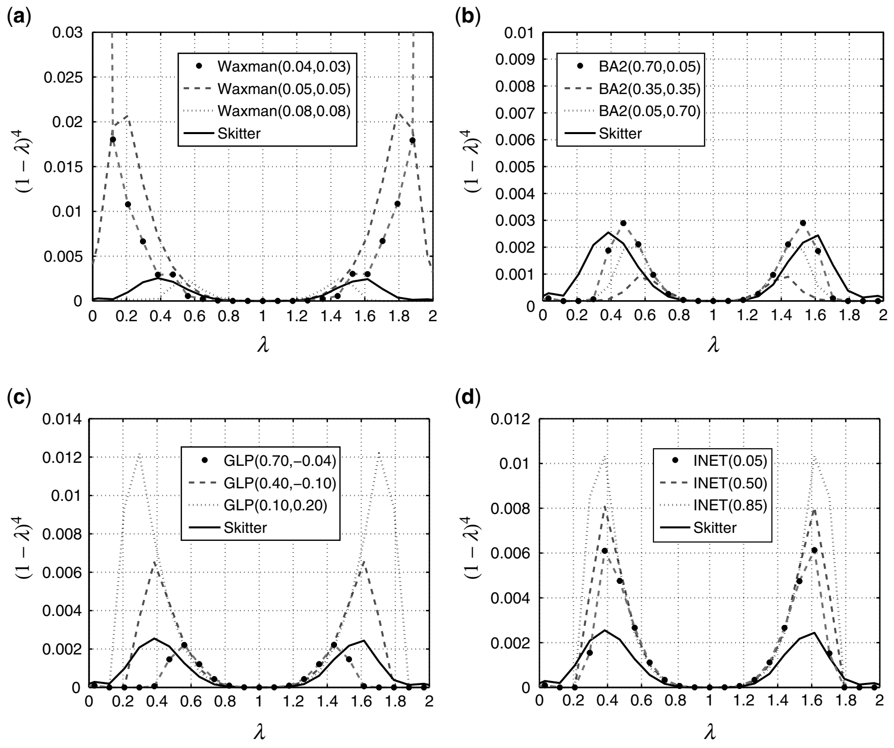


FIGURE 6.15 Weighted spectra grid for generator parameters. (a) Waxman. (b) BA. (c) GLP. (d) Inet.

6.6.4 Weighted Spectra

The previous section illustrated the limitations of using the raw eigenvalues to find optimal topology generator parameters to match the Skitter topology. Figure 6.15 shows a plot of the weighted spectra of the same topologies as those shown in Figure 6.13. As can be seen the results are quite different from those shown in Figure 6.13. The Waxman weighted spectra still shows a bad fit with respect to the Skitter data (mainly around 0 and 2) compared to the other generators. The other generators (BA, GLP, and Inet) now show that they are capable of matching the weighted spectra of the Skitter topology, especially around the point of greatest weight ($\lambda = 0.4$ or 1.6). The difference between the weighted spectra around 1 is no longer of importance (in contrast to Fig. 6.13), reflecting that the weights here approach zero as we approach eigenvalue 1. In the next section, the optimum values and the resulting weighted spectra will be compared.

6.6.5 Weighted Spectra Comparison

Figure 6.16 shows the grids associated with $C_3(\theta)$. As can be seen the grids show that there is a region with a minima in each case and in addition, comparing Figures 6.16

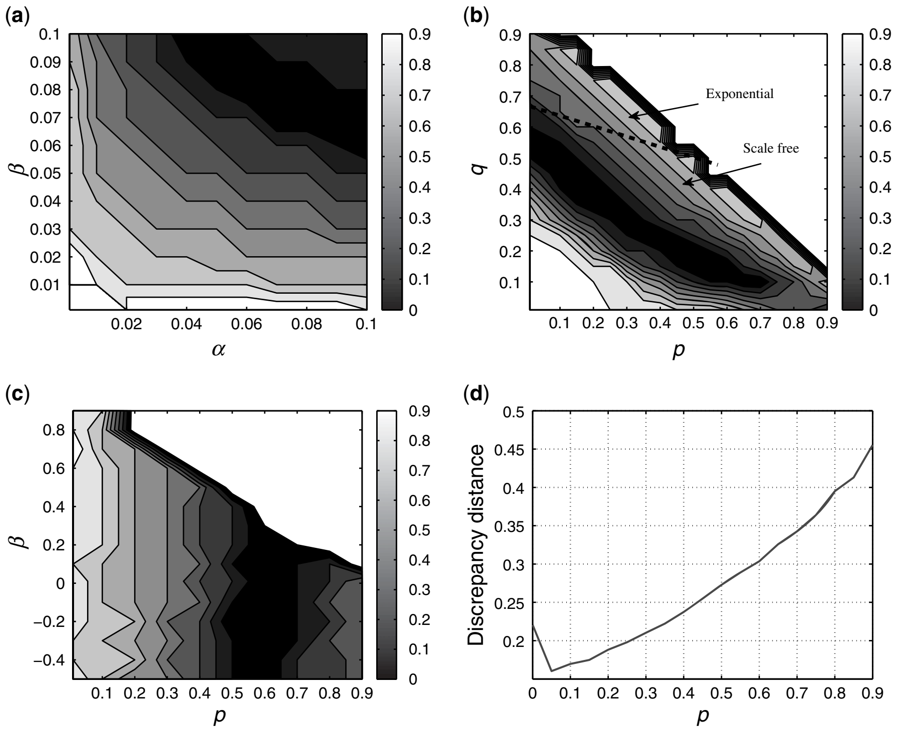


FIGURE 6.16 Grid of sum-squared error of weighted spectra for topology generators. (a) Waxman. (b) BA. (c) GLP. (d) Inet.

and 6.12 it can be seen that these minima lie in a region close to those for $C_1(\theta)$. However, it should be noted that the weighted spectra will try to fit more than just the number of links in a topology. This demonstrates the inherent trade-off. Also of note is that the region of interest for the BA model lies inside the region of scale-free behavior as shown in Figure 6.16b.

6.7 GENERATING TOPOLOGIES WITH OPTIMUM PARAMETERS

Table 6.3 displays the optimum values for the topology generators for generating networks that are close to the Skitter graph. In addition, we give the values for $C_3(\theta)$, which show that PFP gives the closest fit followed by BA, GLP, Waxman, and finally Inet. While these results are mostly expected, the ranking of Inet as the worst topology generator is surprising. We have also listed some of the default parameters used in certain generators such as BRITE [28]. While many of the optimized parameters are close to the default values, which is encouraging, it should be noted that the default parameters are for a *typical* graph and are not selected for any particular situation. Thus, a direct comparison is meaningless.

TABLE 6.3 Optimum Parameter Values for Matching Skitter Topology

Generator	Optimum and Default Parameter Values		$C_3(\theta)$	$\overline{C_3}(\theta)$
Waxman	$\alpha = 0.08$ (def. 0.15)	$\beta = 0.08$ (def. -0.2)	0.0026	0.0797
BA	$p = 0.2865$ (def. 0.6)	$q = 0.3145$ (def. 0.3)	0.0014	0.0300
GLP	$p = 0.5972$ (def. 0.45)	$\beta = 0.1004$ (def. 0.64)	0.0021	0.0446
Inet	$\alpha = 0.1013$ (def. 0.3)	–	0.0064	0.0150
PFP	–	–	0.0014	0.0371

Figure 6.17a shows the weighted spectra for each of the topology generators and inspection of the figure goes some way to explaining the discrepancy in the results. As can be seen the main peak in the weighted spectra for the Skitter data occurs at a value of $\lambda = 0.4$. The Waxman generator peak occurs at $\lambda = 0.6$ which is closer to 1 demonstrating the greater amount of random structure in the Waxman topologies. However, for the Inet generator the peak occurs at the correct point ($\lambda = 0.4$) but the weighted power at this point is far greater than in the Skitter topology. By normalizing the weighted spectrum this point becomes clear

$$\overline{C_3}(\theta) = \sum_{k\in K} \frac{(1-k)^N (f_t(\lambda = k))}{\omega(G_t)} - \frac{(f_2(\lambda = k))^2}{\omega(G_{\text{Skitter}})} \tag{6.23}$$

where $\omega(G)$ is the total power in the WSD as defined in Equation 6.14. Using the normalized weighted spectrum the results in Figure 6.17b show that Inet is the best match for the Skitter data while the Waxman model still performs worse than the other models. Further research is required before stating which version of C_3 is superior.

Figure 6.18 shows a comparison of the optimized topologies with respect to four typical network metrics: the node degree distribution, the average neighbor connectivity, the clustering coefficient, and the rich-club connectivity [17]. As can be seen PFP

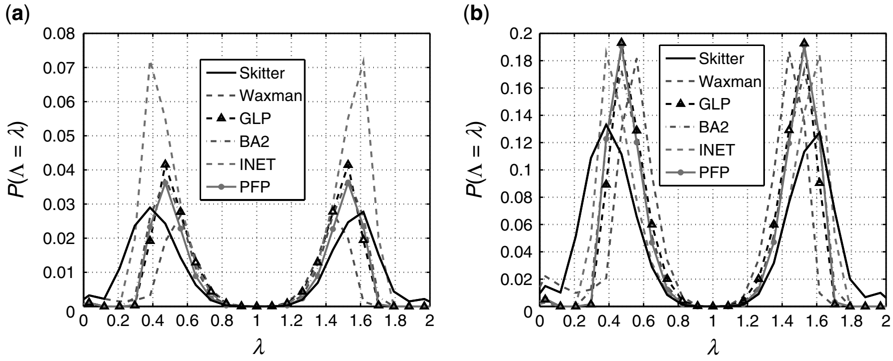


FIGURE 6.17 Comparison of the weighted spectra. (a) Weighted spectra. (b) Normalized weighted spectra.

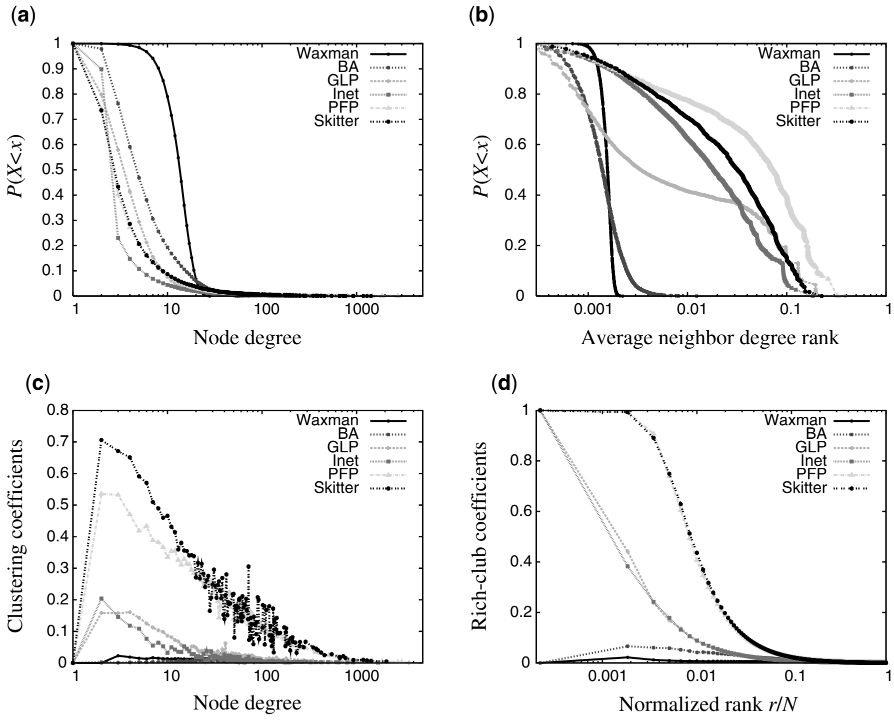


FIGURE 6.18 Comparison of topology generators and Skitter topology. (a) Node degree distribution. (b) Average neighbor connectivity. (c) Clustering coefficients. (d) Rich-Club connectivity.

gives the best match for these metrics in agreement with our proposed metric $C_3(\theta)$. The performance of the other topologies is mixed showing that while one topology is able to match one metric it fails to match another. For example, the GLP generator achieves a reasonable match for the node degree distribution but fails to match the average neighbor connectivity. It is interesting to note that BA does not match the rich-club connectivity which is not evident in our metric.

6.8 INTERNET TOPOLOGY EVOLUTION

The WSD produces a mapping from $\Re^{M\times M} \longmapsto \Re^{|K|}$, where $|K| = 71$ bins are used in the examples in this section. However, a 71 dimensional space is still too large to effectively visualize clustering across graphs. In this section, we introduce *multidimensional scaling* (MDS), a technique mapping the WSD into a lower dimension.

Specifically, given C different graphs we seek a mapping from their WSD's into an l dimensional space: $\Re^{C\times|K|} \longmapsto \Re^{C\times l}$ where $l << |K|$. Typically $l = 2$ or 3 makes visual inspection most straightforward. Note that the methods used are parameter-free and so a *natural* clustering of the data is sought, as opposed to a supervised method which applies a mapping learned from training data.

Multidimensional scaling [38] is a technique mapping *distances* between objects into a reduced dimensional space. An intuitive example involves taking the distance matrix commonly shown in the bottom corner of many road maps and using it to reconstruct the map itself. The technique uses *distance* between the graphs here defined in terms of the metric introduced in Equation 6.16, $\Im(G_1, G_2, N)$. First, a dissimilarity matrix, R, is constructed as

$$R_{(i,j)} = \begin{cases} \Im(G_i, G_j, N) & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{6.24}$$

The goal of MDS is to find a set of vectors $Z_1, Z_2, \ldots, Z_{|K|}$ that incrementally approximate the distance in the dissimilarity matrix. Specifically, we wish to minimize the distance between the projected vectors and the original data as

$$C = \min_{Z_1, Z_2, \ldots Z_{|K|}} \sum_{i<j} (\|Z_i - Z_j\| - R_{(i,j)})^2 \tag{6.25}$$

where C is the cost function to be minimized. We then perform the minimization using numerical optimization based on the eigenvector decomposition of R [39]. Typically, the first and second vectors, Z_1 and Z_2, are sufficient to allow visualization of clustering within the data.

Figure 6.19 shows the evolution of the Internet AS topology over time, as observed in the UCLA dataset described in Section 6.4.3. It is difficult to discern any consistent evolution from the raw WSD plots in Figure 6.19a. However, applying the MDS to reduce the dimensionality from 71 to 2 results in Figure 6.19b, in which each point represents the projection of a computed WSD for a given topology, that is, the WSD

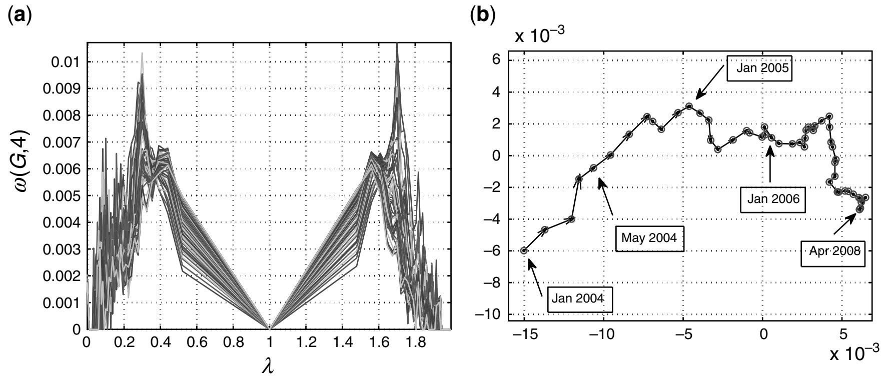


FIGURE 6.19 Structural evolution of the Internet via raw WSD and WSD with MDS applied. (a) Raw WSDs. (b) WSDs after applying MDS.

computed for a given month's observations in the UCLA dataset. Note that the axes are dimensionless, it is not the particular values that are important but the separation of points computed.

Interestingly, plotting with an arrow joining consecutive points, that is, an arrow connects the points for datasets 1 and 2, another connects points for datasets 2 and 3, and so on, shows that the evolution of the WSD for the topology appears to be consistent over time, it represents the "structural walk" of the Internet AS topology observed by the UCLA data. The lack of clustering of points around a center suggests the structure of the Internet is evolving in some way. This evolution is very difficult to see by directly comparing the WSD lines but can easily be observed using this multidimensional scaling technique. This is much more straightforward than the current alternative approach which would involve using a complex set of topological measures to distinguish the different graphs [40]. The reason for this actual evolution is better examined in a different domain; for the interested reader we recommend reading [41]. Here, the aim is merely to show that MDS used in conjunction with WSD can be used to track the structural changes in a network.

6.9 CONCLUSIONS

Comparison of graph structures is a frequently encountered problem across many scientific areas. To perform a meaningful comparison requires the definition of a cost–function that encodes those features of each graph considered important. While the spectrum of a graph encodes a graph's features, the raw spectrum contains too much information to be useful on its own. In this chapter, we have introduced a new metric, the *weighted spectral distribution*, that improves on the raw graph spectrum by discounting those eigenvalues believed to be less significant and noisy, while emphasizing the contribution of those believed to be important and information-rich.

We then showed the use of this cost–function to optimize the selection of parameter values for the subject of Internet topology generation. The cost–function defined by the weighted graph spectrum was shown to lead to parameter choices that are appropriate in the context of the particular problem domain: Internet topology generation. In particular, we showed that the parameter choices so made are close to the default values and, in for one particular graph-generator (BA), fall within the expected region. In addition, as the metric is formed through summation, it is possible to go further and identify the particular eigenvalues that are responsible for significant differences. Although it is currently difficult to assign specific features to specific eigenvalues, we hope that this will also become a feature of the *weighted spectral distribution* in the future. Finally, we briefly demonstrated a technique for projecting the raw WSD distributions into a lower dimensional space. This makes comparison of different distributions straightforward, as shown by the clear evolution of the Internet's topology viewed through the UCLA dataset.

REFERENCES

1. H. Bunke, Graph matching: theoretical foundations, algorithms, and applications, in *Proceedings of the International Conference on Vision Interface*, pp. 82–88, May 2000.
2. V. Kann, On the approximability of the maximum common subgraph problem, in *Proceedings of 9th Annual Symposium on Theoretical Aspects of Computer Science*, pp. 377–388, 1992.
3. A.J. Seary, W.D. Richards, Spectral methods for analyzing and visualizing networks: an introduction, in *Dynamic Social Network Modeling and Analysis*, National Academic Press, pp. 209–228, 2003.
4. B. Nadler, S. Lafon, R. Coifman, I. Kevrekidis, Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators, in *Neural Information Processing Systems (NIPS)*, 2005.
5. A. Ng, M. Jordan, Y. Weiss, On spectral clustering: analysis and an algorithm, in *Advances in Neural Information Processing Systems 14* (T. Dietterich, S. Becker, Z. Ghahramani, ed.), MIT Press, 2002.
6. A.G. Thomason, Pseudo-random graphs, *Random Graphs '85, North-Holland Mathematical Study*, vol. 144, pp. 307–331, 1987.
7. F.R.K. Chung, R.L. Graham, R.M. Wilson, Quasi-random graphs, *Combinatorica*, **9**(4), 345–362 (1989).
8. F.R.K. Chung, *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics)*, American Mathematical Society, 1997.
9. X. Wang, D. Loguinov, Wealth-based evolution model for the internet as-level topology, in *Proceedings of IEEE INFOCOM*, April 2006.
10. D. Fay, H. Haddadi, A.G. Thomason, A.W. Moore, R. Mortier, A. Jamakovic, S. Uhlig, M. Rio, Weighted spectral distribution for internet topology analysis: theory and applications, *IEEE/ACM Trans. Netw. (ToN)*, **18**(1), 164–176 (2010).
11. R. Albert, A.-L. Barabasi, Topology of evolving networks: local events and universality, *Phys. Rev. Lett.* **85**, 2000.
12. H. Haddadi, D. Fay, A. Jamakovic, O. Maennel, A.W. Moore, R. Mortier, M. Rio, S. Uhlig, Beyond node degree: evaluating AS topology models. Technical Report UCAM-CL-TR-725, University of Cambridge, Computer Laboratory, July 2008.
13. H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, W. Willinger, Network topology generators: degree-based vs. structural, in *Proceedings of ACM SIGCOMM 2002*, Pittsburgh, PA, pp. 147–159, 2002.
14. P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, K.C. Claffy, A. Vahdat, The Internet AS-level topology: three data sources and one definitive metric, *SIGCOMM Compu. Commun. Rev.* **36**(1), 17–26 (2006).
15. M.E.J. Newman, Assortative mixing in networks, *Phys. Rev. Lett.* **89**(20), 871–898 (2002).
16. V. Colizza, A. Flammini, M.A. Serrano, A. Vespignani, Detecting rich-club ordering in complex networks, *Nat. Phys.* **2**(2), 110–115 (2006).
17. S. Zhou, Characterising and modelling the Internet topology, the rich-club phenomenon and the PFP model, *BT Technol. J.* **24** (2006).

18. P. Holme, B.J. Kim, C.N. Yoon, S.K. Han, Attack vulnerability of complex networks. *Phys. Rev. E*, **65**(5), 298–305 (2002).
19. M. Baur, U. Brandes, M. Gaertler, D. Wagner, Drawing the AS graph in 2.5 dimensions, in *Graph Drawing* (J. Pach, ed.), Springer, New York, pp. 43–48, 2004.
20. D.R. Wood, An algorithm for finding a maximum clique in a graph, *Oper. Res. Lett.* **21**(7), 211–217 (1997).
21. B.M. Waxman, Routing of multipoint connections, *IEEE J. Selected Areas Commun. (JSAC)*, **6**(9), 1617–1622 (1988).
22. P. Erdös, A. Rényi, On random graphs, in *Mathematical Institute Hungarian Academy, 196*, London, 1985.
23. A.L. Barabasi, R. Albert, Emergence of scaling in random networks, *Science* **286**(5439), 509–512 (1999).
24. M. Faloutsos, P. Faloutsos, C. Faloutsos, On power–law relationships of the Internet topology, in *Proceedings of ACM SIGCOMM 1999*, 1999.
25. T. Bu, D. Towsley, On distinguishing between Internet power–law topology generators, in *Proceedings of IEEE Infocom 2002*, June 2002.
26. J. Winick, S. Jamin, Inet-3.0: internet topology generator, Technical Report CSE-TR-456-02, University of Michigan Technical Report CSE-TR-456-02, 2002.
27. E.W. Zegura, K.L. Calvert, M.J. Donahoo, A quantitative comparison of graph-based models for Internet topology, *IEEE/ACM Trans. Netw. (TON)* **5**(6), 770–783 (1997).
28. A. Medina, A. Lakhina, I. Matta, J. Byers, BRITE: an approach to universal topology generation, in *IEEE MASCOTS*, Cincinnati, OH, USA, pp. 346–353, August 2001.
29. A. Feldmann, O. Maennel, Z.M. Mao, A. Berger, B. Maggs, Locating Internet routing instabilities, in *Proceedings of ACM SIGCOMM 2004*, 2004.
30. Z.M. Mao, J. Rexford, J. Wang, R.H. Katz, Towards an accurate AS-level traceroute tool, in *Proceedings of ACM SIGCOMM 2003*, Karlsruhe, Germany, pp. 365–378, 2003.
31. S. Zhou, G.-Q. Zhang, G.-Q. Zhang, Chinese Internet AS-level topology, *IET Commun.* **1**(2), 209–214 (2007).
32. R. Oliveira, B. Zhang, L. Zhang, Observing the evolution of Internet AS topology, in *Proceedings of ACM SIGCOMM 2007*, Kyoto, Japan, August 2007.
33. P. Mahadevan, D. Krioukov, K. Fall, A. Vahdat, Systematic topology analysis and generation using degree correlations, in *Proceedings of ACM SIGCOMM 2006*, pp. 135–146, Pisa, Italy, 2006.
34. H. Haddadi, D. Fay, S. Uhlig, A. Moore, R. Mortier, A. Jamakovic, M. Rio, Tuning topology generators using spectral distributions, in *Lecture Notes in Computer Science*, SPEC International Performance Evaluation Workshop, Springer Darmstadt, Germany, vol. 5119, 2008.
35. A. Banerjee, J. Jost, Spectral plot properties: towards a qualitative classification of networks, in *European Conference on Complex Systems*, October 2007.
36. J.A. Nelder, R. Mead, A simplex method for function minimization, *Comput. J.* **7**, 308–313 (1965).
37. J.E. Dennis, D.J. Woods, Optimization in microcomputers: the Nelder–Meade simplex algorithm, in *New Computing Environments: Microcomputers in Large-Scale Computing* (A. Wouk, ed.), SIAM, pp. 116–122, 1987.

38. T. Cox, M. Cox, *Multidimensional Scaling*, Chapman and Hall, 1994.
39. G.A.F. Seber, *Multivariate Observations*, John Wiley & Sons, 1984.
40. M. Iliofotou, M. Faloutsos, M. Mitzenmacher, Exploiting dynamicity in graph-based traffic analysis: techniques and applications, in *ACM CoNEXT*, 2009.
41. H. Haddadi, D. Fay, S. Uhlig, A. Moore, R. Mortier, A. Jamakovic, Mixing biases: structural changes in the AS topology evolution, in *Proceedings of the 2nd Traffic Monitoring and Analysis (TMA) Workshop*, Zurich, Switzerland, April 2010.

7

THE STRUCTURE OF AN EVOLVING RANDOM BIPARTITE GRAPH

REINHARD KUTZELNIGG

7.1 INTRODUCTION

In the late 1960s of the last century, the theory of random graphs was developed by Erdös and Rényi [1,2]. Most commonly studied in the literature are the $\mathcal{G}(n, p)$ and the $\mathcal{G}(n, M)$ model. The first consists of all simple graphs possessing n vertices, such that each of the $\binom{n}{2}$ possible edges is chosen independently with probability p. In contrast, by using the $\mathcal{G}(n, M)$ model, a member of the set of all simple graphs consisting of n nodes and M edges is selected, such that each graph is chosen with the same probability. Despite this different definition, these models are closely related. A lot of analysis has been done to address this topic, see, for example, Refs. [1] or [2] for further information.

In this chapter, we consider generalizations of the $\mathcal{G}(n, M)$ model. More precisely, we admit the occurrence of multiple edges and loops. Furthermore, we define $\mathcal{G}(m_1, m_2, n)$, a similar model of bipartite random graphs. To be precise, we deal with graphs possessing two kinds of labeled vertex sets, say V_1 and V_2, where $|V_1| = m_1$ and $|V_2| = m_2$ hold. Starting with an empty graph, we consider a growth process. To insert an edge, we select a node of each type uniformly at random and connect the obtained vertices. Additionally, the ith inserted edge is labeled by i. We repeat these steps, until a total of n edges is generated. Note that multiple edges may occur during this process.

Let us consider the connected components of our bipartite graph. The basic type of component is a tree, that is, a connected but acyclic graph. Furthermore, an

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

isolated node can be considered to build a tree of minimal size. Thus, initially our graph contains tree components only. Whenever an edge is inserted, we hence usually connect two trees to build one new tree of increased size. However, as the components grow, the probability increases that the two chosen nodes already belong to the same tree. In the latter situation, a cycle is created and thus a cyclic component evolves. Furthermore, it might happen after some time, that either two cyclic components are connected, or that a further cycle in an already cyclic component is created. This is the first time that a complex component evolves, that is, a component where the number of edges exceeds the number of vertices. However, as we will see later on, this is quite unlikely to happen, conditioned on the property that the number of edges does not exceed a certain limit. A lot of results concerning the structure of these sparse graphs can be deduced, we present them in the following section.

For nonbipartite graphs, it is well known [3] that there is a high probability that a single giant component evolves, containing most of the vertices. A similar behavior is conjectured for the bipartite case too [4]. Unfortunately, the functions involved in the bipartite case are much more difficult to handle, and no detailed results are known so far.

The bipartite graphs considered here are closely related to a hash table data structure called Cuckoo hashing [5]. Thus, most of the results presented here were originally obtained investigating this algorithm and its variations, see Refs. [6–12]. However, the structure of these graphs is of interest for itself, see for example, Ref. [4] and Refs. [3,13] for nonbipartite graphs.

The remainder of this chapter is organized as follows. First, we present our theorems covering the component structure. For convenience of the reader, the proofs of these results are split up into three parts and are presented in different sections. We begin with a section introducing the tool of generating functions and deduce the functions that enable us to count the graphs considered here. Further, we proceed showing how asymptotic expansions can be obtained by using a saddle point method applied to the previously introduced generating functions. Based on these preparatory works, we put things together and complete the proofs. Finally, we present empirical data to both justify and extend our analysis.

7.2 THE STRUCTURE OF A SPARSE BIPARTITE GRAPH

We start defining the parameters of the graphs considered in this section. The basic case of a bipartite graph is the symmetric one, that is, both sets of vertices have equal cardinality, say $m = m_1 = m_2$. Further, the number of edges is denoted by n. Note that all results provided here are asymptotic approximations. Thus, we somehow fix the ratio of n/m and consider what happens as m turns to infinity. Of course, n has to be an integer number, hence we set it equal to the largest integer that is less than or equal to our aspired target. For instance, if we want to achieve a "constant" ratio of $1 - \varepsilon$, we define $n = \lfloor (1 - \varepsilon)m \rfloor$.

Furthermore, we are also interested in the asymmetric situation, that is, $m_1 \neq m_2$. We may assume without loss of generality, that $m_1 > m_2$ holds. In order to obtain comparable results, we are using the same parameter m as above. Note that a symmetric graph contains in total $2m$ nodes, hence we assure that $m_1 + m_2 = 2m$ is satisfied. This is done by choosing a constant $c \in (0, 1)$ and defining $m_1 = \lfloor m(1 + c) \rfloor$, respectively $m_2 = 2m - m_1$. Thus, we yield $m_1/m_2 = (1 + c)/(1 - c)\left(1 + \mathcal{O}(m^{-1})\right)$ and can obtain any desired constant ratio in the limit. Note that we do not consider further settings, for instance, assuming constant m_2 or sublinear growth of this parameter. This is based on the fact that we see no practical interest in these settings. Moreover our methods could not be easily applied, because of the completely different component structure.

Additionally, we provide results concerning nonbipartite graphs. Again, we assume that the graph contains $2m$ nodes and n edges, where the latter value is defined as above. Thus, we can point out similarities and differences.

To simplify the notation, we assign the following numbers to the cases discussed above:

1. Symmetric bipartite graphs.
2. Asymmetric bipartite graphs, $c \in (0, 1)$ holds.
3. Ordinary, nonbipartite graphs.

For each result, the parameters may depend on the considered case. Thus, we use an index to provide the corresponding number. Furthermore, note that the asymmetry factor c is defined in the asymmetric bipartite case only, hence it does not influence cases 1 and 3 and can thus be omitted in these situations. Furthermore, we consider two different setups:

A. Sparse graph, $\varepsilon \in (0, 1)$ respectively $\varepsilon \in (1 - \sqrt{1 - c^2}, 1)$ is fixed and n, m_1, and m_2 defined as above.
B. "Critical case," n equals m.

First, we consider the probability that all components are either trees or unicyclic, that is, no complex component occurs. Further, we consider the likelihood that a graph possesses a complex part having even more cycles. For symmetric bipartite and usual graphs, it is shown that these probabilities tend to zero, under the conditions of Setup A. However, there is a phase transition if this limit is exceeded, compare with Setup B. We infer a similar behavior considering the asymmetric case, but the critical ratio decreases as the asymmetry increases. In particular, the following results hold:

Theorem 7.1 (Setup A; Cases 1, 2, and 3) [7,10] *The probability that no complex component occurs is given by*

$$1 - \frac{p_i(\varepsilon, c)}{m} + \mathcal{O}\left(\frac{1}{m^2}\right).$$

(Setup A; Cases 1 and 3) [8,11] *The probability that the complex part of the graph contains at most r more edges than nodes is given by*

$$1 - \frac{q_i(\varepsilon, r)}{m^{r+1}} + \mathcal{O}\left(\frac{1}{m^{r+2}}\right).$$

(Setup B; Cases 1 and 3) [3,7] *The probability, that no complex component occurs is given by*

$$\sqrt{\frac{2}{3}} + o(1).$$

We want to emphasize that our approach allows us to compute the given coefficients (and even more detailed expansions) explicitly. For instance, we get

$$p_1(\varepsilon) = \frac{(2\varepsilon^2 - 5\varepsilon + 5)(1-\varepsilon)^3}{12(2-\varepsilon)^2\varepsilon^3}, \quad p_3(\varepsilon) = \frac{(5-2\varepsilon)(1-\varepsilon)^2}{48\varepsilon^3}$$

and,

$$p_2(\varepsilon, c) = \frac{(1-\varepsilon)^3(10 - 2\varepsilon^3 + 9\varepsilon^2 - 3c^2\varepsilon^2 + 9\varepsilon c^2 - 15\varepsilon + 2c^4 - 10c^2)}{12(2\varepsilon - \varepsilon^2 - c^2)^3(c^2-1)}.$$

Note that these results provide practical suitable estimates for sufficiently large m, see Section 7.6 for empirical data. Further, we may compare the individual coefficients. Thus, we yield $p_1(\varepsilon) \le p_3(\varepsilon)$ and $p_1(\varepsilon) \le p_2(\varepsilon, c)$ if ε and c satisfy the required properties. We conclude that symmetric bipartite graphs are less likely to contain complex components under the considered properties.

Several related results are given in the literature. For example, Lemma 2.1 of Ref. [14] treats Case 1 under Setup A, but it does not provide an asymptotic approximation. The nonbipartite case is considered in Ref. [3]. We proceed considering tree components:

Theorem 7.2 (Setup A; Cases 1 and 3) [7] *The number of tree components $T(k, \varepsilon)$ with k vertices satisfies a central limit theorem of the form*

$$\frac{T(k,\varepsilon) - \mu(k,\varepsilon)\, m}{\sqrt{\sigma^2(k,\varepsilon)\, m}} \to N(0,1),$$

where $N(0,1)$ is a normal random variable,

$$\mu(k,\varepsilon) = 2\frac{k^{k-2}(1-\varepsilon)^{k-1}e^{k(\varepsilon-1)}}{k!}$$

and

$$\sigma^2(k,\varepsilon) = \mu(k,\varepsilon) - \frac{2e^{2k(\varepsilon-1)}k^{2k-4}(1-\varepsilon)^{2k-3}(k^2\varepsilon^2 + k^2\varepsilon - 4k\varepsilon + 2)}{(k!)^2}.$$

Further, mean and variance are asymptotically given by $\mathbb{E}\,T(k,\varepsilon) = \mu(k,\varepsilon)\,m + \mathcal{O}(1)$ *and* $\mathbb{V}\text{ar}\,T(k,\varepsilon) = \sigma^2(k,\varepsilon)\,m + \mathcal{O}(1)$ *as m tends to infinity, respectively.*
(Setup A; Case 2) [10] *Let* (x_0, y_0) *be defined by*

$$x_0 = \frac{1-\varepsilon}{1-c}\exp\left(-\frac{1-\varepsilon}{1+c}\right) \quad \textit{and} \quad y_0 = \frac{1-\varepsilon}{1+c}\exp\left(-\frac{1-\varepsilon}{1-c}\right).$$

Then, the number of tree components with k vertices possesses asymptotically the mean

$$m\frac{(1-c^2)}{1-\varepsilon}\sum_{l=0}^{k} l^{k-l-1}(k-l)^{l-1}\frac{x_0^l y_0^{k-l}}{l!(k-l)!} + \mathcal{O}(1).$$

Moreover, a similar formula for the variance can be found, see Ref. [10].

It is surprising that we obtain exactly the same result, both in Case 1 as well as in Case 3. However, asymmetric bipartite graphs behave differently. Note that the number of isolated vertices (i.e., trees of size one) increases, as the asymmetry increases. As a further consequence, we obtain a decreased number of trees of size two, but this trend is not unique. For instance, the number of trees possessing five nodes increases with rising asymmetry. Finally, we present two theorems concerning cycles and cyclic components.

Theorem 7.3 (Setup A; Cases 1, 2, and 3) [7,10] *The number of cyclic components with cycle length* $2k$ *(respectively k in Case 3) has in limit a Poisson distribution* $Po(\lambda_i(k,\varepsilon,c))$*. Further, the number of cycles has in limit a Poisson distribution* $Po(\tilde{\lambda}_i(\varepsilon,c))$*, too. All parameters are given in Table 7.1.*

TABLE 7.1 Parameters of Theorems 7.3 and 7.4

	Case 1 (Symmetric Bipartite)	Case 2 (Asymmetric Bipartite)	Case 3 (Nonbipartite)
$\lambda_i(k,\varepsilon,c)$	$\frac{1}{2k}(1-\varepsilon)^{2k}$	$\frac{1}{2k}\left(\frac{(1-\varepsilon)^2}{1-c^2}\right)^{2k}$	$\frac{1}{2k}(1-\varepsilon)^k$
$\tilde{\lambda}_i(\varepsilon,c)$	$-\frac{1}{2}\log\left(1-(1-\varepsilon)^2\right)$	$-\frac{1}{2}\log\left(1-\frac{(1-\varepsilon)^2}{1-c^2}\right)$	$-\frac{1}{2}\log\varepsilon$
$\phi_i(s)$	$\sqrt{\frac{1-(1-\varepsilon)^2}{1-e^{2is}(1-\varepsilon)^2}}$	$\sqrt{\frac{1-\frac{(1-\varepsilon)^2}{1-c^2}}{1-e^{2is}\frac{(1-\varepsilon)^2}{1-c^2}}}$	$\sqrt{\frac{\varepsilon}{1-e^{is}(1-\varepsilon)}}$
$\lim_{m\to\infty}\mathbb{E}\,V(\varepsilon,c)$	$\frac{(1-\varepsilon)^2}{1-(1-\varepsilon)^2}$	$\frac{(1-\varepsilon)^2}{2\varepsilon-\varepsilon^2-c^2}$	$\frac{1-\varepsilon}{2\varepsilon}$
$\lim_{m\to\infty}\mathbb{V}\text{ar}\,V(\varepsilon,c)$	$\frac{2(1-\varepsilon)^2}{\left(1-(1-\varepsilon)^2\right)^2}$	$\frac{2(1-\varepsilon)^2(1-c^2)}{\left(2\varepsilon-\varepsilon^2-c^2\right)^2}$	$\frac{(1-\varepsilon)}{2\varepsilon^2}$
$\lim_{m\to\infty}\mathbb{E}\,U(\varepsilon,c)$	$\frac{(1-\varepsilon)^2}{\varepsilon\left(1-(1-\varepsilon)^2\right)}$	$\frac{(1-\varepsilon)^2(2-\varepsilon-c^2)}{(2\varepsilon-\varepsilon^2-c^2)^2}$	$\frac{(1-\varepsilon)}{2\varepsilon^2}$
$\lim_{m\to\infty}\mathbb{V}\text{ar}\,U(\varepsilon,c)$	$\frac{(1-\varepsilon)^2(\varepsilon^2-3\varepsilon+4)}{\varepsilon^2\left(1-(1-\varepsilon)^2\right)^2}$	see Ref. [10]	$\frac{(1-\varepsilon)(2-\varepsilon)}{2\varepsilon^4}$

Note that these parameters are somehow related, that is,

$$\tilde{\lambda}_i(\varepsilon, c) = \sum_{k\geq 1} \lambda_i(k, \varepsilon, c)$$

holds.

Theorem 7.4 (Setup A; Cases 1, 2, and 3) [7,10] *The number of vertices contained in cycles $V(\varepsilon, c)$ has a limiting distribution with characteristic function $\phi_i(s)$, as given in Table 7.1. Further, denote the number of vertices contained in unicyclic components by $U(\varepsilon, c)$. Then, mean and variance of $U(\varepsilon, c)$ have in limit the values provided in Table 7.1.*

We notice that asymmetry leads to an increased number of cycles and nodes in cyclic components. Furthermore, both parameters increase if we consider an ordinary graph instead of the symmetric bipartite version.

To prove the results claimed above, we first introduce notations and techniques that are required to deduce the results. This is done in the following sections. The formal proofs are finally given in Section 7.5. However, for the sake of shortness, we consider the symmetric bipartite version only. Note that the further results can be deduced in a similar way, but are either much easier to prove (ordinary graphs) or else very long without providing any further insights. Details omitted here can be found in Refs. [7,10–12].

7.3 ENUMERATING BIPARTITE GRAPHS

In this section, we review the generating functions that enable us to count the number of graphs of certain type respectively to calculate asymptotic approximations of these numbers. Note that similar results for nonbipartite graphs are well known, see, for example, Refs. [3,13]. In contrast, only few results concerning bipartite graphs can be found in the literature. However, trees and unicyclic components have been studied recently, see also Ref. [15].

Let $\mathcal{F}$ denote a family of bipartite graphs. Then, the corresponding trivariate generating function is the formal power series

$$F(x, y, v) = \sum_{\mathcal{G}\in\mathcal{F}} \frac{x^{m_1(\mathcal{G})}}{m_1(\mathcal{G})!} \frac{y^{m_2(\mathcal{G})}}{m_2(\mathcal{G})!} \frac{v^{n(\mathcal{G})}}{n(\mathcal{G})!},$$

where $m_1(\mathcal{G})$ and $m_2(\mathcal{G})$ denote the number of nodes of first and second kind and $n(\mathcal{G})$ denotes the number of edges of $\mathcal{G}$.

Further, we make use of the notation $[x^m]A(x)$ to extract the mth coefficient of a power series $A(x)$, that means

$$[x^m]A(x) = [x^m]\sum_{k\geq 0} a_k x^k = a_k.$$

We start counting all bipartite graphs without restrictions to the type of their components. Let $G_{m_1,m_2,n}$ denote the set of all vertex and edge-labeled bipartite multigraphs (V_1, V_2, E) with $|V_1| = m_1$, $|V_2| = m_2$, and $|E| = n$. By definition, it is clear that the number of all graphs of the family $G_{m_1,m_2,n}$ equals

$$\#G_{m_1,m_2,n} = m_1^n m_2^n. \tag{7.1}$$

Next, let $G^\circ_{m_1,m_2,n}$ denote those graphs in $G_{m_1,m_2,n}$ without complex components, that is, all components are either trees or unicyclic. Further,

$$g^\circ(x, y, v) = \sum_{m_1,m_2,n} \#G^\circ_{m_1,m_2,n} \frac{x^{m_1}}{m_1!} \frac{y^{m_2}}{m_2!} \frac{v^n}{n!}$$

denotes the corresponding generating function. Our next goal is to describe this generating function. For this purpose, we will first consider bipartite trees.

We call a tree bipartite if the vertices are partitioned into two classes V_1 ("black" vertices) and V_2 ("white" vertices) such that no vertex has a neighbor of the same class. They are called labeled if the vertices of type 1, that is vertices in V_1, are labeled by $1, 2, \ldots, |V_1|$ and the vertices of type 2 are labeled independently by $1, 2, \ldots, |V_2|$.

Let $t_1(x, y, v)$ and $t_2(x, y, v)$ denote the generating function of rooted bipartite trees, where the root is contained in V_1 and V_2. Furthermore, we denote the generating function of unrooted bipartite trees by $\tilde{t}(x, y, v)$. However, it is usually simpler to consider bivariate generating functions that do not take the edges into account. Since a tree possesses exactly one more node than edges, the bivariate generating function $t_1(x, y)$ corresponding to $t_1(x, y, v)$ satisfies

$$t_1(x, y) = \frac{1}{v} t(xv, yv).$$

We thus slightly abuse notation by donating bivariate and trivariate generating functions by the same letter, however, the correct interpretation should be obvious from the context.

Lemma 7.1 [7,15] *The generating functions $t_1(x, y)$, $t_2(x, y)$, and $\tilde{t}(x, y)$ are given by*

$$t_1(x, y) = xe^{t_2(x,y)}, \quad t_2(x, y) = ye^{t_1(x,y)} \tag{7.2}$$

and by

$$\tilde{t}(x, y) = t_1(x, y) + t_2(x, y) - t_1(x, y)t_2(x, y). \tag{7.3}$$

Proof: The functional equations (Eq. 7.2) are obvious by their recursive description. To prove Equation 7.3, consider a rooted tree, possessing a black root labeled by 1, as an unrooted tree. Next, examine an unordered pair (t_1, t_2) of rooted bipartite trees of different kind, and join the roots by an edge. If the black vertex labeled by 1 is contained in t_1, consider the root of t_2 as new root, and we obtain a tree possessing a white root and at least one black vertex. Otherwise, consider the root of t_1 as new root, and we obtain a tree with a black vertex not labeled by 1. ■

Note that $t_1(x, y) = t_2(y, x)$ holds and that $t_1(x, x)$ equals the usual tree function $t(x) = \sum_{n\geq 1} n^{n-1}x^n/n!$ that is given by $t(x) = xe^{t(x)}$, see Ref. [16]. Thus, $t_1(x, y)$ and $t_2(x, y)$ are surely analytic functions for $|x| < e^{-1}$ and $|y| < e^{-1}$. This is due to the fact that the radius of convergence of $t(x)$ equals $1/e$.

With the help of these functions, we can describe the generating function $g^\circ(x, y, v)$.

Lemma 7.2 **[7]** *The generating function* $g^\circ(x, y, v)$ *is given by*

$$g^\circ(x, y, v) = \frac{e^{\frac{1}{v}\tilde{t}(xv,yv)}}{\sqrt{1 - t_1(xv, yv)t_2(xv, yv)}}.$$

Proof: We have to count graphs where each component is either an unrooted tree (that is counted by $\tilde{t}(x, y)$) or a graph with exactly one cycle.

Of course, a cycle has to have an even number of vertices (say $2k$), where k vertices are black and the other k vertices are white. A cyclic vertex of black color can be considered as the root of a rooted tree of type 1 and similarly, a white cyclic vertex can be considered as the root of a rooted tree of type 2. Note that we have to divide the product of the generating functions $t_1(x, y)^k t_2(x, y)^k$ by $2k$ to account for cyclic order and change of orientation. Hence, the corresponding generating functions of an unicyclic graph with $2k$ cyclic points is given by

$$\frac{1}{2k} t_1(x, y)^k t_2(x, y)^k.$$

Consequently, the generating function of a connected graph with exactly one cycle is given by

$$c(x, y) = \sum_{k\geq 1} \frac{1}{2k} t_1(x, y)^k t_2(x, y)^k = \frac{1}{2} \log \frac{1}{1 - t_1(x, y)t_2(x, y)}.$$

Since a cyclic component of size $m_1 + m_2$ has exactly the same number of edges as nodes, and since there are $(m_1 + m_2)!$ possible labels, the corresponding generating function that takes the number of edges into account in given by $c(xv, yv)$.

Similarly, a tree of size $m_1 + m_2$ has exactly $n = m_1 + m_2 - 1$ edges. Consequently the generating function $\tilde{t}(xv, yv)/v$ corresponds to a bipartite unrooted tree.

Finally the generating function $g^\circ(x, y, v)$ is given by

$$g^\circ(x, y, v) = e^{\frac{1}{v}\tilde{t}(xv,yv)+c(xv,yv)} = \frac{e^{\frac{1}{v}\tilde{t}(xv,yv)}}{\sqrt{1 - t_1(xv, yv)t_2(xv, yv)}},$$

which completes the proof of the lemma. ■

Note that each of these graphs contains exactly $m_1 + m_2 - n$ trees, thus we yield the bivariate version

$$g^{\circ}(x,y) = n!\frac{\tilde{t}(x,y)^{m_1+m_2-n}}{(m_1+m_2-n)!}\frac{1}{\sqrt{1-t_1(x,y)t_2(x,y)}}. \tag{7.4}$$

To study more general graphs, it is convenient to consider families of graphs according to the excess of edges over vertices in connected components. For instance, trees and unicyclic components have excess -1 and 0. Further, a component is complex, if it has positive excess. We proceed as in Ref. [3], but we have to adopt the calculation to the present situation. Thereby, we make use of the following shortened notation:

Definition 7.1 *Let ϑ_x denote the differential operator $x\frac{\partial}{\partial x}$, that corresponds to marking a vertex of first kind. Similarly, we define the operators ϑ_y and ϑ_v for marking a node of second kind resp. an edge.*

Hence, we yield

Lemma 7.3 **[11]** *Let $E_r(x,y)$ denote the generating function of bipartite graphs consisting of complex components only and having exactly r more edges than vertices. These functions satisfy the partial differential recurrence*

$$\left(r + (1-t_2)\vartheta_x + (1-t_1)\vartheta_y\right) E_r = e^{-C}\vartheta_x\vartheta_y e^{C} E_{r-1}.$$

Moreover, $E_0 = 1$ holds, since only the empty graph is complex and has excess 0.

Note that a complex graph possessing excess 1 must consist of a single bicyclic component, thus E_0 provides the corresponding generating function. We do not provide a proof of this lemma here, since it is rather technical, see Ref. [11] for details.

Solving the recursion of Lemma 7.3 in general seems to be out of reach, but using a computer algebra system, it is quite easy to get some results. In particular, the solutions exhibit a certain pattern, hence it is possible to write down an ansatz and compute the coefficients. Thus, we get for instance

$$E_1 = \frac{t_1t_2(4+3t_2+3t_1+6t_1t_2+2t_1t_2^2+2t_1^2t_2)}{24(1-t_1t_2)^3},$$

and

$$\begin{aligned}E_2 = \frac{t_1t_2}{1152(1-t_1t_2)^6}\big(&720t_1^2t_2+24t_1^2+652t_1^3t_2^2+72t_1^4t_2^3+156t_1^4t_2^2+156t_2^4t_1^2\\&+48+24t_2^2+201t_1^3t_2+4t_2^5t_1^3+688t_1t_2+4t_1^5t_2^3+201t_1t_2^3+8t_1^4t_2^4\\&+348t_1^3t_2^3+72t_2^4t_1^3+652t_1^2t_2^3+1218t_1^2t_2^2+720t_1t_2^2+96t_2+96t_1\big).\end{aligned}$$

7.4 ASYMPTOTIC EXPANSION VIA THE SADDLE POINT METHOD

This section provides the tool that is used to infer asymptotic expansions. The result can be obtained by using a double saddle point approach, see, for example, Refs. [7,16–19] for details concerning this method. Note that further coefficients of the asymptotic expansions can be calculated in the same way, but the expressions are so complicated that it does not make sense to provide them outside a computer algebra system. A maple worksheet is available on request from the author.

Lemma 7.4 **[7]** *Let $f(x, y)$ and $g(x, y)$ be analytic functions locally around $(x, y) = (0, 0)$ such that all coefficients $[x^{m_1} y^{m_2}]f(x, y)$ and $[x^{m_1} y^{m_2}]g(x, y)$ are non-negative and that there exists M such that all indices (m_1, m_2) with $m_1, m_2 \geq M$ can be represented as a finite linear combination of the set $\{(m_1, m_2) | [x^{m_1} y^{m_2}]f(x, y) > 0\}$ with positive integers as coefficients.*

Let R_1 and R_2 be compact intervals of the positive real line such that $R = R_1 \times R_2$ is contained in the regions of convergence of $f(x, y)$ and $g(x, y)$. Furthermore set

$$S = \left\{ \left(\frac{x}{f(x, y)} \frac{\partial}{\partial x} f(x, y), \frac{y}{f(x, y)} \frac{\partial}{\partial y} f(x, y) \right) : (x, y) \in R \right\}.$$

Then, we have

$$[x^{m_1} y^{m_2}]g(x, y)f(x, y)^k = \frac{g(x_0, y_0)f(x_0, y_0)^k}{2\pi x_0^{m_1} y_0^{m_2} k\sqrt{\Delta}} \left(1 + \frac{H}{24\Delta^3} \frac{1}{k} + \mathcal{O}\left(\frac{1}{k^2}\right) \right),$$

uniformly for $(m_1/k, m_2/k) \in S$, where x_0 and y_0 are uniquely determined by

$$\frac{m_1}{k} = \frac{x_0}{f(x_0, y_0)} \left[\frac{\partial}{\partial x} f(x, y) \right]_{(x_0, y_0)}, \qquad \frac{m_2}{k} = \frac{y_0}{f(x_0, y_0)} \left[\frac{\partial}{\partial y} f(x, y) \right]_{(x_0, y_0)}$$

and the constants Δ and H are given in the following way:

Let κ_{ij} and $\overline{\kappa}_{ij}$ be the cummulants

$$\kappa_{ij} = \left[\frac{\partial^i}{\partial u^i} \frac{\partial^j}{\partial v^j} \log f(x_0 e^u, y_0 e^v) \right]_{(0,0)},$$

$$\overline{\kappa}_{ij} = \left[\frac{\partial^i}{\partial u^i} \frac{\partial^j}{\partial v^j} \log g(x_0 e^u, y_0 e^v) \right]_{(0,0)}.$$

Then, $\Delta = \kappa_{20}\kappa_{02} - \kappa_{11}^2$ holds and H is given by

$$H = \alpha + \beta + \hat{\beta} + \gamma\overline{\kappa}_{10} + \hat{\gamma}\overline{\kappa}_{01} + \delta\overline{\kappa}_{10}\overline{\kappa}_{01} + \eta\overline{\kappa}_{10}^2 + \hat{\eta}\overline{\kappa}_{01}^2 + 4\eta\overline{\kappa}_{20} + 4\hat{\eta}\overline{\kappa}_{02} + 4\delta\overline{\kappa}_{11},$$

where

$$\begin{aligned}
\alpha &= 54\kappa_{21}\kappa_{11}\kappa_{12}\kappa_{20}\kappa_{02} + 6\kappa_{22}\kappa_{20}\kappa_{02}\kappa_{11}^2 - 12\kappa_{22}\kappa_{11}^4 + 4\kappa_{03}\kappa_{11}^3\kappa_{30} \\
&\quad + 36\kappa_{21}\kappa_{11}^3\kappa_{12} + 6\kappa_{22}\kappa_{20}^2\kappa_{02}^2 + 6\kappa_{03}\kappa_{11}\kappa_{30}\kappa_{20}\kappa_{02}, \\
\beta &= -5\kappa_{02}^3\kappa_{30}^2 + 30\kappa_{02}^2\kappa_{30}\kappa_{11}\kappa_{21} - 24\kappa_{02}\kappa_{30}\kappa_{12}\kappa_{11}^2 - 6\kappa_{02}^2\kappa_{30}\kappa_{12}\kappa_{20} \\
&\quad - 12\kappa_{11}\kappa_{02}^2\kappa_{31}\kappa_{20} - 36\kappa_{02}\kappa_{21}^2\kappa_{11}^2 - 9\kappa_{02}^2\kappa_{21}^2\kappa_{20} + 3\kappa_{02}^3\kappa_{40}\kappa_{20} \\
&\quad - 3\kappa_{02}^2\kappa_{40}\kappa_{11}^2 + 12\kappa_{11}^3\kappa_{02}\kappa_{31}, \\
\gamma &= 12\Delta\left(\kappa_{02}^2\kappa_{30} - \kappa_{11}\kappa_{20}\kappa_{03} - 3\kappa_{21}\kappa_{11}\kappa_{02} + \kappa_{12}\kappa_{11}^2 + \kappa_{12}(\kappa_{02}\kappa_{20} + \kappa_{11}^2)\right), \\
\delta &= 24\Delta(\kappa_{11}\kappa_{20}\kappa_{02} - \kappa_{11}^3), \\
\eta &= 12\Delta(\kappa_{02}\kappa_{11}^2 - \kappa_{02}^2\kappa_{20}),
\end{aligned}$$

and $\hat{}$ *indicates to replace all functions of type* κ_{ij} *by* κ_{ji}.

Proof: In our proof, we will use the formula

$$\int_{-\infty}^{\infty} e^{-z^2/2} z^k \,\mathrm{d}z = \begin{cases} 1\cdot 3\cdot 5\ldots(k-1)\sqrt{2\pi} & \text{if } k \text{ is even,} \\ 0 & \text{if } k \text{ is odd.} \end{cases} \tag{7.5}$$

The technical conditions on the coefficients of $f(x, y)$ ensure that the function $f(x_0e^{is}, y_0e^{it})$, if $(s, t) \in [-\pi/2, \pi/2]^2$ holds, has its maximal modulus for $s = t = 0$. Furthermore, it can be seen that the saddle point (x_0, y_0) is unique, because the cummulants of second order are strictly positive (compare with Ref. 19).

We start by applying Cauchy's formula and substitute $x = x_0e^{is}$ and $y = y_0e^{it}$:

$$\begin{aligned}
[x^{m_1}y^{m_2}]g(x, y)f(x, y)^k &= -\frac{1}{4\pi^2}\int_{|x|=x_0}\int_{|y|=y_0} \frac{g(x, y)f(x, y)^k}{x^{m_1+1}y^{m_2+1}}\,\mathrm{d}y\,\mathrm{d}x \\
&= \frac{1}{4\pi^2x_0^{m_1}y_0^{m_2}}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi} g\left(x_0e^{is}, y_0e^{it}\right) f\left(x_0e^{is}, y_0e^{it}\right)^k e^{-m_1is-m_2it}\,\mathrm{d}t\,\mathrm{d}s.
\end{aligned}$$

The contribution of the integral taken over the range $I = \big([-\pi, \pi] \times [-\pi, \pi]\big)\backslash \big([-\alpha, \alpha] \times [-\alpha, \alpha]\big)$ is very small compared to the remaining integral, where $\alpha = k^{-1/2+\xi}$ and ξ denotes a real number satisfying $0 < \xi < 1/6$. This can be seen as follows: by continuity we surely have $|f\left(x_0e^{is}, y_0e^{it}\right)| \le f(x_0, y_0) - \delta$ if $|s| \ge \eta$ or $|t| \ge \eta$, where $\delta > 0$ and $\eta > 0$ are chosen appropriately. Furthermore, for $|s| < \eta$ and $|t| < \eta$ we can use a local expansion of the form

$$e^{k\log f\left(x_0e^{is}, y_0e^{it}\right) - m_1is - m_2it} = f\left(x_0, y_0\right)^k e^{-\frac{k}{2}\left(\kappa_{20}s^2 + 2\kappa_{11}st + \kappa_{02}t^2\right) + \mathcal{O}\left(k^{-\frac{1}{2}+3\xi}\right)}$$

to deduce that (for some $c > 0$)

$$\left| \iint_I g\left(x_0 e^{is}, y_0 e^{it}\right) f\left(x_0 e^{is}, y_0 e^{it}\right)^k e^{-m_1 is - m_2 it}\, \mathrm{d}t\, \mathrm{d}s \right| \le 4\pi^2 g(x_0, y_0) f(x_0, y_0)^k e^{-ck^{2\xi}}.$$

Hence, this part of the integral is negligible (as proposed).

Next, we substitute $u = \sqrt{k}s$ and $v = \sqrt{k}t$ and calculate Taylor expansions of the functions $\log f$ and $\log g$. More precisely, we obtain the expansions

$$\begin{aligned} k \log f\left(x_0 e^{i\frac{u}{\sqrt{k}}}, y_0 e^{i\frac{v}{\sqrt{k}}}\right) - m_1 i \frac{u}{\sqrt{k}} - m_2 i \frac{v}{\sqrt{k}} &= k \log f(x_0, y_0) \\ - \frac{1}{2}\left(\kappa_{20}u^2 + 2\kappa_{11}uv + \kappa_{02}v^2\right) &- \frac{i}{6\sqrt{k}}\left(\kappa_{30}u^3 + 3\kappa_{21}u^2v + 3\kappa_{12}uv^2 + \kappa_{03}v^3\right) \\ + \frac{1}{24k}\left(\kappa_{40}u^4 + 4\kappa_{31}u^3v + 6\kappa_{22}u^2v^2 + 4\kappa_{13}uv^3 + \kappa_{04}v^4\right) &+ \mathcal{O}\left(k\alpha^5\right), \end{aligned}$$

and

$$\begin{aligned} &\log g\left(x_0 e^{i\frac{u}{\sqrt{k}}}, y_0 e^{i\frac{v}{\sqrt{k}}}\right) \\ &= \log g(x_0, y_0) + \frac{i}{\sqrt{k}}\left(\overline{\kappa}_{10}u + \overline{\kappa}_{01}v\right) - \frac{1}{2k}\left(\overline{\kappa}_{20}u^2 + 2\overline{\kappa}_{11}uv + \overline{\kappa}_{02}v^2\right) + \mathcal{O}\left(\alpha^3\right) \end{aligned}$$

in the neighborhood of (x_0, y_0). The linear terms vanish due to the choice of the saddle point. By using further expansions in k, we can rewrite the remaining integral in the following way:

$$\begin{aligned} &\frac{1}{4\pi^2 x_0^{m_1} y_0^{m_2}} \int_{-\alpha}^{\alpha}\int_{-\alpha}^{\alpha} g\left(x_0 e^{is}, y_0 e^{it}\right) e^{k \log f\left(x_0 e^{is}, y_0 e^{it}\right) - m_1 is - m_2 it} \mathrm{d}t\, \mathrm{d}s \\ &= \frac{g(x_0, y_0) f(x_0, y_0)^k}{4k\pi^2 x_0^{m_1} y_0^{m_2}} \int_{-\alpha\sqrt{k}}^{\alpha\sqrt{k}}\int_{-\alpha\sqrt{k}}^{\alpha\sqrt{k}} e^{-\frac{1}{2}\left(\kappa_{20}u^2 + 2\kappa_{11}uv + \kappa_{02}v^2\right)} \\ &\quad \times \Bigg(1 - \frac{i}{6\sqrt{k}}\left(\kappa_{30}u^3 + 3\kappa_{21}u^2v + 3\kappa_{12}uv^2 + \kappa_{03}v^3\right) \\ &\qquad + \frac{i}{\sqrt{k}}\left(\overline{\kappa}_{10}u + \overline{\kappa}_{01}v\right) - \frac{1}{72k}\left(\kappa_{30}u^3 + 3\kappa_{21}u^2v + 3\kappa_{12}uv^2 + \kappa_{03}v^3\right)^2 \\ &\qquad + \frac{1}{24k}\left(\kappa_{40}u^4 + 4\kappa_{31}u^3v + 6\kappa_{22}u^2v^2 + 4\kappa_{13}uv^3 + \kappa_{04}v^4\right) \\ &\qquad + \frac{1}{6k}\left(\kappa_{30}u^3 + 3\kappa_{21}u^2v + 3\kappa_{12}uv^2 + \kappa_{03}v^3\right)\left(\overline{\kappa}_{10}u + \overline{\kappa}_{01}v\right) \\ &\qquad - \frac{1}{2k}\left(\overline{\kappa}_{10}u + \overline{\kappa}_{01}v\right)^2 - \frac{1}{2k}\left(\overline{\kappa}_{20}u^2 + 2\overline{\kappa}_{11}uv + \overline{\kappa}_{02}v^2\right) \\ &\qquad + \mathcal{O}\left(\alpha^9 k^3\right)\Bigg) \mathrm{d}v\, \mathrm{d}u. \end{aligned} \tag{7.6}$$

Without loss of generality, we can assume that $\kappa_{20} \geq \kappa_{02}$. We substitute $u = \sqrt{\kappa_{02}/\Delta}\, a$ and $v = -\kappa_{11}/\sqrt{\kappa_{02}\Delta}\, a + b/\sqrt{\kappa_{02}}$, where $\Delta = \kappa_{20}\kappa_{02} - \kappa_{11}^2$. Hence,

$$\int_{-\alpha\sqrt{k}}^{\alpha\sqrt{k}} \int_{-\alpha\sqrt{k}}^{\alpha\sqrt{k}} e^{-\frac{1}{2}\left(\kappa_{20}u^2+2\kappa_{11}uv+\kappa_{02}v^2\right)} \mathrm{d}v\, \mathrm{d}u = \frac{1}{\sqrt{\Delta}} \int_{-\mu}^{\mu} \int_{-\nu(a)}^{\nu(a)} e^{-\frac{a^2+b^2}{2}} \mathrm{d}b\, \mathrm{d}a,$$

where $\mu = \alpha\sqrt{k\Delta/\kappa_{02}}$ and $\nu(a) = \alpha\sqrt{k\kappa_{02}} + a\kappa_{11}/\sqrt{\Delta}$. Note that for all a satisfying $-\mu \leq a \leq \mu$, the inequality $\nu(a) \geq \nu_{\min} = \alpha\sqrt{k}(\sqrt{\kappa_{02}} - \kappa_{11}/\sqrt{\kappa_{02}})$ is valid. Furthermore the relation $\kappa_{20} \geq \kappa_{02}$ implies that $\sqrt{\kappa_{02}} - \kappa_{11}/\sqrt{\kappa_{02}} > 0$. Consequently, we get (for some constant $0 < c < \frac{1}{2}$)

$$\int_{\nu(a)}^{\infty} e^{-\frac{b^2}{2}} b^l\, \mathrm{d}b \leq \int_{\nu_{\min}}^{\infty} e^{-\frac{b^2}{2}} b^l\, \mathrm{d}b = \mathcal{O}\left(e^{-ck^{2\xi}}\right). \tag{7.7}$$

The last part of Equation 7.6 can be estimated by

$$\begin{aligned}
&\left| \int_{-\alpha\sqrt{k}}^{\alpha\sqrt{k}} \int_{-\alpha\sqrt{k}}^{\alpha\sqrt{k}} e^{-\frac{1}{2}\left(\kappa_{20}u^2+2\kappa_{11}uv-\kappa_{02}v^2\right)} \mathcal{O}\left(k^3\alpha^9\right)\, \mathrm{d}v\, \mathrm{d}u \right| \\
&\quad \leq \frac{\mathcal{O}\left(k^3\alpha^9\right)}{\sqrt{\Delta}} \int_{-\mu}^{\mu} \int_{-\nu(a)}^{\nu(a)} e^{-\frac{a^2+b^2}{2}} \mathrm{d}b\, \mathrm{d}a \leq \mathcal{O}\left(k^3\alpha^9\right) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-\frac{a^2+b^2}{2}} \mathrm{d}b\, \mathrm{d}a \\
&\quad = \mathcal{O}\left(k^{-\frac{3}{2}+9\xi}\right).
\end{aligned}$$

Finally, we introduce the notation

$$\begin{aligned}
I(p,q) &= \int_{-\mu}^{\mu} \int_{-\nu(a)}^{\nu(a)} e^{-\frac{a^2+b^2}{2}} a^p b^q\, \mathrm{d}b\, \mathrm{d}a \\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-\frac{a^2+b^2}{2}} a^p b^q\, \mathrm{d}b\, \mathrm{d}a + O\left(e^{-ck^{2\xi}}\right)
\end{aligned} \tag{7.8}$$

Obviously, $I(p,q) = 0$ if either p or q is odd. Hence, the main term of the integral in Equation 7.6 can be rewritten using the functions $I(p,q)$. It remains to complete these integrals $I(p,q)$ to the range $\mathbb{R}^2$ (see Eq. 7.8) and to calculate them according to Equation 7.5. ■

7.5 PROOFS OF THE MAIN THEOREMS

We provide proofs for the symmetric bipartite case only. Note that the two other cases can be treated in a similar manner. However, nonbipartite graphs are much simpler to handle, because the calculations involve univariate instead of bivariate generating functions. On the other hand, the asymmetric bipartite case can be treated similar to the symmetric one, however, the calculations are even more technical due to the lack of symmetry and thus omitted.

Proof (of Theorem 7.1): We start considering the probability that no complex component occurs during the noncritical phase. The generating function of all such graphs is given by Equation 7.4. What is left, is to extract the coefficient of $x^m y^m$ and to divide this result by Equation 7.1, the corresponding total number of graphs. Recall that $\varepsilon > 0$ is fixed. We proceed considering the sequence of integer pairs $(m, n) = (m, \lfloor (1-\varepsilon)m \rfloor)$. For technical reasons, we also define the ratio

$$\varepsilon' = \varepsilon'_m = \frac{m-n}{m} = 1 - \frac{\lfloor (1-\varepsilon)m \rfloor}{m} = \varepsilon + \mathcal{O}\left(m^{-1}\right)$$

which is always very close to ε. Thus, we may apply Lemma 7.4. It turns out that the saddle point is given by

$$x_0 = y_0 = \frac{n}{m} e^{-\frac{n}{m}} = (1-\varepsilon')e^{\varepsilon'-1} < \frac{1}{e}.$$

Combining the result with Stirling's formula, we thus obtain

$$\#G^{\circ}_{m,m,n} = m^{2n}\left(1 - \frac{1}{m}\frac{(2\varepsilon'^2 - 5\varepsilon' + 5)(1-\varepsilon')^3}{12(2-\varepsilon')^2\varepsilon'^3} + \mathcal{O}\left(\frac{1}{m^2}\right)\right)$$

Finally, we can safely replace ε' by $\varepsilon = \varepsilon' + \mathcal{O}(m^{-1})$ without changing the expansion. All changes go into the error term $\mathcal{O}(m^{-2})$.

Note that the critical case, that is Setup B, is somehow different. Lemma 7.4 can not be directly applied, because the saddle point would coincide with the singularity of the denominator. Thus, additional calculations and a different saddle point approach are required, see Ref. [7] for details.

Finally, we consider the probability that the complex part of a symmetric bipartite graph contains at most r more edges than nodes. To calculate an asymptotic expansion for a certain r, we thus proceed counting all graphs possessing excess exactly equal to s for all s less or equal than r. Such a graph contains $2m - n + s$ unrooted tree components, and a possibly empty set of unicyclic components:

$$\#G^{s}_{m,m,n} = (m!)^2 n! [x^m y^m] \frac{\tilde{t}(x, y)^{2m-n+s}}{(2m+n-s)!} e^{c(x,y)} E_s(x, y).$$

An asymptotic expansion can again be derived by applying Lemma 7.4. In particular, we infer the same saddle point (x_0, y_0) as above. Finally, the probability that the excess equals at most r, is given by

$$\sum_{s=0}^{r} \frac{\#G^{s}_{m,m,n}}{\#G_{m,m,n}}.$$

We performed these calculations for $r \leq 2$ using Maple and obtained in particular

$$q_i(\varepsilon, 1) = \frac{(\varepsilon-1)^4(4\varepsilon^6 - 52\varepsilon^5 + 305\varepsilon^4 - 868\varepsilon^3 + 1358\varepsilon^2 - 1120\varepsilon + 385)}{288(-2+\varepsilon)^4\varepsilon^6}.$$

Furthermore, a general proof of this property without the explicit calculation of an asymptotic approximation is based on the results of Ref. [8], together with the observations of Ref. [11]. ■

Our further results concern the component structure of the graph. The proofs are again based on a generating function approach using Equation 7.4. Further, we introduce an additional variable to "mark" the parameter of interest, see for instance Refs. [16,20–22] for further details of this method.

Again, recall that we fix $\varepsilon > 0$ and suppose that $n = \lfloor(1-\varepsilon)m\rfloor$. We also note that it is sufficient to consider graphs of $G^{\circ}_{m,m,n}$, the set of bipartite graphs without complex components, since all results for $G^{\circ}_{m,m,n}$ hold for unrestricted random bipartite graphs too. This can be easily seen in the following way. Consider a random variable ξ defined on the set $G_{m,m,n}$ (with $n = \lfloor(1-\varepsilon)m\rfloor$ and $\varepsilon > 0$) and ξ' its restriction to $G^{\circ}_{m,m,n}$. Then the corresponding distribution functions by F_ξ and $F_{\xi'}$ satisfy the relation

$$|F_\xi - F_{\xi'}| \le \mathbb{P}(G_{m,m,n} \setminus G^{\circ}_{m,m,n}) = \mathcal{O}(1/m).$$

We proceed considering the cyclic components, since they are easier to handle.

Proof (of Theorem 7.3): In particular, we prove the following facts: The moment generating function of the number of cycles $C_{n,m}$ and the number of cycles of length $2k$ $C_{n,m,k}$ in a graph of $G^{\circ}_{m,m,n}$ (with $n = \lfloor(1-\varepsilon)m\rfloor$ and $\varepsilon > 0$) is given by

$$\mathbb{E}\, e^{s\,C_{n,m}} = \exp\left(\frac{\log\left(1-(1-\varepsilon)^2\right)}{2}\left(1-e^s\right)\right)\left(1+\mathcal{O}\left(\frac{1}{m}\right)\right),$$

and

$$\mathbb{E}\, e^{s\,C_{n,m,k}} = \exp\left(-\frac{(1-\varepsilon)^{2k}}{2k}\left(1-e^s\right)\right)\left(1+\mathcal{O}\left(\frac{1}{m}\right)\right),$$

respectively, where s is any fixed real number. Since the moment-generating function of a Poisson distribution $Po(\lambda)$ is given by $e^{\lambda(e^s-1)}$ we immediately deduce Theorem 7.3, once these formulas are proven.

We start with the calculation of the total number of cycles. For this purpose, we introduce a new variable w that marks each cyclic component, that is, the exponent of w counts the number of cycles. Equation 7.4 generalizes to

$$g^{\circ}_c(x, y, v, w) = \exp\left(\frac{1}{v}\tilde{t}(xv, yv) + \frac{w}{2}\log\frac{1}{1-t_1(x, y)t_2(x, y)}\right)$$
$$= \frac{\exp\left(\frac{1}{v}\tilde{t}(xv, yv)\right)}{(1-t_1(xv, yv)t_2(xv, yv))^{w/2}}.$$

Of course, we have $g_c^\circ(x, y, v, 1) = g^\circ(x, y, v)$. Hence, the moment-generating function is given by

$$\mathbb{E}\, e^{s\,C_{n,m}} = \frac{[x^m y^m v^n]\, g^\circ(x, y, v, e^s)}{[x^m y^m v^n]\, g^\circ(x, y, v, 1)}.$$

Again, the number of tree components equals $2m - n$, thus the generating function simplifies to

$$\left[\frac{x^m y^m v^n}{(m!)^2 n!}\right] g_c^\circ(x, y, v, e^s) = \frac{n!(m!)^2}{(2m-n)!}[x^m y^m] \frac{\tilde{t}(x, y)^{2m-n}}{(1 - t_1(x, y)t_2(x, y))^{e^s/2}}.$$

We continue using Cauchy's formula and the double saddle point method described in Lemma 7.4. Note that we can use the same saddle point $x_0 = y_0 = (1 - \varepsilon')e^{\varepsilon' - 1}$. The calculation is even easier because it is sufficient to calculate the leading term.

We make use of the inequality

$$\left|(1 - t_1(x, y)t_2(x, y))^{-e^s/2}\right| \le (1 - t_1(x_0, y_0)t_2(x_0, y_0))^{-e^s/2},$$

that is satisfied on the lines $|x| = x_0$, $|y| = y_0$ of integration. Furthermore, since $e^s = \mathcal{O}(1)$, we obtain a corresponding result:

$$[x^m y^m] \frac{\tilde{t}(x, y)^{2m-n}}{(1 - t_1(x, y)t_2(x, y))^{e^s/2}} \sim \frac{1}{2\pi(x_0 y_0)^m k\sqrt{\Delta}} \frac{\tilde{t}(x_0, y_0)^{2m-n}}{(1 - t_1(x_0, y_0)t_2(x_0, y_0))^{e^s/2}}.$$

Thus, we obtain the moment-generating function

$$\begin{aligned}\mathbb{E}\, e^{s\,C_{n,m}} &= \frac{\sqrt{1 - t_1(x_0, y_0)t_2(x_0, y_0)}}{(1 - t_1(x_0, y_0)t_2(x_0, y_0))^{e^s/2}} \left(1 + \mathcal{O}\left(\frac{1}{m}\right)\right) \\ &= \left(1 - (1 - \varepsilon)^2\right)^{(1-e^s)/2} \left(1 + \mathcal{O}\left(\frac{1}{m}\right)\right),\end{aligned}$$

which completes the proof of the first part.

The proof of the second part is very similar, we just replace g_c° by the generating function

$$g_k^\circ(x, y, v, w) = \frac{\exp\left(\frac{1}{v}\tilde{t}(xv, yv) + (w - 1)\frac{1}{2k} t_1(xv, yv)^k t_2(xv, yv)^k\right)}{\sqrt{1 - t_1(xv, yv)t_2(xv, yv)}}.$$

Hereby, w is used to mark cycles of length $2k$. Recall that the generating function of a component containing a cycle of length $2k$ is given by $\frac{1}{2k}t_1(x,y)^k t_2(x,y)^k$. We proceed as usual and yield

$$\left[\frac{x^m y^m v^n}{(m!)^2 n!}\right] g_k^\circ(x,y,v,e^s)$$
$$= \frac{n!(m!)^2}{(2m-n)!}[x^m y^m] \frac{\exp\left((e^s-1)\frac{1}{2k}t_1(x,y)^k t_2(x,y)^k\right)}{\sqrt{1-t_1(x,y)t_2(x,y)}} \tilde{t}(x,y)^{2m-n}.$$

Finally, the moment-generating function of $C_{n,m,k}$ equals

$$\begin{aligned}\mathbb{E}\, e^{s C_{n,m,k}} &= \frac{[x^m y^m v^n] g_k^\circ(x,y,v,e^s)}{[x^m y^m v^n] g_k^\circ(x,y,v,1)} \\ &= \exp\left((e^s-1)\frac{1}{2k}t_1(x_0,y_0)^k t_2(x_0,y_0)^k\right)\left(1+\mathcal{O}\left(\frac{1}{m}\right)\right) \\ &= \exp\left(-\frac{(1-\varepsilon)^{2k}}{2k}\left(1-e^s\right)\right)\left(1+\mathcal{O}\left(\frac{1}{m}\right)\right),\end{aligned}$$

which completes the proof. ■

Next, we consider the number of vertices contained in cyclic components.

Proof (of Theorem 7.4): For the first part of the proof, we have to count the number of vertices $V_{n,m}$ contained in cycles. We make use of the generating function

$$g_c^\circ(x,y,v,w) = \frac{\exp\left(\frac{1}{v}\tilde{t}(xv,yv)\right)}{\sqrt{1-w^2 t_1(xv,yv)t_2(xv,yv)}},$$

where the exponent of w counts the number of cyclic points. Hence by again using the double saddle point methods, we get the claimed characteristic function, see Ref. [7] for details.

If we count the number of all vertices contained in cyclic components, the generating function modifies to

$$g_v^\circ(x,y,v,w) = \frac{\exp\left(\frac{1}{v}\tilde{t}(xv,yv)\right)}{\sqrt{1-t_1(xvw,yvw)t_2(xvw,yvw)}}.$$

Here, we took care of all vertices of trees that are attached to cycles. It is straightforward to calculate asymptotic mean and variance. ■

Finally, we consider tree components.

Proof (of Theorem 7.2): The proof of this theorem is more complicated, since we also normalize depending on m. As in the formulation of Section 7.2, we use the following notation:

$$\mu = 2\frac{k^{k-2}(1-\varepsilon)^{k-1}e^{k(\varepsilon-1)}}{k!},$$

and

$$\sigma^2 = \mu - \frac{2e^{2k(\varepsilon-1)}k^{2k-4}(1-\varepsilon)^{2k-3}(k^2\varepsilon^2+k^2\varepsilon-4k\varepsilon+2)}{(k!)^2}$$

where $k \geq 1$ is integer and $0 < \varepsilon < 1$ holds.

First, we show that mean value and variance of the number of tree components $T_{m,n,k}$ with k vertices of a randomly chosen graph of $G^\circ_{m,m,n}$ (with $n = \lfloor(1-\varepsilon)m\rfloor$ and $\varepsilon > 0$) are given by

$$\mathbb{E}\, T_{mn,k} = m\mu + \mathcal{O}(1) \tag{7.9}$$

and by

$$\mathbb{V}\text{ar}\, T_{mn,k} = m\sigma^2 + \mathcal{O}(1)\,.$$

In what follows, we make use of the generating function of a bipartite tree components with $2k$ vertices. Because of Lemma 7.1, it is straightforward to calculate the number of unrooted trees possessing m_1 and m_2 nodes of each kind. Thus, it is easy to see that the generating function we are looking for is given by

$$\tilde{t}_k(x,y) = \sum_{m_1+m_2=k} m_1^{m_2-1} m_2^{m_1-1} \frac{x^{m_1}}{m_1!}\frac{y^{m_2}}{m_2!}.$$

We introduce the variable w to mark trees which are of size k and obtain the following generating function

$$g^\circ_{k,t}(x,y,v,w) = \frac{\exp\left(\frac{1}{v}\tilde{t}(xv,yv) + (w-1)\frac{1}{v}\tilde{t}_k(xv,yv)\right)}{\sqrt{1-t_1(xv,yv)t_2(xv,yv)}}.$$

The lth factorial moment is then given by

$$\mathbb{E}\, T_{mn,k}(T_{mn,k}-1)\cdots(T_{mn,k}-l+1) = \frac{[x^m y^m v^n]\left[\frac{\partial^l}{\partial w^l} g^\circ_t(x,y,v,w)\right]_{w=1}}{[x^m y^m v^n] g^\circ_t(x,y,v,1)}.$$

The numerator of this expression simplifies to

$$[x^m y^m v^n]\left[\frac{\partial^l}{\partial w^l} g_t^\circ(x, y, v, w)\right]_{w=1}$$
$$= [x^m y^m]\left[\frac{\partial^l}{\partial w^l}\frac{\left(\tilde{t}(x, y) + (w-1)\tilde{t}_k(x, y)\right)^{2m-n}}{(2m-n)!\sqrt{1-t_1(x, y)t_2(x, y)}}\right]_{w=1}$$
$$= [x^m y^m]\frac{\tilde{t}(x, y)^{2m-n-l}}{(2m-n)!\sqrt{1-t_1(x, y)t_2(x, y)}}(2m-n)^{\underline{l}}\,\tilde{t}_k(x, y)^l.$$

Now, we apply Lemma 7.4 to calculate an asymptotic expansion and obtain that the leading term of $\mathbb{E}\, T_{mn,k}(T_{mn,k}-1)\cdots(T_{mn,k}-l+1)$ equals

$$\frac{(2m-n)^{\underline{l}}}{\tilde{t}(x_0, y_0)^l}\tilde{t}_k(x_0, y_0)^l = \frac{m^l(1+\varepsilon)^l}{(1-\varepsilon^2)^l}\left(2\frac{k^{k-2}}{k!}(1-\varepsilon')^k e^{(\varepsilon'-1)k}\right)^l\left(1+\mathcal{O}\left(m^{-1}\right)\right).$$

Moreover, we conclude that the variance is of order $O(m)$ too, thus its calculation requires to determine the next term of the asymptotic expansion. We do this in a semiautomatic way using Maple and obtain the proposed result.

To infer the limiting distribution, we make use of the characteristic function $\mathbb{E}\, e^{irT_{mn,k}}$. It is given by

$$\mathbb{E}\, e^{irT_{mn,k}} = \frac{[x^m y^m v^n]\, g_{k,t}^\circ(x, y, v, e^{ir})}{[x^m y^m v^n]\, g^\circ(x, y, v)},$$

where we can use the simplification

$$[x^m y^m v^n]\, g_{k,t}^\circ(x, y, v, e^{ir}) = [x^m y^m]\frac{\left(\tilde{t}(x, y) + (e^{ir}-1)\tilde{t}_k(x, y)\right)^{2m-n}}{\sqrt{1-t_1(x, y)t_2(x, y)}}.$$

Using a saddle point approach, it is possible to establish the following result, see Ref. [7] for details. For every $k \geq 1$ and for every real number r we have, as $m \to \infty$,

$$\mathbb{E}\, e^{ir(T_{mn,k}-\mu m)/\sqrt{\sigma^2 m}} = e^{-\frac{1}{2}r^2}\left(1+\mathcal{O}\left(m^{-\frac{1}{2}+\delta}\right)\right),$$

where $0 < \delta < \frac{1}{6}$, what completes the proof. ■

7.6 EMPIRICAL DATA

One might argue that our conclusions are based on asymptotic expansions only, thus it is not certain if the observations hold for practical relevant settings. To overcome this weak point, we obtained numerical results that are provided in this section.

We start providing some results concerning the phase transition and the "critical" value $\varepsilon = 0$, for ordinary and symmetric bipartite graphs. Recall that the latter value corresponds to the relation $m = n$, that is the number of edges equals half the

TABLE 7.2 Number of Graphs Containing a Complex Component out of 5 × 10^5 Graphs Generated for Each Setup

$m(=n)$	5,000	10,000	50,000	100,000	500,000
Nonbipartite	81,053	83,138	86,563	88,021	89,177
Symmetric bipartite	83,100	85,357	87,972	88,415	89,751

According to Theorem 7.1, we expect a value of 91,752 at the moment the critical value is reached.

number of nodes. From Theorem 7.1, we conclude that the probability that no complex component occurs drops from $1 + \mathcal{O}(1/m)$ to $\sqrt{2/3} + o(1)$. The numerical results given in Table 7.2 exhibit a similar behavior, see also Tables 7.4 and 7.5. Note that the observed number of complex components is slightly below the expectation calculated using the asymptotic approximation. However, the accuracy increases as the number of nodes increases.

Table 7.3 provides the numerically obtained average number of edges at the moment before the first bicyclic component is created. From the data given in Table 7.3, we conclude that this number is only slightly larger than m, except if the asymmetry is increased. Due to Ref. [13], we know that the first bicyclic component of a usual random graph appears at “time” $m + \Theta\left(m^{-2/3}\right)$, what is in accordance with our numerical results. Moreover, we conjure that the same (or a very similar) result holds for the bipartite graph too. Concerning asymmetric cuckoo hashing, we observe that the asymmetry reduces the critical ratio of edges and nodes that determines the start of phase transition.

Next, we consider the complex part of the graph in the subcritical phase. Recall that we consider the excess of this part of the graph, that is, how many more edges than nodes it contains. From Theorem 7.1, we deduce that the probability that this number equals r is given by $\mathcal{O}(m^{-r-1})$. Thus, we expect an exponentially decreasing pattern. The corresponding results are depicted in Tables 7.4 and 7.5. From the data given in both the tables, we additionally observe the following properties: for fixed $r > 0$ and m, the percentage of graphs with excess r increases as ε decreases. On the other hand, increasing m while holding ε constant, is likely to decrease the excess.

TABLE 7.3 Average Number of Edges at the Moment When the First Bicyclic Component Occurs

m	5,000	10,000	50,000	100,000	500,000
Symmetric bipartite	5,208	10,322	50,917	101,440	504,143
Asymmetric, $c = 0.1$	5,181	10,272	50,661	100,930	501,618
Asymmetric, $c = 0.2$	5,103	10,118	49,897	99,406	493,992
Asymmetric, $c = 0.3$	4,972	9,855	48,593	96,800	481,019
Asymmetric, $c = 0.4$	4,782	9,476	46,704	93,030	462,228
Nonbipartite	5,204	10,318	50,907	101,428	504,133

The table provides numerical data obtained over a sample size of 5×10^5.

TABLE 7.4 The Excess of the Complex Part of a Symmetric Bipartite Graph Possessing m Nodes of Each Type and $n = (1 - \varepsilon)m$ Keys

m	ε	Excess								
		1	2	3	4	5	6	7	8	>
500	0.2:	73,458	3,312	219	18	1				
	0.1:	283,258	33,842	4,638	778	108	17			
	0.06:	481,752	81,919	16,037	3,213	653	138	24	8	1
	0.04:	621,670	125,597	28,221	6,565	1,510	314	106	20	7
5×10^3	0.2:	12,196	119	1						
	0.1:	93,712	5,359	422	48	2	1			
	0.06:	253,414	29,837	4,563	780	159	29	7	2	
	0.04:	421,374	70,795	14,759	3,350	813	237	48	15	6
5×10^4	0.2:	1,286	1							
	0.1:	14,197	156	2						
	0.06:	61,276	2,507	159	12	1				
	0.04:	153,484	12,716	1,491	220	44	6	2		

For each setting of m and ε, we use a sample size of 10^7 and count the graphs according to the excess. Note that most of the graphs do not have a complex part at all.

Note that these numerical results are in good accordance with the theoretical analysis. Comparing the two different graph models, we observe that the symmetric bipartite case usually leads to a lower occurrence of all positive values of excess. Again, this corresponds well to the findings of our asymptotic analysis.

TABLE 7.5 The Excess of the Complex Part of a Nonbipartite Graph Possessing $2m$ Nodes and $n = (1 - \varepsilon)m$ Keys

m	ε	Excess								
		1	2	3	4	5	6	7	8	>
500	0.2:	86,462	4,001	271	20	3				
	0.1:	313,309	37,468	5,407	753	132	24			
	0.06:	522,531	89,481	17,484	3,482	740	156	22	1	2
	0.04:	668,026	135,375	30,822	7,073	1,773	386	110	23	5
5×10^3	0.2:	14,043	138	1						
	0.1:	101,327	5,578	454	47	8	1			
	0.06:	266,735	31,215	4,898	869	163	35	10	1	1
	0.04:	439,806	73,228	15,373	3,604	891	209	63	24	6
5×10^4	0.2:	1,602	1							
	0.1:	14,965	182	2						
	0.06:	63,861	2,573	176	7					
	0.04:	158,141	13,246	1,601	221	25	10	1	1	

For each setting of m and ε, we use a sample size of 10^7 and count the graphs according to the excess. Note that most of the graphs do not have a complex part at all.

TABLE 7.6 Number of Trees of Sizes One and Two for $\varepsilon = 0.1$

	Isolated Nodes				Trees with 2 Nodes			
m	Sample Mean	Rel. Error (%)	Sample Var.	Rel. Error (%)	Sample Mean	Rel. Error (%)	Sample Var.	Rel. Error (%)
Symmetric bipartite								
5×10^3	4,065	0.009	916	0.925	744	−0.015	642	0.205
5×10^4	40,657	0.000	9,279	−0.313	7,439	−0.004	6,406	0.372
5×10^5	406,568	0.000	92,629	−0.138	74,385	−0.001	64,291	0.018
Asymmetric bipartite, $c = 0.2$								
5×10^3	4,132	0.011	895	−0.124	690	−0.011	589	−0.014
5×10^4	41,328	0.001	8,935	0.064	6,901	−0.004	5,970	−1.324
5×10^5	413,280	0.000	88,870	0.599	69,009	0.000	58,446	0.809
Asymmetric bipartite, $c = 0.3$								
5×10^3	4,219	0.020	853	0.135	623	−0.031	523	0.012
5×10^4	42,202	0.002	8,498	0.528	6,226	−0.003	5,206	0.492
5×10^5	422,029	0.001	85,273	0.182	62,255	−0.002	52,370	−0.111
Nonbipartite								
5×10^3	4,065	0.009	918	0.712	744	−0.013	641	0.335
5×10^4	40,657	0.000	9,333	−0.891	7,439	−0.001	6,482	−0.803
5×10^5	406,571	−0.000	93,055	−0.598	74,384	0.000	64,603	−0.468

The table provides sample mean and sample variance of the number of trees obtained over a sample of size 10^5. Additionally we give the relative error with respect to the asymptotic approximations of Theorem 7.2.

Table 7.6 displays the average number of trees of size one (isolated nodes) and two counted during 10^5 experiments. Further, we consider several different models, including some asymmetric bipartite settings. Recall that higher asymmetry leads to a lower maximum load factor, hence some small values for ε would be invalid for some asymmetric settings. From the data given in Table 7.6, we see that our asymptotic results are good approximations. In particular, we observe that the symmetric bipartite and the ordinary model do not only share the same limiting distribution concerning the number of trees, they also exhibit a indistinguishable behavior in practice. However, we deduce that asymmetry increases the number of isolated nodes.

Finally, we draw our attention on the structure of cyclic components. Note that the symmetric bipartite and the ordinary model are again in some sense very similar, but not identical. Table 7.7 provides numerical data for the number of nodes in cyclic components and the number of cycles. Our experiments, using settings from $m = 5 \times 10^3$ up to $m = 5 \times 10^5$, show that the size of the graph does not have significant influence on this parameters. Because of this, we do not provide data for different sizes. From the results presented in Table 7.7, we see again that the asymptotic results of Theorems 7.3 and 7.4 provide suitable estimates. We notice that asymmetry leads to an increased number of cycles, and nodes in cyclic components. Furthermore, both

TABLE 7.7 The Table Shows Sample Mean and Sample Variance of the Number of Nodes Contained in Cyclic Components and the Number of Cycles

	Nodes in Cyclic Components				Number of Cycles			
ε	Sample Mean	Rel. Error (%)	Sample Var.	Rel. Error (%)	Sample Mean	Rel. Error (%)	Sample Var.	Rel. Error (%)
Symmetric bipartite								
0.2	8.86	0.326	420.9	0.90	0.509	0.381	0.507	0.732
0.1	42.36	0.649	8,191.8	1.59	0.830	0.079	0.827	0.415
0.06	123.54	2.350	64,162.5	7.37	1.069	0.625	1.066	0.876
0.04	279.76	4.803	307,422	15.49	1.268	0.367	1.270	0.269
Asymmetric bipartite, $c = 0.2$								
0.2	11.01	−0.050	619.7	1.94	0.549	0.043	0.548	0.311
0.1	65.66	1.940	19,211.2	3.89	0.924	0.394	0.917	1.234
0.06	272.48	5.264	292,639	16.07	1.257	0.680	1.258	0.561
0.04	897.51	25.207	2590,900	56.16	1.568	2.576	1.551	3.660
Asymmetric bipartite, $c = 0.3$								
0.2	15.00	0.052	1,132.7	0.135	0.608	−0.041	0.610	−0.465
0.1	143.88	1.865	86,487.5	6.546	1.11	−0.144	1.100	0.409
0.06	1,353.49	42.292	5,116,340	77.17	1.689	4.558	1.650	6.758
Nonbipartite								
0.2	9.95	0.507	445.3	1.05	0.801	0.496	0.802	0.377
0.1	44.81	0.422	8,375.5	2.04	1.149	0.241	1.150	0.137
0.06	127.24	2.543	65,382.4	7.07	1.406	0.046	1.406	0.041
0.04	284.33	5.223	305,917	16.8	1.612	−0.140	1.603	0.380

We provide numerically obtained results using a sample of size 10^5 and give the relative error with respect to the asymptotic approximations of Theorems 7.3 and 7.4. All depicted values are obtained using a fixed table size determined by the parameter $m = 5 \times 10^5$, but our numerical data obtained using different table sizes are almost identical.

parameters are higher if we consider the ordinary model instead of the symmetric bipartite version.

7.7 CONCLUSION AND SUMMARY

We considered the growth process of sparse random bipartite graphs. Our analysis was based on a generating function approach by applying a double saddle point method. Thus, we obtained asymptotic results concerning the component structure of the graph. In particular, we considered the distribution of the number of tree components of given size, the number of cycles, and the number of nodes contained in cycles, and the probability that components of certain complexity occur. Hence, we showed that it is very likely that the graph consists of trees and unicyclic components only, if the number of edges is less than a critical value. Using further calculations, we obtained a Gaussian limit law for the number of trees, and limiting Poisson distribution for the number of cycles. Finally, we provided some results concerning the critical value, where a phase transition occurs.

We considered both symmetric bipartite graphs possessing an equal number of nodes of both kinds as well as an asymmetric model. Furthermore, we provided corresponding results concerning nonbipartite graphs and found substantial similarities. Finally, we provided numerical results, which positively supported our theoretical hypothesis.

As future work, we suggest a detailed analysis of the partial differential recursion of the generating functions of complex bipartite graphs with positive excess. Using these results, it will be possible to study the phase transition in full detail, similar to the analysis given in Ref. [3].

REFERENCES

1. B. Bollobás, *Random Graphs*, 2nd edn. Cambridge University Press, Cambridge, UK, 2001.
2. S. Janson, T. Łuczak, A. Rucinski, *Random Graphs*, Wiley, New York, 2000.
3. S. Janson, D.E. Knuth, T. Łuczak, B. Pittel. The birth of the giant component. *Random Struct. Algor.* **4**(3), 233–359 (1993).
4. J. Blasiak, R. Durrett, Random oxford graphs. *Stochastic Process. Appl.* **115**(8), 1257–1278 (2005).
5. R. Pagh, F.F. Rodler. Cuckoo hashing. *J. Algorithms* **51**(2), 122–144 (2004).
6. L. Devroye, P. Morin, Cuckoo hashing: further analysis. *Inform. Process. Lett.* **86**(4), 215–219 (2003).
7. M. Drmota, R. Kutzelnigg, A precise analysis of cuckoo hashing. *ACM Trans. Algorithms* **8**(2), (2012).
8. A. Kirsch, M. Mitzenmacher, U. Wieder, More robust hashing: cuckoo hashing with a stash, in *Proceedings of the 16th Annual European Symposium on Algorithms*, 2008.
9. R. Kutzelnigg, Bipartite random graphs and cuckoo hashing, in *Proceedings of the 4th Colloquium on Mathematics and Computer Science*, Discrete Mathematics and Theoretical Computer Science, pp. 403–406, 2006.
10. R. Kutzelnigg, *Random Graphs and Cuckoo Hashing*, Südwestdeutscher Verlag für Hochschulschriften, Saarbrücken, 2009.
11. R. Kutzelnigg, A further analysis of cuckoo hashing with a stash and random graphs of excess r. *DMTCS* **12**(3), 81–102 (2010).
12. R. Kutzelnigg, An improved version of cuckoo hashing: average case analysis of construction cost and search operations. *Math. Comput. Sci.* **3**(1), 47–60 (2010).
13. P. Flajolet, D.E. Knuth, B. Pittel, The first cycles in an evolving graph. *Discrete Math.* **75**(1–3), 167–215 (1989).
14. I.B. Kalugin, The number of components of a random bipartite graph. *Discrete Math. Appl.* **1**(3), 289–299 (1991).
15. O. Gimenez, A. de Mier, M. Noy, On the number of bases of bicircular matroids. *Ann. Comb.* **9**(1), 35–45 (2005).

16. P. Flajolet, R. Sedgewick, *Analytic Combinatorics*, Cambridge University Press, Cambridge, UK, 2009.
17. M. Drmota, A bivariate asymptotic expansion of coefficients of powers of generating functions. *Eur. J. Combin.* **15**(2), 139–152 (1994).
18. D. Gardy, Some results on the asymptotic behaviour of coefficients of large powers of functions. *Discrete Math.* **139**(1–3), 189–217 (1995).
19. I.J. Good, Saddle-point methods for the multinomial distribution. *Ann. Math. Stat.* **28**(4), 861–881 (1957).
20. M. Drmota, M. Soria, Marking in combinatorial constructions: generating functions and limiting distributions. *Theor. Comput. Sci.* **144**(1&2), 67–99 (1995).
21. M. Drmota, M. Soria, Images and preimages in random mappings. *SIAM J. Discrete Math.* **10**(2), 246–269 (1997).
22. P. Flajolet, A.M. Odlyzko, Random mapping statistics. *LNCS* **434**, 329–354 (1990).

8 GRAPH KERNELS

Matthias Rupp

Graph kernels are formal similarity measures defined directly on graphs. Because they are positive semidefinite functions, they correspond to inner products. This property makes them suitable for use with kernel-based machine learning algorithms, such as support vector machines and Gaussian processes. In this chapter, I present different types of graph kernels (based on random walks, shortest paths, tree patterns, cyclic patterns, graphlets, and optimal assignments), give an overview of successful applications in bio- and cheminformatics, and discuss advantages and limitations of kernels between graphs.

8.1 INTRODUCTION

Graphs are highly versatile data structures [1] that are used in a wide range of research areas. Consequently, the comparison of two graphs is a problem with many applications. Examples include cheminformatics [2,3], bioinformatics [4], sociology [5,6], telecommunication (e.g., the internet), computer vision [7], and natural language processing [8].
Typical approaches for comparing two graphs are based on

(i) comparing sets of vertices and edges, which is fast (runtime often scales linearly or quadratically in the size of the graphs), but neglects graph topology,
(ii) mining frequent or discriminative subgraphs, which can be slow (with potentially exponentially many subgraphs to be considered),

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

(iii) graph kernels, which constitute a trade-off between runtime and discriminative power.

Another possibility [9] is to categorize graph (dis)similarity measures into those based on (sub)graph isomorphism [10,11] (e.g., Zelinka distance [12], or, distances based on largest common subgraph and smallest common supergraph [13]), graph transformations (e.g., edit distance [14]), adjacency matrices [15], grammars [16], and others. Yet another view is to distinguish between the approaches that construct explicit features, such as graph invariants, and those that perform implicit comparisons, such as graph kernels. The latter are thus a special case of graph similarity measures.

Informally, a graph kernel is a function defined directly on two graphs that corresponds to an inner product, that is, it encodes geometrical information such as length and angle between representations of two graphs in some vector space. Through this property, a graph kernel allows the complete repertoire of kernel-based machine learning to be used directly with graphs as inputs, an approach that has proved to be fruitful in a large number of real-world applications.

This chapter gives an overview of graph kernels. The rest of the introduction recapitulates the idea of kernel-based machine learning, introduces the concept of a graph kernel, and discusses the basic computational aspects. The following sections present major types of graph kernels, followed by applications in bio- and cheminformatics. The chapter ends with a discussion of the advantages and disadvantages of graph kernels.

8.1.1 Kernel Learning in a Nutshell

The two key ideas of kernel-based machine learning are to (nonlinearly) *transform* the input data, and to do inference *implicitly* in the transformed space.

Figure 8.1 illustrates the utility of transforming input samples into another space. For real-world data, the right transformation is usually not known in advance, although domain knowledge can aid in choosing a good transformation. Fortunately, good

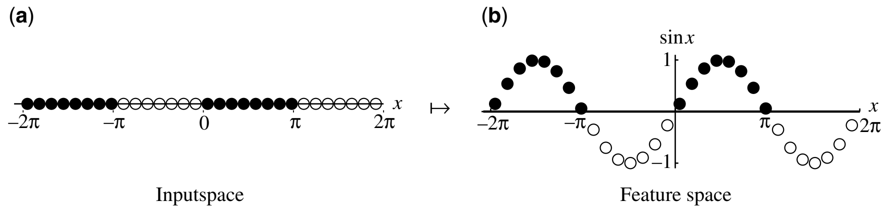


FIGURE 8.1 The utility of input transformations. In the original one-dimensional input space (a), samples are not linearly separable (there is no point such that all filled circles are on one side, and all empty circles are on the other side). In the transformed space (b), obtained by adding another dimension that is the sine of the original one, samples are linearly separable (there is a line, the x-axis, that separates both classes).

default transformations are available, for example, the radial basis function kernel (also called Gaussian kernel, or squared exponential kernel).

Usually, it is computationally not feasible to run a machine learning algorithm directly in the transformed space due to its high dimensionality; sometimes, it is outright impossible, for example, for transformations into infinite-dimensional vector spaces. With the *kernel trick*, one solves this problem by rewriting the machine learning algorithm in question such that it uses only inner products between the input samples.[1] The key observation is that there exists a class of functions (the "kernels" in kernel-based learning) that can be computed in the original input space, but yield the same result as the inner product in the transformed space. Replacing all inner product evaluations with a kernel results in an algorithm that performs implicit computations in feature space (but does only explicit calculations in the original input space).

As an example, consider the polynomial kernel of degree 2 on three-dimensional input samples,

$$k : \mathbb{R}^3 \times \mathbb{R}^3 \mapsto \mathbb{R}, \qquad k\big(\boldsymbol{x}, \boldsymbol{x}'\big) = \big\langle \boldsymbol{x}, \boldsymbol{x}' \big\rangle^2 . \tag{8.1}$$

The kernel k can be computed in the original input space (Eq. 8.2a), but yields the same result as computing the inner product in a six-dimensional vector space given by the transformation $\phi(\boldsymbol{x}) = (\boldsymbol{x}_1^2, \boldsymbol{x}_2^2, \boldsymbol{x}_3^2, \sqrt{2}\boldsymbol{x}_1\boldsymbol{x}_2, \sqrt{2}\boldsymbol{x}_1\boldsymbol{x}_3, \sqrt{2}\boldsymbol{x}_2\boldsymbol{x}_3)$ (Eq. 8.2b),

$$\begin{aligned} k\big(\boldsymbol{x}, \boldsymbol{x}'\big) &= \big\langle \boldsymbol{x}, \boldsymbol{x}' \big\rangle^2 = (\boldsymbol{x}_1\boldsymbol{x}_1' + \boldsymbol{x}_2\boldsymbol{x}_2' + \boldsymbol{x}_3\boldsymbol{x}_3')^2 && (8.2a)\\ &= \boldsymbol{x}_1^2\boldsymbol{x}_1'^2 + \boldsymbol{x}_2^2\boldsymbol{x}_2'^2 + \boldsymbol{x}_3^2\boldsymbol{x}_3'^2 + 2\boldsymbol{x}_1\boldsymbol{x}_1'\boldsymbol{x}_2\boldsymbol{x}_2' + 2\boldsymbol{x}_1\boldsymbol{x}_1'\boldsymbol{x}_3\boldsymbol{x}_3' + 2\boldsymbol{x}_2\boldsymbol{x}_2'\boldsymbol{x}_3\boldsymbol{x}_3' \\ &= \big\langle (\boldsymbol{x}_1^2, \boldsymbol{x}_2^2, \boldsymbol{x}_3^2, \sqrt{2}\boldsymbol{x}_1\boldsymbol{x}_2, \sqrt{2}\boldsymbol{x}_1\boldsymbol{x}_3, \sqrt{2}\boldsymbol{x}_2\boldsymbol{x}_3), \\ &\quad (\boldsymbol{x}_1'^2, \boldsymbol{x}_2'^2, \boldsymbol{x}_3'^2, \sqrt{2}\boldsymbol{x}_1'\boldsymbol{x}_2', \sqrt{2}\boldsymbol{x}_1'\boldsymbol{x}_3', \sqrt{2}\boldsymbol{x}_2'\boldsymbol{x}_3')\big\rangle = \big\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \big\rangle . && (8.2b) \end{aligned}$$

In general, a *kernel* is a symmetric positive definite function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is a finite set (the input space). A positive definite function corresponds to an inner product in some Hilbert space $\mathcal{H}$, that is, there is a transformation $\phi : \mathcal{X} \to \mathcal{H}$ such that $k\big(x, x'\big) = \big\langle \phi(x), \phi(x') \big\rangle$ [17]. Note that the transformation ϕ need not be unique, and that no structural assumptions are made with respect to $\mathcal{X}$; in particular, $\mathcal{X}$ need not be a vector space. For further information on kernel-based machine learning, see the literature [17–20].

8.1.2 Graph Kernels

A *graph kernel* is a kernel defined directly on graphs, that is, a symmetric positive definite function $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$, where $\mathcal{G}$ is a nonempty set of graphs.

Note that technically, this definition includes kernels that use an explicit intermediate vector representation, that is, kernels that first construct a vector representation of the input graphs, and then apply a vector-based kernel. An example of such an

[1]This is possible because the inner product encodes geometrical information about the distance and angle between two vectors.

approach is the graphlet spectrum (Section 8.7). While all graph kernels *implicitly* represent graphs as vectors in some inner product space, most do not do so explicitly. I will use the term graph kernel in the more narrow sense of a kernel that is defined *directly* on the input graphs, without any explicit intermediate vector representation.

Graph kernels are kernels *between* two graphs, and are different from, but related to, kernels *on* graphs, that is, kernels defined on the vertices of a single graph, such as the diffusion kernel [21,22], the regularized Laplacian kernel [23], or the von Neumann kernel [24]. In this chapter, only kernels between graphs, and no kernels between graph vertices, trees, and (graphical) generative models [25,26] have been considered.

Table 8.1 presents an overview of graph kernels, sorted by first publication year.

8.1.3 Computational Considerations

All graph kernels are by necessity a trade-off between discriminative capability and computational speed. To see this, consider the notion of a complete graph kernel: two graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic* if they are identical up to a relabeling of their vertices,

$$G \simeq G' \;\Leftrightarrow\; \exists \pi : V \to V' \;\; \forall (u, v) \in V \times V : (u, v) \in E \Leftrightarrow (\pi(u), \pi(v)) \in E'. \tag{8.3}$$

Let $k(G, G') = \langle \phi(G), \phi(G') \rangle$, where $\phi : \mathcal{G}_{/\simeq} \to \mathcal{H}$ maps from the quotient set $\mathcal{G}_{/\simeq}$ of all graphs with respect to isomorphism into some Hilbert space $\mathcal{H}$. The graph kernel k is *complete* if ϕ is injective, that is, if it can distinguish between all nonisomorphic graphs. Gärtner et al. [58] have shown that computing any complete graph kernel is at least as hard as deciding whether two graphs are isomorphic. The latter problem is

TABLE 8.1 Graph Kernels, Listed in Order of First Publication Year

Year	Graph Kernel	References
2003	Marginalized graph kernels	[27–30]
2004	Cyclic pattern kernels	[31,32]
2005	Shortest path kernels	[33]
2005	Optimal assignment kernels	[34–41]
2005	Fingerprint kernels	[42,43]
2005	Decomposition kernels	[44,45]
2006	Matching-based kernels	[46–48]
2007	Complement graph kernels	[49]
2008	Point cloud kernels	[50]
2008	Graphlet kernels	[51–53]
2009	Finite length random walk kernels	[54]
2009	Tree pattern kernels	[55,56]
2009	Neighborhood hash kernels	[57]

believed to have complexity between P and NP for general graphs[2] [60]. Thus, there is no graph kernel for general graphs that is both complete and efficient.

8.2 CONVOLUTION KERNELS

Many kernels for structured data, including, but not limited to graphs, are based on the idea of convolution kernels introduced by Haussler in 1999 [62].[3]

8.2.1 Definition

Assume that a sample $x \in \mathcal{X}$ can be decomposed into parts $x_1, \ldots, x_d \in \mathcal{X}_1, \ldots, \mathcal{X}_d$, for example, a decomposition of a graph into subgraphs. Here, $\mathcal{X}, \mathcal{X}_1, \ldots, \mathcal{X}_d$ are nonempty, separable metric spaces. The relation R indicates possible decompositions, where $R(x, x_1, \ldots, x_d)$ is true if and only if x can be decomposed into $x_1, \ldots, x_d$. Given positive definite kernels $k_i : \mathcal{X}_i \times \mathcal{X}_i \to \mathbb{R}$, $1 \leq i \leq d$, the *convolution kernel*

$$(k_1 \times \cdots \times k_d)(x, x') = \sum_R \prod_{i=1}^{d} k_i(x_i, x_i') \tag{8.4}$$

is positive definite for finite R [62]. The sum runs over all decompositions of x and x' into d parts for which R is true; if a sample cannot be decomposed, the sum is zero. Random walk kernels, path-based kernels, tree kernels, and cyclic pattern kernels are convolution kernels.

8.2.2 Variants and Extensions

Vishwanathan et al. [64] point out that "there have been a few attempts to extend R-convolution kernels to abstract semirings." A semiring $(S, \oplus, \otimes, 0, 1)$ is an algebraic structure that consists of a set S, a commutative associative addition $\oplus$ with identity element 0, and an associative multiplication $\otimes$ with identity element 1, where multiplication distributes over addition, and $0 \otimes a = a \otimes 0 = 0$ for all $a \in S$. In a semiring, Equation 8.4 becomes

$$(k_1 \times \cdots \times k_d)(x, x') = \bigoplus_R \bigotimes_{i=1}^{d} k_i(x_i, x_i'). \tag{8.5}$$

See Section 8.8.3 for an example.

[2]Note that for many specialized graph classes, graph isomorphism is in P [59,60]. Also, for a suitable distribution on graphs, graph isomorphism is efficiently solvable in expected polynomial time [61].

[3]Structured data kernels also relate to work by Watkins on dynamic alignment kernels [63], also in 1999, although his work is not as directly related to graph kernels.

8.3 RANDOM WALK GRAPH KERNELS

The idea of random walk graph kernels is to perform random walks on two graphs and compare the resulting label sequences. They can be seen as kernels on label sequences marginalized with respect to these random walks, and thus are also called *marginalized graph kernels*. Specific kernels differ in their random walk model and the kernels used to compare vertex and edge labels. Random walk kernels include label sequence kernels [28,58], marginalized graph kernels [27], and geometric graph kernels [65].

8.3.1 Definition

A random walk on a graph G (Fig. 8.2) starts in vertex $x_1 \in V$ with probability p_s, goes from x_{i-1} to x_i with transitional probability p_t conditional on x_{i-1}, and ends with probability p_q. A random walk instance $x = x_1 \cdots x_l$ has probability

$$\mathbb{P}(x \mid G) = p_s(x_1)\Big(\prod_{i=2}^{l} p_t(x_i|x_{i-1})\Big)\, p_q(x_l). \tag{8.6}$$

Default choices are $p_s(v_i) = |V|^{-1}$, $p_q(v_i) = c$ for a small constant $0 < c \leq 1$, and, $p_t(v_i|x_{i-1}) = \frac{1-c}{d(x_{i-1})}$, where $d(v)$ is the degree of v. The random walk x has *label sequence* $h_x = \ell(x_1)\,\ell(\{x_1, x_2\})\,\ell(x_2)\cdots\ell(x_l)$, where $\ell(\cdot)$ denotes the label of a vertex or edge. The probability of a label sequence is the sum of the probabilities of all random walks that generate it. A kernel on label sequences of equal length is given by the product of the label kernels

$$k_z(h_x, h'_x) = k_v(h_1, h'_1)\prod_{i=1}^{l-1} k_e(h_{2i}, h'_{2i})k_v(h_{2i+1}, h'_{2i+1}). \tag{8.7}$$

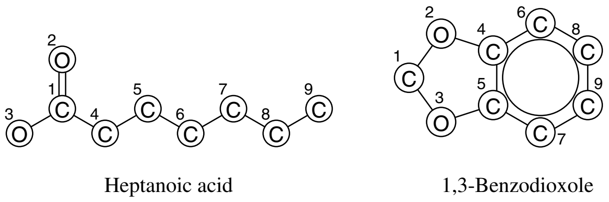


FIGURE 8.2 Random walks on two molecular structure graphs. Vertices are labeled with element type, edges with covalent bond type (– single, = double, |(aromatic). Using only vertex labels, the random walks 12131456 on the left graph and 53124689 on the right graph both produce the label sequence COCOCCCC. Taking edge labels into account (s = single, d = double, a = aromatic) leads to different label sequences CdOdCsOsCsCsCsC and CsOsCsOsCaCaCaC of the same random walks. The tottering walks 13131414 and 42424646 reproduce the first label sequence, but use only three vertices.

For different lengths, $k_z(h_x, h'_x) = 0$. A (*label sequence*) *random walk graph kernel* is given by the expectation of k_z over all possible label sequences:

$$k(G, G') = \sum_{h,h'} k_z(h, h')\mathbb{P}(h \mid G)\,\mathbb{P}\big(h' \bigm| G'\big)\,. \tag{8.8}$$

For non-negative k_v and k_e, Equation 8.8 is positive definite [28]. It is an example of a *marginalized kernel* [27], that is, a kernel between visible and hidden variables—here, graphs and random walks—computed by taking the expectation over the hidden variables.

An alternative formulation is based on the observation that simultaneous random walks on G and G' correspond to one random walk on the direct product graph $G \times G'$ and vice versa [1,49,64]. The start, transition, and stop probabilities of $G \times G'$ are the products of the corresponding probabilities of G and G',

$$p_s^{\times}\left(\{x_i, x'_j\}\right) = p_s(x_i)p_s(x'_j), \tag{8.9}$$

$$p_q^{\times}\left(\{x_i, x'_j\}\right) = p_q(x_i)p_q(x'_j), \tag{8.10}$$

$$p_t^{\times}\left(\{x_i, x'_j\}\big|\{x_{i-1}, x'_{j-1}\}\right) = p_t(x_i|x_{i-1})p_t(x_j|x_{j-1}). \tag{8.11}$$

8.3.2 Computation

For acyclic graphs, Equation 8.8 can be computed using topological sorting and dynamic programming in time $O\big(c\,c'\,|V|\,|V'|\big)$, where c, c' are the maximum vertex degrees in G and G'. In the general case, Equation 8.8 can be computed by solving a system of linear equations, or, equivalently, by inverting a sparse $|V|\,|V'| \times |V|\,|V'|$ matrix. In both cases, the number of nonzero coefficients is upper-bounded by $c\,c'\,|V|\,|V'|$. The solution exists if a convergence condition on the involved probabilities and kernels is met. For random walk models with constant $p_q(\cdot) = \gamma$, the requirement is $k_v(\cdot,\cdot)k_e(\cdot,\cdot) < \frac{1}{(1-\gamma)^2}$, which is met if $0 \leq k_v, k_e \leq 1$. The solution can be computed using matrix power series [66], fixed point iterations [28], the Sylvester or Lyapunov equation [64], or, conjugate gradient methods [49] in time $O(n^3)$, where $n = \max(|V|, |V'|)$.

8.3.3 Variants and Extensions

Tottering is the immediate revisiting of a vertex, that is, $x_i = x_{i+2}$ for some i (Fig. 8.2). Such excursions are likely to be uninformative and to add noise, particularly because the ratio of tottering to nontottering walks increases rapidly. It can be prevented by switching to second-order Markov random walks [30]. *Halting* [1] describes the dominance of short walks due to the diminishing probability of longer walks (the decay factor in other formulations of random walk kernels). Path-based graph kernels have been proposed to counter the effects of tottering and halting ([1]; see Section 8.4). In *label enrichment*, contextual information is embedded into the labels using the Morgan index, which improves computation time by decreasing the number of common paths while still giving comparable performance on test data sets [67]. In analogy to sequence matching, gaps can be allowed when comparing walks [54,58].

8.4 PATH-BASED GRAPH KERNELS

Path-based graph kernels are similar to random walk kernels, but use paths instead of walks. A path is a walk that contains each vertex at most once. Path kernels therefore do not suffer from either tottering (by definition) or halting, as the number of paths is finite and paths thus do not have to be downweighted to ensure convergence. Through this, path-based kernels have one free parameter less than random walk graph kernels.

8.4.1 Definition and Computation

Let $P(G)$ denote the set of all paths in G, and let k_z be a kernel defined on the corresponding label sequences as in Equation 8.7. The all-paths graph kernel

$$k(G, G') = \sum_{\substack{p \in P(G) \\ p' \in P(G')}} k_z(p, p'), \tag{8.12}$$

is an R-convolution kernel [1], and therefore positive definite. However, enumerating all paths in a graph is NP-hard, as it would allow to decide whether a graph contains a Hamiltonian path (a path that visits each vertex once), a known NP-complete problem [1]. The same reasoning holds for the restriction to longest paths.

Shortest paths, however, can be computed in polynomial time [68], for example, by the Floyd–Warshall algorithm, which solves the all-pairs-shortest path problem in time $O(|V|^3)$. Let $G = (V, E)$ denote an edge-weighted graph. Its *shortest-path graph* $\tilde{G} = (V, \tilde{E})$ has an edge between two vertices if and only if there is a path between them; the weight of the edge is the distance between the two vertices. The *shortest path graph kernel*

$$k(G, G') = \sum_{\substack{\tilde{e} \in \tilde{E} \\ \tilde{e}' \in \tilde{E}'}} k_e(\tilde{e}, \tilde{e}') \tag{8.13}$$

is an R-convolution kernel [1], where k_e is a kernel on edges. We exclude cycles with negative edge weights, as these would lead to shortest paths of length $-\infty$. If G is connected then $\tilde{G}$ has $|V|^2$ edges, and the pairwise comparison of all edges in Equation 8.13 leads to a runtime of $O(|V|^4)$. In practice, this can be faster than the cubic runtime of random walk kernels due to a smaller constant [1].

Path kernels lead to a full matrix representation of (connected) graphs, which might be problematic for larger graphs. They also ignore information from longer paths. The notion of shortest paths seems most meaningful if edge labels are distances of some kind.

8.4.2 Variants and Extensions

Label enrichment (Section 8.3.3) can be done by labeling vertices and edges with additional information. Since shortest paths are not unique, additional criteria might be required, for example, one might consider only shortest paths with minimum number of edges. As an example, in the *equal length shortest path kernel*, kernel values are set to zero for pairs of shortest paths with different numbers of edges, saving on the evaluation of vertex and edge kernels.

In case the shortest path does not contain enough information, other short paths can be taken into account at the expense of runtime cost. In principle, any algorithm that finds k loopless shortest paths [69,70] can be used for this, resulting in *k-shortest paths kernels*. As an example, on a connected graph the algorithm by Yen [69] takes time $O(k|V|^5 + k^2|V|^4)$ [1]. Repeated use of Dijkstra's algorithm [68], followed each time by removal of the found shortest path, leads to the k-disjunct shortest paths kernel, with runtime in $O(k|V|^4)$ for connected graphs [33].

8.5 TREE-PATTERN GRAPH KERNELS

Tree-based graph kernels [55] compare subtrees of graphs.

8.5.1 Definition

Let $G = (V, E)$ be a graph and let $T = (W, F)$, $W = \{w_1, \ldots, w_t\}$ be a rooted directed tree. A *tree pattern* of G with respect to T consists of vertices $v_1, \ldots, v_t \in G$ such that

$$\begin{aligned}
&\ell(v_i) = \ell(w_i) \text{ for } 1 \leq i \leq t\\
&\text{and } \{v_i, v_j\} \in E \wedge \ell\big(\{v_i, v_j\}\big) = \ell\big((w_i, w_j)\big) \text{ for } (w_i, w_j) \in W \qquad (8.14)\\
&\text{and } j \neq k \Leftrightarrow v_j \neq v_k \text{ for } (w_i, w_j), (w_i, w_k) \in W.
\end{aligned}$$

Each vertex w in the tree is assigned a vertex v in the graph such that edges and labels match. The $v_1, \ldots, v_t$ need not be distinct, as long as vertices assigned to sibling vertices in T are distinct (Fig. 8.3). The *tree pattern counting function* $\psi(G, T)$ returns

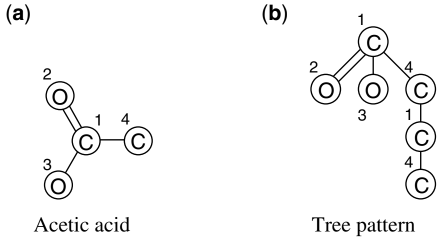


FIGURE 8.3 Tree patterns. Shown are the annotated graph of acetic acid (a) and a tree pattern contained in it (b). Numbers indicate assigned vertices. Note that vertices 1 and 4 appear twice; this is the equivalent of tottering in random walk kernels.

the number of times the tree pattern T occurs in the graph G, that is, the number of distinct tuples $(v_1, \ldots, v_t)$ that are tree patterns of T in G.

Let $G = (V, E)$, $G' = (V', E')$ be graphs, let $\mathcal{T}$ be a set of trees, and, let $w : \mathcal{T} \rightarrow \mathbb{R}_+$ weight the trees in $\mathcal{T}$. Then

$$k(G, G') = \sum_{T \in \mathcal{T}} w(T)\psi(G, T)\psi(G', T) \tag{8.15}$$

is positive definite as it corresponds to a weighted inner product in the feature space indexed by the trees in $\mathcal{T}$. Specific examples of w include weighting by size $w(T) = \lambda^{|T|-h}$ and by branching cardinality $w(T) = \lambda^{\text{branch}(T)}$ for balanced trees T of order h, where $|T|$ denotes the number of vertices in T and branch(T) denotes branching cardinality. The parameter λ controls the weight put on complex tree patterns: more weight is put on them for $\lambda > 1$, and less for $\lambda < 1$. In the limit of $\lambda \rightarrow 0$, only linear trees have nonzero weight, and the two kernels converge to the walk counting kernel. The branching cardinality weighting scheme can be extended to arbitrary trees of depth up to h.

8.5.2 Computation

Mahé and Vert [55] show how to compute such kernels using dynamic programming and the notion of neighborhood matching sets in time $O\big(|V|\,|V'|\,h\,c^{2c}\big)$, where c denotes the maximum vertex degree. Tottering (Fig. 8.3) can be prevented by additional constraints on ψ, where algorithms can be retained by transforming the input graphs, increasing runtime by a factor of

$$\frac{(|V| + |E|)(|V'| + |E'|)}{|V|\,|V'|}. \tag{8.16}$$

8.5.3 Variants and Extensions

Shervashidze and Borgwardt [56] use the "naive vertex refinement" version of the Weisfeiler–Lehman graph isomorphism test [71] to define a fast graph kernel that compares subtrees of (labeled) graphs. The test iteratively constructs multisets of label strings $s_i(v)$ that describe the neighborhood of each vertex; it terminates if either the multisets of the two graphs differ in any given round, or the maximum number h of iterations is reached. Consequently, the number of distinguishable isomorphic graphs depends on the free parameter h. The *Weisfeiler–Lehman* graph kernel simply counts the multiset strings common to both graphs:

$$k\big(G, G'\big) = \Big|\Big\{ \big(s_i(v), s_i(v')\big) \Big| \; f\big(s_i(v)\big) = f\big(s_i(v')\big), 1 \leq i \leq h, v \in V, v' \in V' \Big\}\Big|, \tag{8.17}$$

where f is an injective function that compresses strings of concatenated labels. For two graphs, this kernel can be computed in time $O(h\,|E|)$ using bucket sort [68] to exploit the bounded size of the label alphabet (and thus strings over it) in various sorting steps of the algorithm. The simultaneous computation of a kernel matrix for n graphs can be done in time $O(n\, h \max\{|E_1|, \ldots, |E_n|\} + n^2 h \max\{|V_1|, \ldots, |V_n|\})$ by using hashing to speed up label compression.

8.6 CYCLIC PATTERN KERNELS

Cyclic pattern graph kernels [31,32] are based on the idea of mapping graphs to sets of cyclic and tree pattern strings that are compared using the intersection kernel.

8.6.1 Definition

A subgraph is a *simple cycle* if it is connected and each vertex has degree 2. Let $S(G)$ denote the set of simple cycles in a graph G. An edge not belonging to a simple cycle is a *bridge*. We denote the subgraph of all bridges in G, which is a forest, by $\mathcal{B}(G)$ (Fig. 8.4). Let π be a mapping, computable in polynomial time, from the set of labeled simple cycles and trees to label strings that is injective modulo isomorphism. Note that such a mapping can always be constructed [31,72]. The sets of *cyclic* and *tree patterns* are given by $\mathcal{C}(G) = \{\pi(C) \mid C \in S(G)\}$ and $\mathcal{T}(G) = \{\pi(T) \mid T \text{ is a connected component of } \mathcal{B}(G)\}$. The *cyclic pattern kernel* is given by

$$k(G, G') = k_\cap\big(\mathcal{C}(G), \mathcal{C}(G')\big) + k_\cap\big(\mathcal{T}(G), \mathcal{T}(G')\big), \tag{8.18}$$

where $k_\cap(S, S') = |S \cap S'|$ denotes the intersection kernel.

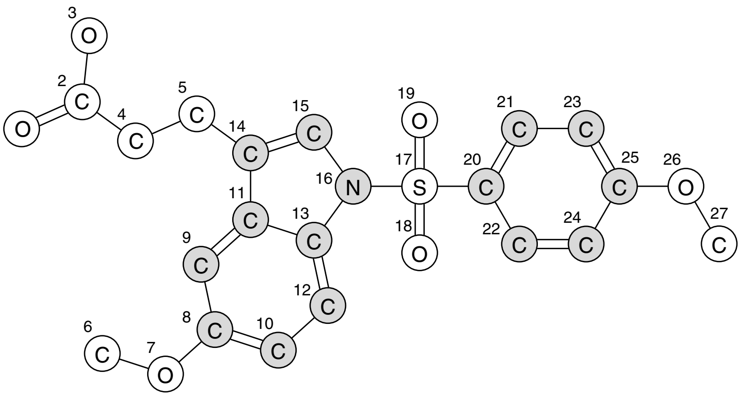


FIGURE 8.4 Cyclic and tree patterns of the molecular structure graph of indeglitazar. Shown are vertices belonging to simple cycles (shaded) and to bridges (white).

8.6.2 Computation

Computing Equation 8.18 is at least as hard as counting simple cycles in a graph, which is computationally not tractable if P $\neq$ NP [73]. For graphs with a small number of simple cycles, Equation 8.18 can be computed via enumeration of $\mathcal{B}, \mathcal{T}, \mathcal{S}, \mathcal{C}$ [31]. The number of cyclic and tree patterns in a graph can be exponential in $|V|$, leading to computational infeasibility of the cyclic pattern kernel for general graphs [31]. The restriction of inputs to graphs with few simple cycles can be relaxed to graphs of bounded treewidth, for which many NP-complete problems become tractable [74]. For graphs of constant bounded treewidth, Equation 8.18 can be computed in time polynomial in $\max\big\{|V|, |V'|, |\mathcal{C}(G)|, |\mathcal{C}(G')|\big\}$ [32].

8.6.3 Variants and Extensions

An alternative relaxation is to consider a different class of cycles. Horváth [32] uses algebraic graph theory to compute the cyclic pattern kernel on monotone increasing subsets of simple cycles generated by relevant cycles [75], with a similar runtime bound, but with different cyclic patterns. The number of relevant cycles is exponential in the worst case; for molecular graphs, it is typically cubic in $|V|$ [76].

8.7 GRAPHLET KERNELS

Graphlet kernels are based on the idea of randomly sampling small (connected) subgraphs of size $k \in \{3, 4, 5\}$. These samples can then be used to compare frequency distributions (count-based approach [52]) or to construct graph invariants (algebraic approaches, e.g., skew spectrum [51], graphlet spectrum [53]). This type of kernel is suited for the comparison of larger graphs with hundreds of vertices and thousands of edges.

8.7.1 Definition

Let $G_1, \dots, G_N$ denote a set of graphs of size $k \in \{3, 4, 5\}$, the graphlets. We explicitly construct a feature space of dimension N via the k-spectrum

$$\phi(G) = \big(\#(G_1 \sqsubseteq G), \dots, \#(G_N \sqsubseteq G)\big), \tag{8.19}$$

where $\#(H \sqsubseteq G)$ denotes the number of embeddings of H in G. The (count-based) *graphlet kernel* is the inner product in this space,

$$k(G, G') = \big\langle \phi(G), \phi(G') \big\rangle . \tag{8.20}$$

To remove the dependency on the size of G, we normalize the count vector ϕ to a probability vector, yielding the normalized graphlet kernel

$$k(G, G') = \left\langle ||\phi(G)||_1^{-1}\phi(G), ||\phi(G')||_1^{-1}\phi(G')\right\rangle . \tag{8.21}$$

Isomorphic graphs have the same graphlet k-spectrum. Whether the reverse holds is currently not known.[4]

8.7.2 Computation

Computing Equation 8.20 or 8.21 requires counting all $\binom{|V|}{k} = O(|V|^k)$ graphlets. By sampling a sufficient number of graphlets, one can approximate the result with a given confidence. Shervashidze et al. [52] state (without proof) that the L_1 distance between the empirical and the true distribution of graphlets in a graph exceeds ϵ with probability at most δ for

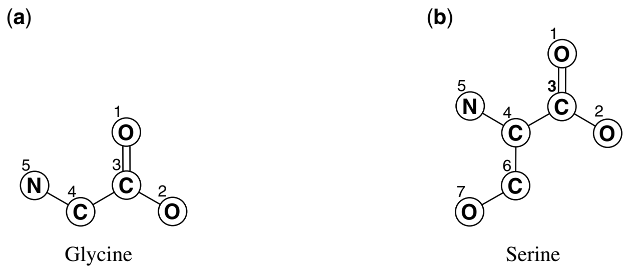
$$\left\lceil \frac{2\left(N \ln 2 + \ln(\frac{1}{\delta})\right)}{\epsilon^2} \right\rceil \tag{8.22}$$

or more samples, assuming that the number of occurrences of a graphlet (up to isomorphism) are independent and identically distributed. Here, N denotes the number of all size k graphlets. For graphs with bounded vertex degree, all connected graphlets can be enumerated in $O(|V|\, c^{k-1})$ for $k \in \{3, 4, 5\}$, where c is the maximum vertex degree in the graph [52]. The computation is based on a partitioning of the graphlets into those that contain a simple path of length $k - 1$, and those that do not.

8.7.3 Variants and Extensions

In an algebraic approach, Kondor and Borgwardt [51] define the *skew spectrum* of a graph, a fixed-size set of graph invariants. This explicit vectorial representation of a graph can be computed in time $O(|V|^6)$ for the full version with 85 components, and in time $O(|V|^3)$ for the reduced version with 49 components. Its derivation and computation are based on Kakarala's results [79] on the bispectra of functions on compact groups, and Young's orthonormal representation. The skew spectrum is limited in its representational power due to its fixed size, and in that it does not consider vertex and edge labels. In further work, they introduce the *graphlet spectrum* of a graph relative to a set of given graphlets [53]. In contrast to the graphlet kernel, the graphlet spectrum takes the relative positions of graphlets into account, and considers vertex and edge labels.

[4]This problem is related to another open problem, the *graph reconstruction problem*, which asks whether a graph can be reconstructed from the multiset of all of its subgraphs obtained by deleting one vertex. For graphs up to size $|V| = 11$, this has been verified [77], and it has been shown that almost every (random) graph can be reconstructed from three such subgraphs [78]. Note, however, that normally $k \ll |V|$.



	1	2	3	4	5	6	7
1	$\boxed{0.98}$	0.50	0.00	0.00	0.00	0.00	0.50
2	0.50	$\boxed{0.98}$	0.11	0.16	0.34	0.17	0.89
3	0.00	0.11	$\boxed{0.96}$	0.68	0.14	0.78	0.13
4	0.00	0.24	0.67	$\boxed{0.81}$	0.17	0.77	0.20
5	0.00	0.33	0.14	0.13	$\boxed{0.91}$	0.20	0.38

Pairwise vertex similarities

FIGURE 8.5 The ISOAK optimal assignment kernel [37] between the molecular structure graphs of glycine (a) and serine (b). Vertex assignments are shown boxed. Note how pairwise vertex similarities are highest in the identical parts of the graphs and slowly die off toward vertices 6 and 7.

8.8 OPTIMAL ASSIGNMENT KERNELS

Optimal assignment kernels were proposed in the context of cheminformatics[5] [34]. Their idea is to optimally assign vertices between graphs based on pairwise vertex similarities. Variants of these kernels differ in the type of pairwise vertex similarity used.

8.8.1 Definition

Let $G = (V, E)$ and $G' = (V', E')$ be two graphs, and assume without loss of generality that $|V| \leq |V'|$. Based upon a measure $k_{G,G'}$ of similarity[6] between the vertices of G and G', the *optimal assignment kernel* injectively assigns the vertices of V to vertices of V' such that the total similarity between the assigned vertices is maximized (Fig. 8.5):

$$k_{\mathrm{oa}}(G, G') = \max_{\pi} \sum_{i=1}^{|V|} k_{G,G'}(v_i, v'_{\pi(i)}). \tag{8.23}$$

[5]The idea of "aligning" two graphs has been rediscovered in various contexts, for example, in cheminformatics [80], bioinformatics [81], and applied mathematics [82].

[6]Note that we allow $k_{G,G'} : V \times V' \to \mathbb{R}_+$ to depend on G and G'.

The maximum is over all possible assignments π of the vertices in V to vertices in V', that is, all prefixes of length $|V|$ of permutations of size $|V'|$. To prevent the value of the kernel depending on the size $|V|$ of the smaller graph, one uses the *normalized optimal assignment kernel*

$$k(G, G') = \frac{k_{\text{oa}}(G, G')}{\sqrt{k_{\text{oa}}(G, G)k_{\text{oa}}(G', G')}}. \tag{8.24}$$

Whether or not k_{oa} is positive definite depends on the underlying vertex similarity $k_{G,G'}$ [38].

Choices for $k_{G,G'}(v, v')$ include the mean similarity between the neighbors of v and v' up to a given topological distance, where the influence of distant neighbors is downweighted [35], optimal assignments of the neighbors [34], and, an equilibrium definition of vertex similarity based on iterative graph similarity (iterative similarity optimal assignment kernel, ISOAK) [37].

8.8.2 Computation

Equation 8.23 can be computed in two steps: first, the matrix of pairwise vertex similarities is calculated. Then, an optimal assignment (one column assigned uniquely to each row) is computed using the Kuhn–Munkres assignment algorithm (also called Hungarian algorithm) [83–85]) in time $O(|V'|^3)$.

For graphs with bounded vertex degree, the matrix of pairwise vertex similarities can be computed in time $O(|V'|^2)$ if mean similarity or optimal assignments of neighbors are used. The ISOAK iterative graph similarity approach requires time in $O\big(|V'|^3 + |V'|^2 \log_\alpha((1-\alpha)\epsilon)\big)$, where ϵ is the desired precision and $\alpha \in (0, 1)$ controls the extent of neighbor influence.

8.8.3 Variants and Extensions

Optimal assignment kernels are not always positive definite[7] [38] because positive definite functions are not closed under the max operation [86]. Vishwanathan et al. [64] sketch a potential remedy to this problem based on an approximation of the tropical semiring via the logarithmic semiring augmented with a temperature parameter. The basic idea is to remove the max in Equation 8.23 and to exponentiate the summed terms; the result will be dominated by the value of the optimal assignment.

Optimal assignment kernels have also been combined with frequent subgraph mining [48].

[7]Note that the ISOAK graph kernel has been shown empirically to be positive definite on many data sets of molecular structure graphs for $\alpha \to 1$ [37,41].

8.9 OTHER GRAPH KERNELS

Several other approaches to graph kernels have been proposed.

Neighborhood hash kernels [57] use binary representations of discrete vertex labels. In consecutive rounds, the labels of the neighbors of each vertex and the vertex itself are combined using hash functions based on the exclusive-or and left-shift operations. This has the effect of propagating information about neighborhood structure throughout the graph. The hashed labels generated in each round are used to calculate a kernel function based on the Jaccard–Tanimoto coefficient. The procedure takes time in $O(DR\bar{c}\,|V|)$, where D is the bit length of the used binary representations, R is the number of rounds, and $\bar{c}$ is the average vertex degree, making it an essentially linear algorithm applicable to graphs with several thousand vertices.

The *edit distance* $d(G, G')$ [87,88] is the minimum number of vertex and edge insertions, deletions, and substitutions required to transform G into G' or vice versa. It requires computation time exponential in the number of vertices, but can be efficiently approximated [89]. The kernel

$$k_{G_0}(G, G') = \frac{1}{2}\big(d^2(G, G_0) + d^2(G', G_0) - d^2(G, G')\big), \tag{8.25}$$

where G_0 is fixed and takes the role of origin, is positive definite if $-d^2$ is conditionally positive definite [14]. An advantage of edit distances is their robustness against noise in the input graphs.

Complement graph kernels $k(G, G') + k(\bar{G}, \bar{G}')$ are random walk kernels k that use both the graph $G = (V, E)$ itself and its complement graph $\bar{G} = \big(V, V \times V \setminus E\big)$ [49]. Through this, they also take the absence of an edge in both graphs into account. This is relevant, for example, for the comparison of protein–protein interaction networks.

Other kernels include *fingerprint kernels* [42,43] which extract fingerprints from graphs, for example, based on common walks of length up to d, *matching-based kernels* [46] which decompose graphs into walks of fixed length, compared using set distances and the proximity space representation, convolution kernels augmented by the *context* of a substructure [45], and kernels based on spatial alignments, vertex matching, and the Jaccard index [90]. The latter are not positive definite.

8.10 APPLICATIONS IN BIO- AND CHEMINFORMATICS

Graph kernels constitute an alternative to vector-based representations of molecules (descriptors) in cheminformatics applications such as ligand-based virtual screening [91] and quantitative structure–property relationships [92]. In bioinformatics, strings as a natural representation of base pair sequences have played a more prominent role from the beginning, but graphs have gained momentum recently, mainly due to advances in systems biology, for example, protein–protein interaction networks.

Although molecular graphs have been used as examples early on, the application of graph kernels in bio- and cheminformatics is still in its beginnings. Until now, a

TABLE 8.2 Selected References by Study Type

Study Type	References
Reviews	[1,64,66,94–97]
Theoretical studies	[23,38,58,62,98,99]
Retrospective studies	[27–29,31–37,39,40,42–46,48,49,51–57,67,90,100–103]
Prospective studies	[41,93]

TABLE 8.3 Selected References by Topic

Topic	References
Absorption, bioavailability	[34–36,48]
Blood–brain barrier	[34,36,37,41]
Drug / nondrug	[37,41]
Methodology	[1,23,38,58,62,64,66,95–99]
Lead hopping	[54]
Mutagenicity, carcinogenicity	[28–30,42,42,43,45,46,48,51–57,67]
Protein binding	[31,32,36,37,39–41,44,45,47,48,54,90,93,102]
Protein function	[27,33,44,51–53,56,100]
Protein interactions	[49,101,103]
Protein structure	[57]
Protein transport	[44]
Toxicology	[28–30,34,35,37,41,42,44,47,52,55,57]

good two dozen retrospective studies (Table 8.2) have been done, covering a moderate range of topics in bio- and cheminformatics (Table 8.3), including the establishment of quantitative structure–activity and structure–property relationships, estimation of absorption, distribution, metabolism, excretion, and toxicity of compounds, and, prediction of protein function. A common theme of most of these studies is that approaches based on graph kernels were either able to achieve state-of-the-art results or surpass them. Unfortunately, the use of different retrospective validation schemes, together with problematic aspects in their design and execution, make an objective comparison of retrospective results in the literature almost impossible.

While retrospective studies paint a promising picture, there are almost no prospective applications yet. In a recent prospective study [93] in drug development, graph kernels were used together with vector-based descriptors and Gaussian process regression to discover new agonists of the peroxisome proliferator–activated receptor.

Tables 8.2 and 8.3 give an overview of published work.

8.11 SUMMARY AND CONCLUSIONS

Graph kernels allow the utilization of graph theory in a formal framework suitable for kernel-based machine learning. They have the advantage that they can potentially exploit information not available to vector-based representations alone. In many application domains, graphs are a natural representation of input data, for example, the molecular structure graph in cheminformatics. On the downside, most graph kernels are computationally demanding, with cubic or higher degree runtimes. This prevents them from being used with large graphs. They are not always interpretable, although this depends highly on the kernel.

What makes a good graph kernel? Borgwardt and Kriegel [33] have argued that a graph kernel "should be (i) a good measure of graph similarity, (ii) computable in polynomial time, (iii) positive definite, and (iv) applicable to all graphs, not just a small subset of graphs." While (i), (ii), (iii) are reasonable requirements, with (i) being somewhat vague, I disagree with (iv). From basic considerations (Section 8.1.3), a graph kernel is a trade-off between completeness (expressivity) and complexity (runtime). A kernel that exploits characteristics of specific graph classes (e.g., graphs with bounded vertex degrees, such as molecular structure graphs) can take advantage of these characteristics to achieve a better trade-off on this graph class. Restriction to a specific graph class is a way to introduce domain knowledge; it is also related to the principle stated by Vapnik [104] that "when solving a problem of interest, do not solve a more general problem as an intermediate step."

Many graph kernels count particular types of subgraphs (the features), such as walks, paths, trees, cyclic patterns, or fixed-size subgraphs. Only counting subgraphs ignores their relative position in the graph, information that can be crucial in some application domains.

Some graph kernels, for example, random walk kernels and substructure-based kernels, rely on comparing all possible subgraphs, for example, all random walks, or all subtrees of a certain type. It has been argued [46] that this might negatively affect performance due to the combinatorial growth of the number of such subgraphs, most of which will not be related with the target property. Proposed solutions include downweighting of the contribution of larger subgraphs, prior knowledge-based selection of relevant subgraphs, or, considering contextual information for limited-size subgraphs [45].

Graph kernels often use other kernels to compare vertex or edge labels, such as the *Dirac kernel* (also Kronecker kernel, $k(x, x') = 1$ if $x = x'$, and 0 otherwise). This simple approach works surprisingly well, but might reduce expressivity, and more sophisticated label comparisons might prove beneficial, depending on the application.

Certain graph kernels support only edge labels. An edge kernel $k_e : E \times E' \to \mathbb{R}$ can be extended to include information from a vertex kernel $k_v : V \times V' \to \mathbb{R}$ via

$$\tilde{k}_e\big((v_i, v_j), (v'_{i'}, v'_{j'})\big) = k_v(v_i, v'_{i'})k_e\big((v_i, v_j), (v'_{i'}, v'_{j'})\big)k_v(v_j, v'_{j'}). \tag{8.26}$$

Many theoretical properties of graph kernels have not been sufficiently investigated, for example, completeness (Section 8.1.3). It has been shown [58] that even approximating a complete graph kernel is NP-hard. This result, however, is for general graphs; it is not clear whether it holds for specialized graph classes. The related graph isomorphism problem, for example, is in P for bounded degree graphs [105].

TABLE 8.4 Implementations of Graph Kernels

Kernel	Type	Language	License	Availability	References
ISOAK	OA	Java	BSD	`mrupp.info`	[37]
Optimal assignment	OA	Java	None	`dkfz.de/mga2/people/froehlich`[a]	[34]
Marginalized	RW	Matlab	None	`kyb.tuebingen.mpg.de/bs/people/spider`[b]	[28]
Marginalized	RW	C++	LGPL	`chemcpp.sourceforge.net`	[28,30]
Geometric	RW	C++	LGPL	`chemcpp.sourceforge.net`	[58]
Tree pattern	TP	C++	LGPL	`chemcpp.sourceforge.net`	[55]
All-paths	P	Python	GPL	`mars.cs.utu.fi/PPICorpora`	[101]
Page rank	O	Java	LGPL	`www.cs.iastate.edu/~ftowfic` → BiNA	[103]
Shortest paths	P	Java	LGPL	`www.cs.iastate.edu/~ftowfic` → BiNA	[103]
Random walk	RW	Java	LGPL	`www.cs.iastate.edu/~ftowfic` → BiNA	[103]

[a] Also available at `www.ra.cs.uni-tuebingen.de/software/OAKernels/`. Contains a Java implementation of the marginalized graph kernel.
[b] Files `basic/@kernel/kmgraph.m` and `basic/@kernel/kmgraph_classical.m`.
The list is not comprehensive. OA = optimal assignment kernel, RW = random walk kernel, TP = tree pattern kernel, P = path kernel, O = other kernel.

Graph kernels have some interesting connections to other topics, for example, random walk kernels are a special case of rational kernels on weighted transducers [64], and shortest path kernels are a generalization of the Wiener index [1,106].

Further guidance is necessary with regard to the conditions under which it is advantageous to use a graph kernel, and as to which graph kernel to use. Intuitively, one should consider graph kernels when there is a natural representation of inputs as graphs. If several graph kernels match all application requirements such as the use of vertex or edge labels, one should choose the kernel that most closely matches the specialized graph class of the application. Besides that, it is suggested to try different graph kernels, possibly in combination with vector-based representations, for example, via multiple kernel learning [107].

Table 8.4 lists several available implementations of graph kernels. It is intended to provide a starting point for scientists who want to apply graph kernels in their own research. Compared to the available machine learning software (see, e.g., `www.mloss.org`), the number of readily available graph kernel implementations is rather limited.

Graph kernels constitute a promising and relatively new approach in kernel-based machine learning and its applications. A growing number of retrospective validation studies attests to their potential usefulness, often reaching or surpassing state-of-the-art performance, but application studies are still lacking. In my opinion, research in the following directions would be most useful to exploit graph kernels further: (i) the conduction of prospective application studies, (ii) the development of graph kernels (or the adaptation of existing ones) that take advantage of domain-specific graph characteristics, such as the bounded vertex degree of molecular structure graphs, (iii) investigations of theoretical graph kernel properties. In addition, an increase in available graph kernel implementations would be of great practical benefit.

ACKNOWLEDGMENTS

The author acknowledges partial support from European Union FP7-ICT programme (PASCAL2), Deutsche Forschungsgemeinschaft (DFG grant MU 987/4-2), and Institute for Pure and Applied Mathematics (IPAM, UCLA, long program on “navigating chemical compound space for materials and bio design”).

REFERENCES

1. K. Borgwardt, *Graph Kernels*, PhD thesis, Faculty for Mathematics, Informatics and Statistics, Ludwig-Maximilians-University Munich, Germany, 2007.
2. D. Bonchev, D. Rouvray, *Chemical Graph Theory. Introduction and Fundamentals*, Taylor & Francis, London, 1991.
3. J. Gasteiger, T. Engel, *Chemoinformatics*, Wiley-VCH, Weinheim, 2003.

4. R. Sharan, T. Ideker, Modeling cellular machinery through biological network comparison. *Nat. Biotechnol.* **24**(4), 427–433 (2006).
5. S. Wasserman, K. Faust, Social network analysis, in *Structural Analysis in the Social Sciences*, Vol. 8, Cambridge University Press, 1995.
6. R. Kumar, J. Novak, A. Tomkins, Structure and evolution of online social networks, in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, Philadelphia, USA, August 20–23, pp. 611–617, 2006.
7. Z. Harchaoui, F. Bach, Image classification with segmentation graph kernels, in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, Minnesota, USA, June 18–23, IEEE Computer Society, 2007.
8. M. Collins, N. Duffy, Convolution kernels for natural language, in (T. Dietterich, S. Becker, Z. Ghahramani, eds.), *Advances in Neural Information Processing Systems 14 (NIPS 2001)*, Vancouver, British Columbia, Canada, December 8–3, MIT Press, pp. 625–632, 2002.
9. F. Emmert-Streib, M. Dehmer, Networks for systems biology: conceptual connection of data and function, *IET Syst. Biol.* **5**(3), 185–207 (2011).
10. R.C. Read, D.G. Corneil, The graph isomorphism disease. *J. Graph Theor.* **1**(4), 339–363 (1977).
11. J.R. Ullman, An algorithm for subgraph isomorphism. *J. Assoc. Comput. Mach.* **23**(1), 31–42 (1976).
12. B. Zelinka, On a certain distance between isomorphism classes of graphs. *Časopis pro Pěstování Matematiky* **100**(4), 371–373 (1975).
13. M.-L. Fernández, Gabriel Valiente, A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recogn. Lett.* **22**(6–7) (2001).
14. M. Neuhaus, H. Bunke, Edit distance-based kernel functions for structural pattern classification. *Pattern Recogn.* **39**(10), 1852–1863 (2006).
15. L.G. Shapiro, Organization of relational models, in *Proceedings of the 6th International Conference on Pattern Recognition (ICPR 1982)*, Munich, Germany, October 19–22, pp. 360–365, 1982.
16. D. Gernert, Measuring the similarity of complex structures by means of graph grammars, *Bull. Eur. Assoc. Theor. Comput. Sci.* **7**, 3–9 (1979).
17. T. Hofmann, B. Schölkopf, A. Smola, A review of kernel methods in machine learning, Technical Report 156, Max-Planck-Institute for Biological Cybernetics, 2006.
18. K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, B. Schölkopf, An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Netw.* **12**(2), 181–201 (2001).
19. B. Schölkopf, A. Smola, *Learning with Kernels*, MIT Press, Cambridge, 2002.
20. J. Shawe-Taylor, Nello Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, New York, 1st edn., 2004.
21. R. Kondor, J. Lafferty, Diffusion kernels on graphs and other discrete structures, in (C. Sammut, A. Hoffmann, eds.), *Proceedings of the 19th International Conference on Machine Learning (ICML 2002)*, Sydney, Australia, July 8–12, Morgan Kaufmann, pp. 315–322, 2002.

22. R. Kondor, J.-P. Vert, Diffusion kernels, in (B. Schölkopf, K. Tsuda, J.-P. Vert, eds.), *Kernel Methods in Computational Biology*, MIT Press, Cambridge, pp. 171–191, 2004.
23. A. Smola, R. Kondor, Kernels and regularization on graphs, in (B. Schölkopf, M. Warmuth, eds.), *Learning Theory and Kernel Machines: Proceedings of the 16th Annual Conference on Learning Theory and 7th Kernel Workshop (COLT / Kernel 2003)*, Washington DC, USA, August 24–27, Vol. 2777, *Lecture Notes in Computer Science*, Springer, pp. 144–158, 2003.
24. J. Kandola, J. Shawe-Taylor, N. Cristianini, Learning semantic similarity, in (S. Becker, S. Thrun, K. Obermayer, eds.), *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, Cambridge, Massachusetts, USA, December 10–12, MIT Press, pp. 657–664, 2003.
25. T. Jaakkola, M. Diekhans, D. Haussler, A discriminative framework for detecting remote protein homologies. *J. Comput. Bio.* **7**(1–2), 95–114 (2000).
26. K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, K.-R. Müller, A new discriminative kernel from probabilistic models. *Neural Comput.* **14**(10), 2397–2414 (2002).
27. K. Tsuda, T. Kin, K. Asai, Marginalized kernels for biological sequences. *Bioinformatics* **18**(7), S268–S275 (2002).
28. H. Kashima, K. Tsuda, A. Inokuchi, Kernels for graphs, in (B. Schölkopf, K. Tsuda, J.-P. Vert, eds.), *Kernel Methods in Computational Biology*, MIT Press, Cambridge, pp. 155–170, 2004.
29. H. Kashima, K. Tsuda, A. Inokuchi, Marginalized kernels between labeled graphs, in (T. Fawcett, N. Mishra, eds.), *Proceedings of the 20th International Conference on Machine Learning (ICML 2003)*, Washington DC, USA, August 21–24, Menlo Park, CA, AAAI Press, pp. 321–328, 2003.
30. P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, J.-P. Vert, Extensions of marginalized graph kernels, in (C. Brodley, ed.), *Proceedings of the 21st International Conference on Machine Learning (ICML 2004)*, Banff, Alberta, Canada, July 4–8, Omnipress, Madison, WI, USA, pp. 552–559, 2004.
31. T. Horváth, T. Gärtner, S. Wrobel, Cyclic pattern kernels for predictive graph mining, in (R. Kohavi, J. Gehrke, W. DuMouchel, J. Ghosh, eds.), *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004)*, Seattle, Washington, USA, August 22–25, ACM Press, pp. 158–167, 2004.
32. T. Horváth, Cyclic pattern kernels revisited, in (J. Carbonell, J. Siekmann, eds.), *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2005)*, Hanoi, Vietnam, May 18–20, Vol. 3518, *Lecture Notes in Computer Science*, Springer, pp. 791–801, 2005.
33. K. Borgwardt, H.-P. Kriegel, Shortest-path kernels on graphs, in *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005)*, Houston, Texas, USA, November 27–30, IEEE Computer Society, pp. 74–81, 2005.
34. H. Fröhlich, J. Wegner, F. Sieker, A. Zell, Optimal assignment kernels for attributed molecular graphs, in (L. de Raedt, S. Wrobel, eds.), *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*, Bonn, Germany, August 7–11, Madison, WI, USA, pp. 225–232, Omnipress, 2005.
35. H. Fröhlich, J. Wegner, A. Zell, Assignment kernels for chemical compounds, in *Proceedings of the 2005 International Joint Conference on Neural Networks (IJCNN 2005)*, Montréal, Canada, July 31–August 4, IEEE Computer Society, pp. 913–918, 2005.

36. H. Fröhlich, J. Wegner, F. Sieker, A. Zell, Kernel functions for attributed molecular graphs: a new similarity-based approach to ADME prediction in classification and regression, *QSAR Comb. Sci.* **25**(4), 317–326 (2006).
37. M. Rupp, E. Proschak, G. Schneider, Kernel approach to molecular similarity based on iterative graph similarity. *J. Chem. Inform. Model.* **47**(6), 2280–2286 (2007).
38. J.-P. Vert, The optimal assignment kernel is not positive definite, Technical Report HAL-00218278, Centre for Computational Biology, Mines ParisTech, Paris, France, 2008.
39. N. Fechner, A. Jahn, G. Hinselmann, A. Zell, Atomic local neighborhood flexibility incorporation into a structured similarity measure for QSAR, *J. Chem. Inform. Model.* **49**(3), 549–560 (2009).
40. A. Jahn, G. Hinselmann, N. Fechner, A. Zell, Optimal assignment methods for ligand-based virtual screening, *J. Cheminform.* **1**(14) (2009).
41. M. Rupp, *Kernel Methods for Virtual Screening*, PhD thesis, Johann Wolfgang Goethe-University, Frankfurt am Main, Germany, 2009.
42. L. Ralaivola, S. Swamidass, H. Saigo, P. Baldi, Graph kernels for chemical informatics. *Neural Netw.* **18**(8), 1093–1110 (2005).
43. J. Swamidass, J. Chen, J. Bruand, P. Phung, L. Ralaivola, P. Baldi, Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. *Bioinformatics* **21**(Suppl. 1), i359–i368 (2005).
44. S. Menchetti, F. Costa, P. Frasconi, Weighted decomposition kernels, in (L. de Raedt, S. Wrobel, eds.), *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*, Bonn, Germany, August 7–11, Omnipress, Madison, WI, USA, pp. 585–592, 2005.
45. A. Ceroni, F. Costa, P. Frasconi, Classification of small molecules by two- and three-dimensional decomposition kernels. *Bioinformatics* **23**(16), 2038–2045 (2007).
46. A. Woźnica, A. Kalousis, M. Hilario, Matching based kernels for labeled graphs, in (T. Gärtner, G. Garriga Thorsten Meinl, eds.), *Proceedings of the International Workshop on Mining and Learning with Graphs (MLG 2006)*, Berlin, Germany, September 18, pp. 97–108, 2006.
47. A. Smalter, J. Huan, G. Lushington, GPM: a graph pattern matching kernel with diffusion for chemical compound classification, in *Proceedings of the 8th IEEE International Conference on Bioinformatics and Bioengineering (BIBE 2008)*, Athens, Greece, October 8–10, IEEE Computer Society, 2008.
48. A. Smalter, J. Huan, G. Lushington, Chemical compound classification with automatically mined structure patterns, in (A. Brazma, S. Miyano, T. Akutsu, eds.), *Proceedings of the 6th Asia-Pacific Bioinformatics Conference (APBC 2008)*, Kyoto, Japan, January 14–17, Imperial College Press, pp. 39–48, 2008.
49. K. Borgwardt, H.-P. Kriegel, V. Vishwanathan, N. Schraudolph, Graph kernels for disease outcome prediction from protein–protein interaction networks, in (R. Altman, K. Dunker, L. Hunter, T. Murray, T. Klein, eds.), *Proceedings of the 12th Pacific Symposium on Biocomputing (PSB 2007)*, Maui, Hawaii, USA, January 3–7, pp. 4–15, 2007.
50. F.R. Bach, Graph kernels between point clouds, in (A. McCallum, S. Roweis, eds.), *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, Helsinki, Finland, July 5–9, Omnipress, pp. 25–32, 2008.

51. R. Kondor, K.M. Borgwardt, The skew spectrum of graphs, in (A. McCallum, S. Roweis, eds.), *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, Helsinki, Finland, July 5–9, Omnipress, pp. 496–503, 2008.
52. N. Shervashidze, V. Vishwanathan, T. Petri, K. Mehlhorn, K. Borgwardt, Efficient graphlet kernels for large graph comparison, in (D. van Dyk, M. Welling, eds.), *Proceedings of the 12th International Workshop on Artificial Intelligence and Statistics (AISTATS 2009)*, Clearwater Beach, Florida, USA, April 16–18, pp. 488–495, 2009.
53. R. Kondor, N. Shervashidze, K.M. Borgwardt, The graphlet spectrum, in (L. Bottou, M. Littman, eds.), *Proceedings of the 26th International Conference on Machine Learning (ICML 2009)*, Montreal, Quebec, Canada, June 14–18, Omnipress, pp. 529–536, 2009.
54. A. Demco, *Graph Kernel Extensions and Experiments with Application to Molecule Classification, Lead Hopping and Multiple Targets*. PhD thesis, School of Electronics and Computer Science, University of Southampton, England, 2009.
55. P. Mahé, J.-P. Vert, Graph kernels based on tree patterns for molecules. *Mach. Learn.* **75**(1), 3–35 (2009).
56. N. Shervashidze, K. Borgwardt, Fast subtree kernels on graphs, in (Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, A. Culotta, eds.), *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, Vancouver, Canada, December 7–12, MIT Press, pp. 1660–1668, 2009.
57. S. Hido, H. Kashima, A linear-time graph kernel, in *Proceedings of the 9th IEEE International Conference on Data Mining (ICDM 2009)*, Miami, Florida, USA, December 6–9, IEEE Computer Society, 2009.
58. T. Gärtner, P. Flach, S. Wrobel, On graph kernels: hardness results and efficient alternatives, in *Learning Theory and Kernel Machines* (B. Schölkopf, M. Warmuth, eds.), *Proceedings of the 16th Annual Conference on Learning Theory and 7th Kernel Workshop (COLT/Kernel 2003)*, Washington DC, USA, August 24–27, Vol. 2777, *Lecture Notes in Computer Science*, Springer, pp. 129–143, 2003.
59. D. Johnson, The NP-completeness column: an ongoing guide. *J. Algorithms* **2**(4), 393–405 (1981).
60. D. Johnson, The NP-completeness column. *ACM Trans. Algorithms* **1**(1), 160–176 (2005).
61. D. Johnson, The NP-completeness column: an ongoing guide. *J. Algorithms* **5**(1), 147–160 (1984).
62. D. Haussler, Convolution kernels on discrete structures, Technical Report UCSC-CRL-99-10, Department of Computer Science, University of California at Santa Cruz, California, USA, 1999.
63. C. Watkins, Dynamic alignment kernels, Technical Report CSD-TR-98-11, Royal Holloway, London, England, 1999.
64. V. Vishwanathan, N. Schraudolph, R. Kondor, K. Borgwardt, Graph kernels. *J. Mach. Learn. Res.* **11**(4), 1201–1242 (2010).
65. T. Gärtner, Exponential and geometric kernels for graphs, in *Neural Information Processing Systems (NIPS) Workshop on Unreal Data: Principles of Modeling Nonvectorial Data*, 2002.
66. T. Gärtner, A survey of kernels for structured data. *ACM SIG Knowledge Discov. Data Mining Explorations Newsletter* **5**(1), 49–58 (2003).

67. P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, J.-P. Vert, Graph kernels for molecular structure–activity relationship analysis with support vector machines. *J. Chem. Inform. Model.* **45**(4), 939–951 (2005).
68. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, 3rd edn., MIT Press, Cambridge, 2009.
69. J.Y. Yen, Finding the k shortest loopless paths in a network. *Manag. Sci.* **17**(11), 712–716 (1971).
70. E.L. Lawler, A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Manag. Sci.* **18**(7), 401–405 (1972).
71. B. Weisfeiler, A.A. Lehman, A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauch.-Techn. Inform.* **2**(9), 12–16 (1968).
72. M. Zaki, Efficiently mining frequent trees in a forest: algorithms and applications. *IEEE Trans. Knowledge Data Eng.* **17**(8), 1021–1035 (2005).
73. J. Flum, M. Grohe, The parameterized complexity of counting problems. *SIAM J. Comput.* **33**(4), 892–922 (2004).
74. H. Bodlaender, A tourist guide through treewidth. *Acta Cybernetica* **11**(1–2), 1–21 (1993).
75. M. Plotkin, Mathematical basis of of ring-finding algorithms in CIDS. *J. Chem. Doc.* **11**(1), 60–63 (1971).
76. P. Gleiss, P. Stadler, Relevant cycles in biopolymers and random graphs, in *Proceedings of the 4th Slovene International Conference in Graph Theory*, Lake Bled, Slovenia, June 28–July 2, 1999.
77. B.D. McKay, Small graphs are reconstructible. *Austral. J. Combin.* **15**, 123–126 (1997).
78. B. Bollobás, Almost every graph has reconstruction number three. *J. Graph Theor.* **14**(1), 1–4 (1990).
79. R. Kakarala, A group-theoretic approach to the triple correlation, in *Proceedings of the IEEE Signal Processing Workshop on Higher-Order Statistics (SPW-HOS 1993)*, South Lake Tahoe, California, USA, June 7–9, IEEE, pp. 28–32, 1993.
80. N. Weskamp, E. Hüllermeier, D. Kuhn, G. Klebe, Graph alignments: a new concept to detect conserved regions in protein active sites, in (R. Giegerich, J. Stoye, eds.), *Proceedings of the German Conference on Bioinformatics (GCB 2004)*, Bielefeld, Germany, October 4–6, Gesellschaft für Informatik, pp. 131–140, 2004.
81. J. Berg, M. Lässig, Local graph alignment and motif search in biological networks. *Proc. Natl. Acad. Sci. U.S.A.* **101**(41), 14689–14694 (2004).
82. M. Dehmer, F. Emmert-Streib, J. Kilian, A similarity measure for graphs with low computational complexity. *Appl. Math. Comput.* **182**(1), 447–459 (2006).
83. H. Kuhn, The Hungarian method for the assignment problem, *Bull. Am. Math. Soc.* **61**, 557–558 (1955).
84. J. Munkres, Algorithms for the assignment and transportation problems. *J. Soc. Indus. Appl. Math.* **5**(1), 32–38 (1957).
85. F. Bourgeois, J.-C. Lassalle, An extension of the Munkres algorithm for the assignment problem to rectangular matrices. *Commun. ACM* **14**(12), 802–804 (1971).
86. C. Berg, J. Christensen, P. Ressel, *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*, Springer, 1984.

87. V. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals. *Sov. Phys. Doklady* **10**(8), 707–710 (1966).

88. D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, Cambridge, 1997.

89. M. Neuhaus, H. Bunke, An error-tolerant approximate matching algorithm for attributed planar graphs and its application to fingerprint classification, in (A. Fred, T. Caelli, R. Duin, A. Campilho, D. de Ridder, eds.), *Proceedings of the 10th Joint IAPR International Workshops on Structural, Syntactic, and Statistical Pattern Recognition (SSPR 2004, SPR 2004)*, Lisbon, Portugal, August 18–20, Vol. 3138, *Lecture Notes in Computer Science*, Springer, pp. 180–189, 2004.

90. J. Mohr, B. Jain, K. Obermayer, Molecule kernels: a descriptor- and alignment-free quantitative structure–activity relationship approach. *J. Chem. Inform. Model.* **48**(9), 1868–1881 (2008).

91. D. Douguet, Ligand-based approaches in virtual screening. *Curr. Comput.-Aided Drug Design* **4**(3), 180–190 (2008).

92. C. Selassie, History of quantitative structure–activity relationships, in (D. Abrahams, ed.), *Burger's Medicinal Chemistry and Drug Discovery*, Wiley, Vol. 1, Chapter 1, pp. 1–48, 6th edn., 2003.

93. M. Rupp, T. Schroeter, R. Steri, H. Zettl, E. Proschak, K. Hansen, O. Rau, O. Schwarz, L. Müller-Kuhrt, M. Schubert-Zsilavecz, K.-R. Müller, G. Schneider, From machine learning to natural product derivatives selectively activating transcription factor PPARγ. *ChemMedChem* **5**(2), 191–194 (2010).

94. T. Gärtner, Q.Viet Le, A. Smola, A short tour of kernel methods for graphs, Technical Report, 2006.

95. T. Gärtner, T. Horváth, Q.Viet Le, A. Smola, S. Wrobel, Kernel methods for graphs, in *Mining Graph Data* (D. Cook, L. Holder, eds.), Wiley, pp. 253–282, 2007.

96. P. Mahé, J.-P. Vert, Virtual screening with support vector machines and structure kernels, Technical Report HAL-00166188, Ecole des Mines de Paris, Center for Computational Biology, 2007.

97. T. Gärtner, *Kernels for Structured Data*, Number 72 in Machine Perception and Artificial Intelligence, World Scientific Publishing, 2009.

98. J. Ramon, T. Gärtner, Expressivity versus efficiency of graph kernels, in (L. de Raedt, T. Washio, eds.), *Proceedings of the 1st International Workshop on Mining Graphs, Trees and Sequences (MGTS 2003)*, Cavtat-Dubrovnik, Croatia, September 22–23, pp. 65–74, 2003.

99. V. Vishwanathan, K. Borgwardt, N. Schraudolph, Fast computation of graph kernels, in (B. Schölkopf, J. Platt, T. Hofmann, eds.), *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, Vancouver, Canada, December 4–7, MIT Press, pp. 1449–1456, 2006.

100. K. Borgwardt, C.S. Ong, S. Schönauer, V. Vishwanathan, A. Smola, H.-P. Kriegel, Protein function prediction via graph kernels, in (H.V. Jagadish, D. States, B. Rost, eds.), *Proceedings of the 13th International Conference on Intelligent Systems for Molecular Biology (ISMB 2005)*, Detroit, USA, June 25–29, Vol. 21 (Suppl. 1) *Bioinformatics*, Oxford, pp. i47–i56, 2005.

101. A. Airola, S. Pyysalo, J. Björne, T. Pahikkala, F. Ginter, T. Salakoski, A graph kernel for protein–protein interaction extraction, in (D. Demner-Fushman, S. Ananiadou,

B. Cohen, J. Pestian, J. Tsujii, B. Webber, eds.), *Proceedings of the ACL-08/HLT Workshop on Current Trends in Biomedical Natural Language Processing (BioNLP 2008)*, Columbus, Ohio, USA, June 19, Association for Computational Linguistics, pp. 1–9, 2008.

102. T. Fober, M. Mernberger, V. Melnikov, R. Moritz, E. Hüllermeier, Extension and empirical comparison of graph-kernels for the analysis of protein active sites, in (M. Hartmann, F. Janssen, eds.), *Joint Workshop on Lernen, Wissen, Adaptivität (LWA 2009)*, Darmstadt, Germany, September 21–23, Technical University of Darmstadt, pp. 30–36, 2009.
103. F. Towfic, M. Heather West Greenlee, V. Honavar, Aligning biomolecular networks using modular graph kernels, in S.L. Salzberg, T. Warnow, eds.), *Proceedings of the 9th International Workshop on Algorithms in Bioinformatics (WABI 2009)*, Philadelphia, Pennsylvania, USA, September 12–13, Vol. 5724, *Lecture Notes in Bioinformatics*, pp. 345–361, 2009.
104. V. Vapnik, *The Nature of Statistical Learning Theory*, 1st edn., Springer, New York, 1995.
105. E. Luks, Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Comput. Syst. Sci.* **25**(1), 42–65 (1982).
106. H. Wiener, Structural determination of paraffin boiling points. *J. Am. Chem. Soc.* **69**(1), 17–20 (1947).
107. S. Sonnenburg, G. Rätsch, C. Schäfer, B. Schölkopf, Large scale multiple kernel learning. *J. Mach. Learn. Res.* **7**(7), 1531–1565 (2006).

9

NETWORK-BASED INFORMATION SYNERGY ANALYSIS FOR ALZHEIMER DISEASE

XUEWEI WANG, HIROSHA GEEKIYANAGE, AND CHRISTINA CHAN

9.1 INTRODUCTION

Alzheimer's disease (AD) is a progressive neurodegenerative disorder characterized by amyloid beta plaques and neurofibrillary tangles. It is not only the principal cause of dementia in the United States but also one of the fastest growing diseases in the developed countries [1]. Currently, there are over 4 million Americans diagnosed with AD and the number is estimated to double during the next 25 years [1]. In an AD brain, the cortex shrivels damaging the areas of cognition, planning, and memory [2]. Further, the hippocampus of the cortex shrinks hindering formation of new memory. The molecular mechanisms underlying the pathologies of AD are being uncovered [3,4]. Thus far, familial AD ($<$60 years old) which accounts for less than 5% of AD cases is due to mutations in amyloid precursor protein (APP), presenilin-1 and presenilin-2, whereas sporadic AD ($>$65 years old) which accounts for more than 95% cases of AD, is genetically linked to apolipoprotein E isoform 4. Despite these recent progresses in the characterization of the pathologies of AD, existing treatments for AD are far from satisfactory [5]. A more comprehensive understanding of the molecular mechanisms underlying AD is needed for better identification of molecular targets as well as development of more effective therapeutics.

In recent years, network-based methods have been widely applied to identify biomarkers or targets for various diseases [6], typically by integrating gene expression data and available physical interaction data (i.e., protein–protein interaction).

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

Studies have shown that disease genes oftentimes cooperate with each other within the same biological modules [7], that is, signaling pathways, regulatory modules, protein complexes, or protein interaction subnetworks, suggesting a strong association or interaction between genes/proteins in rendering the disease. Integrating either physical or functional interaction data into a network has become a powerful tool for identifying novel genes involved in complex diseases, that is, cerebral ataxias [8], breast cancer [9], glioblastoma [10], and so on. In the case of AD, protein interaction data specific to AD have accumulated in the literature [11], generated through high-throughput experiments [12]. Concomitantly, computational analyses have shown that the integration of gene expression data with physical and functional interaction data can be useful in characterizing the pathways and the cross-talk [13,14] or in prioritizing candidate genes [15–17] involved in AD.

A majority of the existing network-based studies of diseases integrate a global protein–protein interaction (PPI) network with the gene expression data [18], whereby the genes in the microarray datasets are mapped to the proteins in the network. Then differential analysis (i.e., Student's *t*-test) and correlation analysis (i.e., Pearson correlation [19] or mutual information [20]) between the genes are applied to identify disease-specific protein networks, a subset of the global PPI network. For example, in a recent network-based AD study, only the genes that were differentially expressed and had high correlations and physical interactions remained in the final network [13]. However, the applications of correlation analysis for identifying a phenotype-specific PPI network, thus far, have not directly incorporated the phenotype into the analysis. Instead, typically a network of gene pairs (highly correlated and physically interacting for each of the conditions) is built and compared to identify the interactions that are specific to a condition or phenotype. Consequently, these methods become sensitive to the quality of data. Since the noise level and the size of the samples can affect the correlation calculation for each of the conditions, it is difficult to determine whether the differences in the networks across conditions are real changes that provide insight into the mechanisms or simply an artifact due to the size or noise levels.

Alternatively, some computational approaches have been developed to select sets of gene pairs relevant to a phenotype based on classification models, such as support vector machine [21,22], decision tree [23], and probabilistic model [24]. Intuitively, if a phenotype prediction based on a pair of genes performs better than that based on either one of the genes then the pair of genes is suggested to have cooperative effects on the phenotype. However, such classification methods fail to distinguish the cooperative effects of the genes pairs from the independent contributions of the individual genes [25]. To address this drawback, we present an information theoretical method that distinguishes the difference between the cooperative versus individual effects of the genes.

Molecular machinery in complex diseases usually involves multiple factors, many of which function cooperatively, that is, synergistically. Indeed, the factors or pathways that function synergistically in the development or pathophysiologyy of AD need to be characterized [26,27]. Some studies have uncovered the genes that work synergistically to increase the risk of AD progression, that is, APOE4 and BCHE-K [28], or HO-1 and tau genetic variants [29]. A method that can identify these synergistic interactions between genes would provide information complementary to existing approaches (i.e., correlation analysis), and help to enhance our understanding of the complex mechanisms underlying AD. Computational approaches for systematic assessment of synergy were first proposed in neuroscience, where the goal was to understand the neuron code by evaluating the strength of correlations between the neurons upon activation by stimuli [30,31]. More recently the concept of information synergy has been applied to the field of systems biology [32–34]. Investigators developed an information theoretical measure of synergy from discretized gene expression data, and applied this measure to identify cooperative gene interactions associated with neural interconnectivity [32] and prostate cancer development [33]. More recently, the concept of synergy and the information theoretical measure of synergy have been applied directly to continuous gene expression data [25].

We adopt this concept of information synergy to evaluate the synergistic effects of two genes on a phenotype (AD in this case). For two genes in a multivariate system, their synergistic effect on a phenotype is defined as the gain in the "mutual information" over the sum of the information provided by each gene on a phenotype. A positive synergy denotes that two genes regulate a phenotype, either cooperatively (e.g., coactivating) or antagonistically (e.g., competitive inhibiting). Thus, one can predict the phenotype from either of the two genes at a certain confidence level, whereas knowing both genes brings additional information, which further enhances the confidence of the prediction. Negative synergy on the other hand denotes redundancy, thus knowing both genes brings redundant information to the prediction. Zero synergy denotes that at least one of the two genes does not affect the phenotype, and therefore brings neither additional nor redundant information to the prediction of the phenotype.

In this chapter, we introduce an integrative methodology to build a protein network based on information synergy analysis that is specific to AD. First, we collected a publicly available microarray dataset for AD and mapped to a global network collected from experimental PPI databases. Next, we assess the synergistic effects between the genes that are mapped onto the PPI network. Unlike other computational methods used to identify gene interactions, the fundamental concept of synergy is to identify the cooperative gene interactions responsible for the phenotype. Finally, with the identified synergistic gene pairs, a synergy network is built which is a subset of the global PPI network. Topological analyses reveal the structural characteristics of the network while the hub genes provide insights into potential mechanism(s) involved in the induction of the phenotype. Further, a comparison with differential expression or differential correlation analyses indicates that the information synergy approach could provide complementary information to these traditional approaches.

9.2 DATASETS AND METHODS

9.2.1 Microarray Dataset and Protein–Protein Interaction Data

The AD microarray dataset (GSE5281) in GEO database [35] was used for the analysis, including 13 control samples and 10 AD samples collected from the entorhinal cortex region in the brains of AD patients with a mean age of 79.8 ± 9.1 years. The intensities of the probe-sets were first normalized by robust microarray adjustment (RMA) and logarithmized to base two. The expressions of the multiple probes for the same genes on the microarray were then averaged. Experimental PPI data was collected from two major protein interaction databases for human, including BioGRID [36] and HPRD [37]. Duplicate and self-interactions were removed from the analysis.

9.2.2 Calculation of the Synergy Scores of Gene Pairs

An information theory-based score was calculated to quantify the synergy between the genes [34]. Given two genes, G1 and G2, and a phenotype P, the synergy score between G1 and G2 with respect to the phenotype P is defined as

$$\mathrm{Syn}(\mathrm{G1}, \mathrm{G2}; \mathrm{P}) = \mathrm{I}(\mathrm{G1}, \mathrm{G2}; \mathrm{P}) - [\mathrm{I}(\mathrm{G1}; \mathrm{P}) + \mathrm{I}(\mathrm{G2}; \mathrm{P})]$$

where I(G1;P) is the mutual information between G1 and P, I(G2;P) is the mutual information between G2 and P, and I(G1,G2;P) is the mutual information between (G1,G2) and P. This equation reflects the definition of synergy, the additional contribution provided by the “whole” as compared to the sum of the contributions of the individual “parts.” Mutual information (I) was calculated using a clustering-based method from continuous data [25].

The synergy scores range from -1 to 1. A positive synergy score indicate that two genes jointly provide additional information on the phenotype, a negative synergy score indicate that the two genes provide redundant information about the phenotype, and a zero score indicate that the two genes provide no additional information about the phenotype.

9.2.3 Permutation Test to Evaluate the Significance of the Synergy

A permutation test was performed to assess the statistical significance of the information synergy scores of the gene pairs. The phenotypes, that is, AD versus normal, were randomly shuffled to be uncorrelated with the gene expression profiles. The information synergy scores of the genes were then recalculated based on the shuffled phenotype. This process was repeated 100 times to estimate the distribution of random information synergy scores based on kernel-density approach, and the p values of the real information synergy scores were then calculated for each gene pair based on the distribution of random information synergy score. Finally, Benjamin–Hochberg false discovery rate procedure [38] was performed to adjust the p values for all the gene pairs and thereby control the expected false discoveries. The p value cutoff was set at 0.05.

9.2.4 Characterization of the Network Topology

Structural network theory and the characterization of network topology have contributed to our understanding of the architectures of networks [39]. Structural network analyses have revealed that existing biological networks, including gene regulatory networks, metabolic networks, signaling networks, and PPI networks, are very different from randomly organized networks. These biological networks have a "scale-free" feature that is characterized by few hub nodes that contain many connections and many nodes with very few connections [40]. Further, structural network analysis has contributed to our understanding of the functional organizations in biological systems. The hub nodes in scale-free networks have been shown to play essential roles in certain biological systems, for example, the essential proteins (critical for cell viability) are more significantly over-represented in the hubs than in the nonhub nodes in the yeast PPI network [41]. In addition to exploring the fundamental principles of biological networks, structural analyses of human metabolic networks have contributed insights into disease comorbidity [42].

In our analysis, the synergy network was built with gene pairs that have statistically significant synergy scores and physical interaction in PPI network. The network composed of nodes that represented the genes, and edges that represented the synergy of the gene pairs. Topological analysis of the networks obtained from information synergy, differential expression analysis, and differential correlation, was performed to reveal their topological characteristics, including the distribution of node degrees and the distribution of shortest path length. Degree distribution provides a distribution of the number of edges associated with the nodes. Shortest path length is the lowest number of edges that connect two nodes, and is measured using a breadth-fast search algorithm [43].

9.3 RESULTS

9.3.1 Information Synergy for Simulated Gene Pairs

Existing network-based approaches typically incorporate differential analysis or correlation analysis to identify the genes or gene pairs specific to phenotypes or diseases. In order to illustrate how information synergy analysis differs with differential analysis and correlation analysis, we simulated scenarios where the genes vary in differential expression and correlation, and calculated the information synergy for each of the gene pairs. Six representative scenarios were simulated, including (1) both genes differentially expressed between phenotypes, and correlated in both phenotypes; (2) both genes differentially expressed, and have no correlation with each phenotype; (3) both genes are not differentially expressed, but correlated in both phenotype; (4) both genes are not differentially expressed, and have no correlation with each phenotype; (5) both genes are not differentially expressed, but similarly correlated in each phenotype; (6) both genes are not differentially expressed, but differentially correlated between phenotypes. The joint expression patterns of these six scenarios are provided in Figure 9.1, see panels a–f, respectively.

The information synergy scores for each simulated gene pair is also provided in Figure 9.1. The observations in Figure 9.1 are consistent with the definition of information synergy. For example, in panels a and b, each gene is differentially expressed

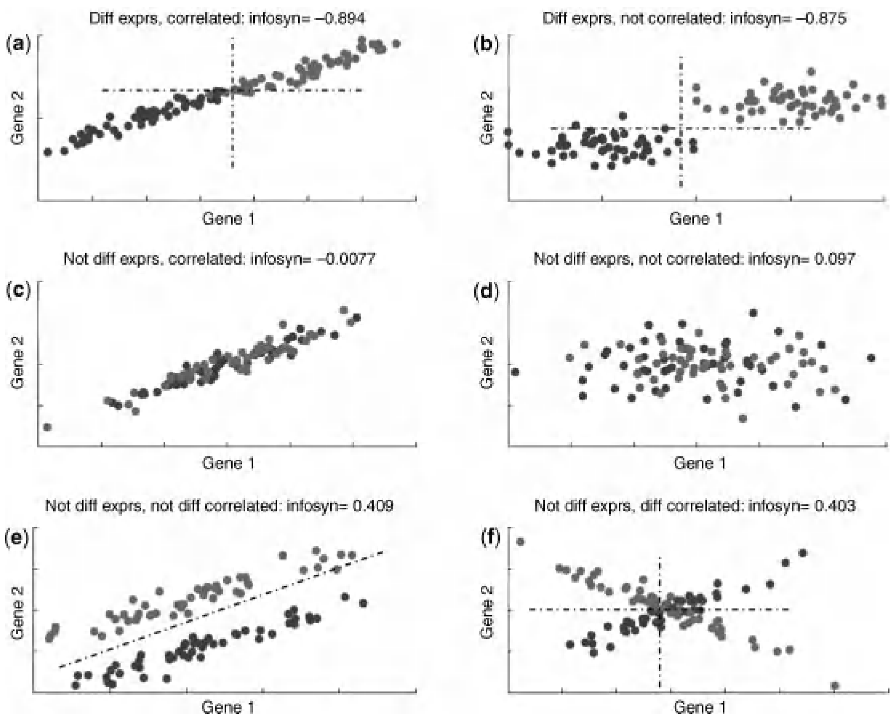


FIGURE 9.1 Information synergy scores for simulated gene pairs.

and the phenotype could be distinguished by either of the genes (as indicated by dash lines), thus they provide redundant information on the phenotype, indicating their information synergy is significantly negative. Each of the genes in panel c and d cannot individually distinguish the phenotype, and taking them together does not help to classify the phenotypes, thus the information synergy of each of these two genes is around zero. In contrast, each of the genes in panel e and f individually is not predictive of the phenotype, but their joint expression pattern can clearly discriminate the phenotype (i.e., as indicated by dash line in panels e and f). This indicated that the two genes in panels e or f together provide more information than the sum of the information provided by the individual genes. Thus, their information synergy is significantly positive. Therefore, four of these six patterns (pattern a, b, e, and f) would be of interest in further investigations since the genes in each of these four patterns are predictive of the phenotypes.

Generally, differential analysis can help to identify the genes with patterns a or b, and correlation analysis can help to identify the genes with pattern f. However, either of these two approaches cannot capture the genes with pattern e; in which the genes are not differentially expressed and the correlation of genes is not altered between the phenotypes. In contrast, synergy analysis allows one to capture all four patterns, by accounting for gene pairs with both positive and negative synergies. Note that there are many other possible joint expression patterns for the genes in term of their expression and correlation with the phenotypes. Nevertheless, the observations from these six representative patterns suggested that synergy analysis could be a promising alternative method in network-based analysis in the pursuing disease biomarkers or targets. Next, information synergy analysis was applied to identify potential mechanisms underlying AD through integrating gene expression data and protein–protein interaction network.

9.3.2 Topological Characteristics of Synergy Network for AD

By mapping the gene expression data (from GSE5281 in GEO database) to the PPI data collected from BioGRID and HPRD, we obtained a global PPI network for AD. For each connection in this global network, we calculated the information synergy and evaluated its statistical significance based upon permutation test. Finally, a subnetwork consisting of genes with significant information synergy (either positive or negative) was obtained (noted as "synergy network" thereafter), which is composed of 3518 genes with 4819 connection edges (511 with positive and 4308 with negative information synergy) as shown in Figure 9.2. We characterized the topology of the synergy network and compared with the global PPI network and the networks derived from differential expression (herein denoted as DiffExprs network) and differential correlation (herein denoted as DiffCorr network) analyses. The statistics for each network are listed in Table 9.1. The distribution of the lengths of shortest paths between nodes and the node degrees are shown in Figure 9.3.

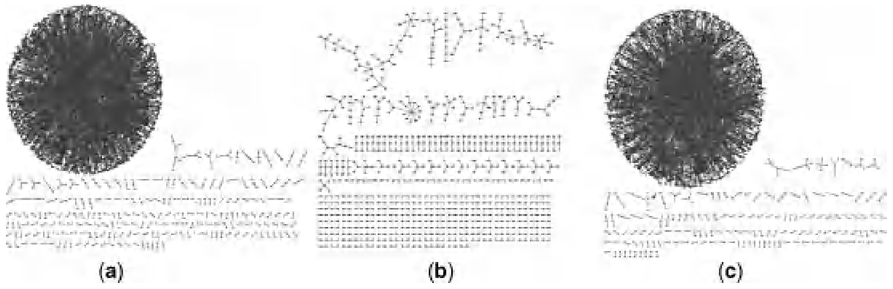


FIGURE 9.2 Synergy network (a) all pairs (3518 genes with 4819 connections); (b) pairs with positive synergy (780 genes with 511 connections); and (c) pairs with negative synergy (3083 genes with 4308 connections).

TABLE 9.1 Topological Statistics for Synergy Network, DiffExprs Network, DiffCorr Network, and Global PPI Network

Network	No. of Nodes	No. of Connections	No. of Components	Characteristic Path Length
Synergy network	3,518	4,819	213	5.70
DiffExprs network	873	1,259	52	5.24
DiffCorr network	1,985	1,744	353	9.40
Global PPI network	10,226	49,621	84	4.02

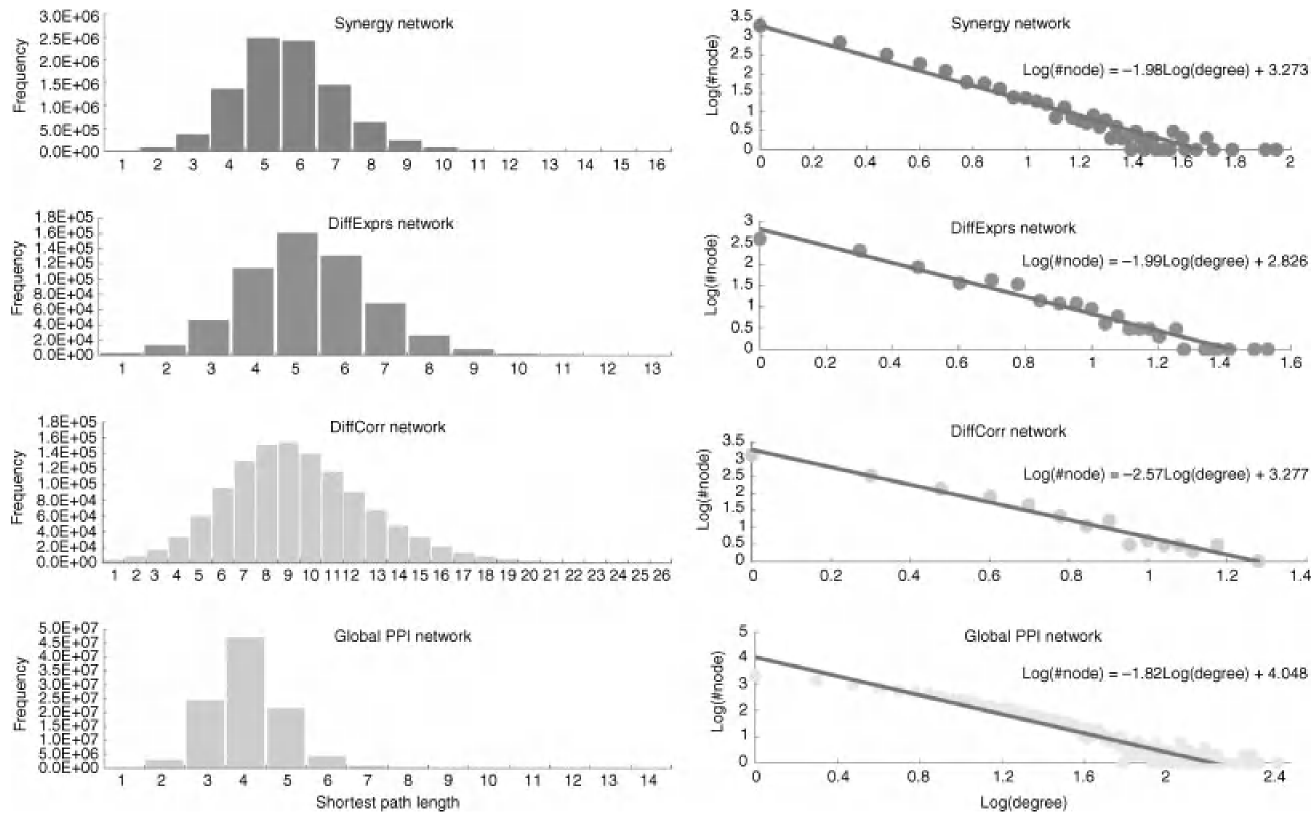


FIGURE 9.3 Distribution of shortest path length and node degree for synergy network, DiffCorr network, DiffExprs network, and global PPI network.

The synergy network is characterized by relatively short path lengths, ranging from 2 to 10 (Fig. 9.3), while the characteristic path length, or average diameter, of the network is 5.70 (Table 9.1). Therefore, the synergy network exhibits small-world network characteristics as in real biological networks [44], suggesting that communication between the genes is relatively fast. The node degree distribution, $P(k)$, describes the likelihood that a randomly selected node has k links to its neighbors in the network. A power–law distribution refers to the function $P(k) \sim k^{-\gamma}$, where k is the degree and γ is the degree exponent, and in most biological, scale-free networks γ ranges around 2 and 3 [44]. The degree distribution of our synergy network of AD is shown in Figure 9.3, and $\gamma \approx 1.98$, suggesting the synergy network, similar to other biological networks, is scale-free.

The scale-free nature of synergy network indicated that the majority of the genes are sparsely connected, while few of the genes (hubs) are connected to many other genes and may play important roles in sustaining the integrity of the network, suggesting their potential importance in the induction of the phenotype. In summary, the topology of the synergy network differs from that of a random network, typified by a bell-like Poisson distribution for node degrees, and suggests the existence of hub genes.

As shown in Figure 9.3 and Table 9.1, synergy network shares similar topological characteristics with the other three networks, that is, all the node degrees in these networks follow power–law distribution, with slightly different exponentials. Note that the number of connected components and average shortest path length varies across these four networks. For example, the global PPI network is much denser than the other networks, making the characteristic path length much shorter, and indicates the proteins in this network may communicate more efficiently. Meanwhile, the edges in the DiffCorr network are more sparsely connected (as indicated by the number of components), leading to a longer characteristic path length.

9.3.3 Differential Expression and Correlation Patterns for the Pairs in Synergy Network

In the simulation analysis as described above, we demonstrated that information synergy could capture the genes with joint expression patterns that could not be identified by either differential or correlation analyses. To confirm whether this observation holds with the AD dataset, we proceeded to investigate the differential expressions and correlations of the genes for each connection in the AD synergy network, which is summarized in Figure 9.4. Only a small fraction of the genes in the AD synergy network are differentially expressed, that is, 11.92% of the 780 genes with positive synergy and 29.58% of the 3083 genes with negative synergy (see Fig. 9.4a). Meanwhile, the majority of the gene pairs with significant information synergy are not differentially correlated (see Fig. 9.4b), that is, 90.6% of the 511 gene pairs with positive and 92.4% of 4308 gene pairs with negative synergy have similar correlations in each phenotype. These observations indicate that information synergy provides complementary information to that obtained from either differential expression or differential correlation analyses.

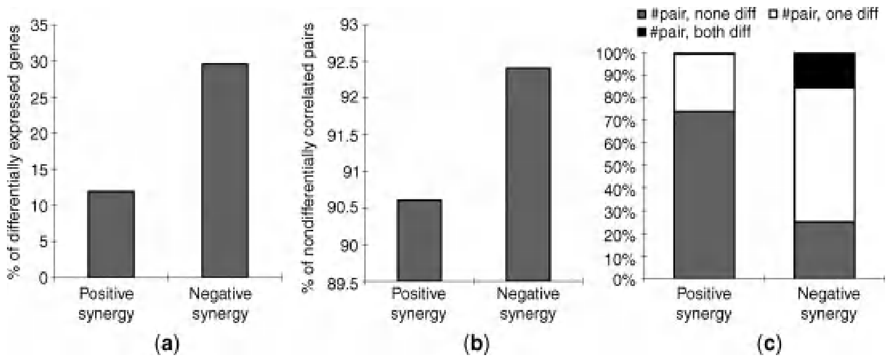


FIGURE 9.4 Percentages of differentially expressed genes and differentially correlated gene pairs in synergy network.

We further investigated the distribution of the gene pairs with zero, one, or two differentially expressed genes (see Fig. 9.4c). Interestingly, 377 of the 511 gene pairs (over 70%) with positive synergy consist of two nondifferentially expressed genes (grey bar in Fig. 9.4c), whereas most of the 377 gene pairs (349 out of 377) are not differentially correlated between AD versus normal (data not shown). This suggested the majority of these gene pairs with positive synergy likely share joint expression pattern shown in Figure 9.1e. Meanwhile, in a majority of the gene pairs with negative synergy, just one gene is differentially expressed and the other gene is not (gray bar in Fig. 9.4c). For both positive and negative synergy gene pairs, only a small fraction of them consist of two differentially expressed genes (purple bar in Fig. 9.4c). These observations again highlight that information synergy can complement differential analysis or differential correlation analysis, particularly by providing genes with positive information synergy. Next, we focus on the pairs with positive synergy, and explore their biological relevance by looking at the hub genes in the network.

9.3.4 Hub Genes in the Positive Synergy Network

We have shown that positive synergy network is of particular interest, since the majority of the gene pairs with positive synergy could not be captured by either differential expression or correlation analysis. To investigate whether a positive synergy network could contribute information on potential targets, we specifically evaluated the top 10 hub genes in the positive synergy network listed in Table 9.2.

Notably, several of these top 10 hub genes have been shown to be involved in AD. The top gene, HDAC1 (histone deacetylase 1), can replace the occupancy of APP intracellular domain (AICD) on the promoter of NEP (neprilysin, a Aβ-degrading enzyme), consequently increasing the expression level of Aβ [45]. Meanwhile, inhibiting HDAC1 was shown to rescue cognitive deficits in a mouse model of AD [46]. It is worthwhile to mention that HDAC1 is not included in the network of differential expression and is ranked much lower, that is, 395th among the 1985 genes in the

TABLE 9.2 The Top 10 Hub Genes in the Synergy Network

Gene	Degree in Synergy Network	Degree in Global PPI Network	Description
HDAC1	11	259	Histone deacetylase 1
ATXN1	6	156	Ataxin 1
CARM1	6	22	Coactivator-associated arginine methyltransferase 1
UBE2N	5	59	Ubiquitin-conjugating enzyme E2N
KAT5	5	88	K(lysine) acetyltransferase 5
RPAP1	5	32	RNA polymerase II associated protein 1
AR	4	181	Androgen receptor
CDKN1B	4	45	Cyclin-dependent kinase inhibitor 1B (p27, Kip1)
ELAVL1	4	17	ELAV (embryonic lethal, abnormal vision, Drosophila)-like 1 (Hu antigen R)
CTTN	4	32	Cortactin

network of differential correlation. Further, the next gene, ATXN1, is involved in AD pathogenesis as a genetic risk modifier that regulates the levels of Aβ and the activity of beta-secretase [47]. In addition, the role of AR in the etiology of AD [48,49] and the degradation of CDKN1B (p27) in AD [50,51] have also been confirmed. With these genes being known to play a role in AD, it questions whether the functions of the other identified hub genes (Table 9.2), with less well-established roles in AD, should be investigated in future studies. For example, ubiquitin–proteasome system (UPS) is essential for the function of protein repair, turnover, and degradation functions in AD [52], thus UBE2N (4th highest hub gene), as a component of UPS, may be a potential target for further investigation.

9.4 SUMMARY AND CONCLUSIONS

In this study, we proposed a network-based information synergy approach to identify candidate genes involved in AD. Results obtained from simulation data and AD microarray data suggested that, information synergy, particularly positive information synergy, could identify gene pairs with specific joint expression patterns that would otherwise be overlooked by differential and correlation analyses. Meanwhile, some of the hub genes in the PPI subnetworks, consisting of positive information synergy interactions, show biological relevance to the pathogenesis of AD, suggesting that the networks obtained through information synergy could potentially identify genes involved in diseases. Moreover, the information theory used in synergy analysis allows one to capture gene pairs with different types of relationship (either linear or nonlinear), as long as the gene pairs provide additional information on the phenotype. This advantage renders information synergy particularly attractive for capturing gene pairs in complex scenarios, where the interactions between genes are not linear. Taken together, information synergy is a promising complementary approach to network-based studies.

The concept of information synergy has been applied to identify gene pairs predictive of phenotypes based upon microarray data. The synergistic gene pairs inferred from microarray data may not necessarily interact with each other physically. This makes it difficult to interpret the biological roles of individual gene pairs, especially when the structural or functional information is not available for many genes. This issue is of less concern when PPI data is incorporated with the gene expression data as physical interactions in the network, which can help facilitate the interpretation of the functions of the genes of interest, that is, by inferring the function of the genes through their neighbors in the network.

We have demonstrated that information synergy provides results complementary to existing approaches in network-based studies. Previous network-based studies have shed much light onto the biochemical features of the genes and proteins that show changes in correlation across phenotypes, that is, smaller binding domain size or enriched in signaling domains, and so on [53]. Exploring such structural characteristics of the genes/proteins with positive information synergy could provide insight on their general mechanisms and further help elucidate the molecular mechanisms underlying diseases. In this chapter, we exemplified the potential role of the synergy network in identifying genes of interest for AD by investigating the hub genes in a positive synergy subnetwork. In future, further exploration of the synergy network, that is, enrichment analysis to identify canonical pathways over-represented in the networks, pathway–gene association analysis to reveal the genes associated with specific pathways, or modular analysis to characterize the network modules with specific patterns in terms of information synergy (i.e., do the connections within a module have exclusively positive or negative synergy), and so on would likely provide more information on potential disease mechanisms.

ACKNOWLEDGMENT

This research was supported in part by the NIH (R01GM079688, R01GM089866, and R21RR024439), the NSF (CBET 0941055, CBET 1049127, and DBI 0701709), and the MSU Foundation.

REFERENCES

1. Y. Huang, Apolipoprotein E and Alzheimer disease, *Neurology* **66**, S79–S85 (2006).
2. Alzheimer's Association website (http://www.alz.org/braintour/healthy_vs_alzheimers.asp).
3. K. Blennow, M.J. de Leon, H. Zetterberg, Alzheimer's disease, *Lancet* **368**, 387–403 (2006).

4. K. Bettens, K. Sleegers, C. Van Broeckhoven, Current status on Alzheimer disease molecular genetics: from past, to present, to future. *Hum. Mol. Genet.* **19**, R4–R11 (2010).
5. M. Citron, Alzheimer's disease: strategies for disease modification. *Nat. Rev. Drug Discov.* **9**, 387–398 (2010).
6. A.L. Barabasi, N. Gulbahce, J. Loscalzo, Network medicine: a network-based approach to human disease. *Nat. Rev. Genet.* **12**, 56–68 (2011).
7. X. Wang, E. Dalkic, M. Wu, C. Chan, Gene module level analysis: identification to networks and dynamics. *Curr. Opin. Biotechnol.* **19**, 482–491 (2008).
8. J. Lim, T. Hao, C. Shaw, A.J. Patel, G. Szabo, et al., A protein–protein interaction network for human inherited ataxias and disorders of Purkinje cell degeneration. *Cell* **125**, 801–814 (2006).
9. M.A. Pujana, J.D. Han, L.M. Starita, K.N. Stevens, M. Tewari, et al., Network modeling links breast cancer susceptibility and centrosome dysfunction. *Nat. Genet.* **39**, 1338–1349 (2007).
10. J. Ladha, S. Donakonda, S. Agrawal, B. Thota, M.R. Srividya, et al., Glioblastoma-specific protein interaction network identifies PP1A and CSK21 as connecting molecules between cell cycle-associated genes. *Cancer Res.* **70**, 6437–6447 (2010).
11. V.M. Perreau, S. Orchard, P.A. Adlard, S.A. Bellingham, R. Cappai, et al., A domain level interaction network of amyloid precursor protein and Abeta of Alzheimer's disease. *Proteomics* **10**, 2377–2395 (2010).
12. M. Soler-Lopez, A. Zanzoni, R. Lluis, U. Stelzl, P. Aloy, Interactome mapping suggests new mechanistic details underlying Alzheimer's disease. *Genome Res.* **21**, 364–376 (2011).
13. Z.P. Liu, Y. Wang, X.S. Zhang, L. Chen, Identifying dysfunctional crosstalk of pathways in various regions of Alzheimer's disease brains. *BMC Syst. Biol.* **4** (Suppl. 2), S11 (2010).
14. M. Ray, W. Zhang, Analysis of Alzheimer's disease severity across brain regions by topological analysis of gene co-expression networks. *BMC Syst. Biol.* **4**, 136 (2010).
15. M. Krauthammer, C.A. Kaufmann, T.C. Gilliam, A. Rzhetsky, Molecular triangulation: bridging linkage and molecular-network information for identifying candidate genes in Alzheimer's disease. *Proc. Natl. Acad. Sci. U.S.A.* **101**, 15148–15153 (2004).
16. J.Y. Chen, C. Shen, A.Y. Sivachenko, Mining Alzheimer disease relevant proteins from integrated protein interactome data. *Pac. Symp. Biocomput.* **11**, 367–378 (2006).
17. B. Liu, T. Jiang, S. Ma, H. Zhao, J. Li, et al., Exploring candidate genes for human brain diseases from a brain-specific gene network. *Biochem. Biophys. Res. Commun.* **349**, 1308–1314 (2006).
18. T. Ideker, R. Sharan, Protein networks in disease. *Genome Res.* **18**, 644–652 (2008).
19. A. Almudevar, L.B. Klebanov, X. Qiu, P. Salzman, A.Y. Yakovlev, Utility of correlation measures in analysis of gene expression. *NeuroRx* **3**, 384–395 (2006).
20. C. Olsen, P.E. Meyer, G. Bontempi, On the impact of entropy estimation on transcriptional regulatory network inference based on mutual information. *EURASIP J. Bioinform. Syst. Biol.* **2009**, 308959 (2009).
21. T.S. Furey, N. Cristianini, N. Duffy, D.W. Bednarski, M. Schummer, et al., Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics* **16**, 906–914 (2000).
22. E.K. Tang, P.N. Suganthan, X. Yao, Gene selection algorithms for microarray data based on least squares support vector machine. *BMC Bioinform.* **7**, 95 (2006).

23. R. Diaz-Uriarte, S. Alvarez de Andres, Gene selection and classification of microarray data using random forest. *BMC Bioinform*. **7**, 3 (2006).

24. T.K. Paul, H. Iba, Gene selection for classification of cancers using probabilistic model building genetic algorithm. *Biosystems* **82**, 208–225 (2005).

25. J. Watkinson, X. Wang, T. Zheng, D. Anastassiou, Identification of gene interactions associated with disease from gene expression data using synergy networks. *BMC Syst. Biol.* **2**, 10 (2008).

26. G. Munch, R. Schinzel, C. Loske, A. Wong, N. Durany, et al., Alzheimer's disease—synergistic effects of glucose deficit, oxidative stress and advanced glycation endproducts. *J. Neural Transm.* **105**, 439–461 (1998).

27. R.X. Santos, S.C. Correia, X. Wang, G. Perry, M.A. Smith, et al., A synergistic dysfunction of mitochondrial fission/fusion dynamics and mitophagy in Alzheimer's disease. *J. Alzheimers Dis.* **20** (Suppl. 2), S401–S412 (2010).

28. R. Lane, H.H. Feldman, J. Meyer, Y. He, S.H. Ferris, et al., Synergistic effect of apolipoprotein E epsilon4 and butyrylcholinesterase K-variant on progression from mild cognitive impairment to Alzheimer's disease. *Pharmacogenet. Genomics* **18**, 289–298 (2008).

29. I. Mateo, P. Sanchez-Juan, E. Rodriguez-Rodriguez, J. Infante, J.L. Vazquez-Higuera, et al., Synergistic effect of heme oxygenase-1 and tau genetic variants on Alzheimer's disease risk. *Dement. Geriatr. Cogn. Disord.* **26**, 339–342 (2008).

30. E. Schneidman, W. Bialek, M.J., 2nd Berry Synergy, redundancy, and independence in population codes. *J. Neurosci.* **23**, 11539–11553 (2003).

31. N. Brenner, S.P. Strong, R. Koberle, W. Bialek, R.R. de Ruyter van Steveninck, Synergy in a neural code. *Neural Comput*. **12**, 1531–1552 (2000).

32. V. Varadan, D.M. Miller, 3rd, D. Anastassiou, Computational inference of the molecular logic for synaptic connectivity in *C. elegans*. *Bioinformatics* **22**, e497–e506 (2006).

33. V. Varadan, D. Anastassiou, Inference of disease-related molecular logic from systems-based microarray analysis. *PLoS Comput. Biol.* **2**, e68 (2006).

34. D. Anastassiou, Computational analysis of the synergy among multiple interacting genes. *Mol. Syst. Biol.* **3**, 83 (2007).

35. T. Barrett, D.B. Troup, S.E. Wilhite, P. Ledoux, C. Evangelista, et al., NCBI GEO: archive for functional genomics data sets—10 years on. *Nucleic Acids Res*. **39**, D1005–1010 (2011).

36. C. Stark, B.J. Breitkreutz, A. Chatr-Aryamontri, L. Boucher, R. Oughtred, et al., The BioGRID Interaction Database: 2011 update. *Nucleic Acids Res*. **39**, D698–D704 (2011).

37. T.S. Keshava Prasad, R. Goel, K. Kandasamy, S. Keerthikumar, S. Kumar, et al., Human Protein Reference Database—2009 update. *Nucleic Acids Res*. **37**, D767–D772 (2009).

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

39. M. Dehmer, ed. *Structural Analysis of Complex Networks*. 1st edn. Birkhäuser Publishing, 2011.

40. A.L. Barabasi, Z.N. Oltvai, Network biology: understanding the cell's functional organization. *Nat. Rev. Genet.* **5**, 101–113 (2004).

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

42. D.S. Lee, J. Park, K.A. Kay, N.A. Christakis, Z.N. Oltvai, et al., The implications of human metabolic network topology for disease comorbidity. *Proc. Natl. Acad. Sci. U.S.A.* **105**, 9880–9885 (2008).
43. T.H. Cormen, C.E. Leiserson, R.L. Rivest, *Introduction to Algorithms*: MIT Press, 2001.
44. A.L. Barabasi, Z.N. Oltvai, Network biology: understanding the cell's functional organization. *Nat. Rev. Genet.* **5**, U101–U115 (2004).
45. N.D. Belyaev, K.A. Kellett, C. Beckett, N.Z. Makova, T.J. Revett, et al., The transcriptionally active amyloid precursor protein (APP) intracellular domain is preferentially produced from the 695 isoform of APP in a beta-secretase-dependent pathway. *J. Biol. Chem.* **285**, 41443–41454 (2010).
46. M. Kilgore, C.A. Miller, D.M. Fass, K.M. Hennig, S.J. Haggarty, et al., Inhibitors of class 1 histone deacetylases reverse contextual memory deficits in a mouse model of Alzheimer's disease. *Neuropsychopharmacology* **35**, 870–880 (2010).
47. C. Zhang, A. Browne, D. Child, J.R. Divito, J.A. Stevenson, et al., Loss of function of ATXN1 increases amyloid beta-protein levels by potentiating beta-secretase processing of beta-amyloid precursor protein. *J. Biol. Chem.* **285**, 8515–8526 (2010).
48. S.J. Fuller, R.S. Tan, R.N. Martins, Androgens in the etiology of Alzheimer's disease in aging men and possible therapeutic interventions. *J. Alzheimers Dis.* **12**, 129–142 (2007).
49. J. Raber, AR, apoE, and cognitive function. *Horm. Behav.* **53**, 706–715 (2008).
50. U. Munoz, F. Bartolome, F. Bermejo, A. Martin-Requero, Enhanced proteasome-dependent degradation of the CDK inhibitor p27(kip1) in immortalized lymphocytes from Alzheimer's dementia patients. *Neurobiol. Aging* **29**, 1474–1484 (2008).
51. U. Munoz, F. Bartolome, N. Esteras, F. Bermejo-Pareja, A. Martin-Requero, On the mechanism of inhibition of p27 degradation by 15-deoxy-delta12,14-prostaglandin J2 in lymphoblasts of Alzheimer's disease patients. *Cell Mol. Life Sci.* **65**, 3507–3519 (2008).
52. B.M. Riederer, G. Leuba, A. Vernay, I.M. Riederer, The role of the ubiquitin proteasome system in Alzheimer's disease. *Exp. Biol. Med. (Maywood)* **236**, 268–276 (2011).
53. I.W. Taylor, R. Linding, D. Warde-Farley, Y. Liu, C. Pesquita, et al., Dynamic modularity in protein interaction networks predicts breast cancer outcome. *Nat. Biotechnol.* **27**, 199–204 (2009).

10

DENSITY-BASED SET ENUMERATION IN STRUCTURED DATA

Elisabeth Georgii and Koji Tsuda

10.1 INTRODUCTION

One of the central topics in analyzing structured data is the detection of dense substructures, often called *clusters*, *modules*, or *communities*. Such sets of strongly interrelated entities provide interesting information in many different contexts, for instance, interaction of proteins, webpage linking, citation of articles, email communication between individuals, spatial proximity of objects in an image, 3D-contact of atoms in a macromolecule, co-occurrence of words in documents, or similarity of genomic sequences. In graph terminology, the entities are referred to as *nodes*, and interactions are represented by edges between the nodes. Different strengths of interactive relationships can be indicated by edge weights.

This chapter presents an enumerative approach to find node sets that satisfy an explicit interaction density criterion [1], conceptually generalizing traditional clique search [2]. Beyond that, the described framework can enumerate dense cluster patterns from asymmetric, bipartite graph structures and from higher-order associations that involve more than two entities at the same time, forming a hypergraph [3,4]. Moreover, the method allows to integrate additional user-defined constraints in order to systematically discover relevant substructures. The density of an entity set is generally defined as the total interaction weight between entities within the set divided by the maximum possible amount of interaction. We describe a reverse search approach to detect all set patterns satisfying a user-defined minimum density threshold. The main idea is to specify a parent–child relationship between set patterns that

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

results in an antimonotonic search space, which allows for effective pruning. Mining approaches that directly control the density properties of patterns are promising in several application fields. In molecular systems biology for instance, one important problem is to infer protein complexes or groups of functionally related genes from experimental measurement data. By considering different density cutoffs, it is possible to trade off precision and recall levels for the discovered patterns. Furthermore, the enumerative method identifies overlapping patterns, and integration with gene activity profiles can reveal context-dependent variation of interaction patterns.

The chapter is organized as follows. The next section sketches related work on pattern detection in structured and relational data. Section 10.3 explains the dense cluster enumeration algorithm for the basic case where input data are represented as an undirected weighted interaction network. Section 10.4 addresses the generalization of the method to higher-order association data and hypergraphs. Section 10.5 discusses the presented approach and points out potential extensions.

10.2 UNSUPERVISED PATTERN DISCOVERY IN STRUCTURED DATA

Computational analysis of structural and relational data is a very broad and active field of research. Here, we briefly review central approaches that are related to relevant substructure detection, including graph mining, optimal subgraph search, graph clustering, biclustering, itemset mining, and higher-order relational data mining.

10.2.1 Graph Mining

Graph mining refers to the search for subgraph patterns with predefined characteristics, in a database of one or multiple graphs. In many cases, it is possible to design algorithms that yield the complete set of solutions; such approaches are called *enumerative*. In the following presentation, we focus on the most common tasks. We start with the classical problem of frequent subgraph mining [5,6]. Given a database of labeled graphs $G_1, \ldots, G_l$, the task is to find all connected subgraphs that occur in at least m graphs, where m is a positive integer referred to as the minimum support threshold. As the same node label may appear multiple times in each graph of the database (e.g., atom names in a database of molecule graphs [7]), these methods generally have to deal with the problem of subgraph isomorphism. In biological networks considering gene or protein relationships, the node labels within a graph are typically unique. However, as the graphs are large, the frequency criterion is typically combined with other criteria such as interaction density or cut thresholds, in order to restrict the size of the output [8,9].

Other mining approaches search for substructure patterns in a single graph. While a subgraph frequency criterion can be applied to graphs with nonunique node labeling [10], a popular analysis tool for uniquely labeled graphs is clique finding [2].

Definition 10.1 (Clique) *Given a graph G with node set V, a clique is defined as a subset of nodes $U \subset V$ that induces a complete subgraph, that is, all pairs of* *nodes are connected by an edge. A clique is maximal if it is not contained in any other clique.*

In general, clique search is NP-complete [11]. Nevertheless, it is frequently used in practical applications [12,13], and even more flexible pattern definitions have been considered, for example, quasi-cliques [14–17] and pseudo-cliques [18]. They correspond to dense subgraphs rather than complete subgraphs; exact definitions will be given in Section 10.3.

10.2.2 Optimal Subgraph Search

In addition to subgraph enumeration algorithms, there exist multiple approaches to search for (approximately) optimal subgraphs. With respect to the criterion of subgraph density, a number of problems have been studied. First of all, it has been shown that finding a k-node subgraph with the maximum number of edges is NP-hard [19]. Tight approximation bounds have been derived for a simple greedy optimization scheme [20]; the same approximation scheme has been used for directed graphs [21]. Equivalently, the problem of finding a k-node subgraph with the maximum average number of edges per node is NP-hard [22]. On the other hand, without a size constraint, a subgraph with the maximum average number of edges per node can be found in polynomial time by flow-based techniques [22,23].

Moreover, local search approaches have been used to discover dense subgraphs around seed cliques [24,25]. Recently, linear integer programming has been successfully applied for finding connected subgraphs with the maximum sum of node weights, an NP-complete problem [26]. Another type of pattern is the so-called connection subgraph [27], where the aim is to maximize the connectivity between a set of given nodes while removing a large portion of the graph.

10.2.3 Graph Clustering

Graph clustering is the task of assigning the nodes of a graph into distinct groups (clusters) such that there are many edges within a group and few edges between different groups. This topic has been studied extensively, see Ref. [28] for a review. One seminal work in this area is the Kernighan–Lin algorithm [29], which is a heuristic strategy to divide a graph into components with fixed maximum size. If neither the (maximum) size nor the number of partitions is known beforehand, a popular choice is hierarchical clustering methods, which yield a hierarchy of clusters instead of a single partitioning. Hierarchical methods can be divided into two classes: agglomerative and divisive. Agglomerative strategies build the hierarchy bottom-up, starting from single-node clusters and iteratively merging the "closest" pair into a common cluster, for example, see Refs. [30–33].

Divisive hierarchical strategies, in contrast, work in a top-down manner, starting with the entire graph and iteratively dividing it into smaller parts. An obvious splitting criterion are graph cuts [34]. Girvan and Newman [35] use the so-called edge betweenness measure, which is the number of node pairs with the shortest path passing through a specific edge; edges bridging between clusters are expected to have large betweenness values, so edges are removed from the graph in decreasing order of betweenness; the same technique has also been applied to weighted graphs [36]; Luo et al. combine this method with an agglomerative approach [37]. Several alternatives of these measures have been investigated [38,39]. Also, divisive clustering has been integrated with graph coarsening to achieve efficient procedures [40].

Other methods directly partition the graph into a set of clusters, without using hierarchical decomposition steps. The most basic approach is to extract the connected components [41]. An increasingly popular method is spectral clustering [42]. Technically, it is based on eigen decomposition of graph Laplacians, and has interpretations related to graph cuts and random walks. Markov clustering [43,44] is also motivated by random walks; it performs two alternating matrix operations; in contrast to spectral clustering, one does not fix the number of clusters beforehand. Furthermore, probabilistic latent variable models have been used for graph clustering [45], which often also allow cluster overlaps, that is, the same node may belong to different clusters [46,47].

In many bioinformatics applications, graph clustering and dense subgraph mining methods are augmented by integrating multiple data sources, the most common scenario being the combination of protein–protein interactions and gene expression data. One straightforward strategy is to build a new network where protein interaction links and coexpression links are simply pooled [47], or where the edge weights are determined as a function of multiple data sources [48]. Tanay et al. [49] combine edges of different types in a single bipartite network. Edge weights of different datasets have to be normalized appropriately in order to be comparable in the integrated setting. In contrast to that, other approaches keep the data sources separate and define individual constraints for each of them. A variety of criteria have been proposed, including global and local similarity measures for gene expression profiles [14,45,50–54].

10.2.4 Bicluster Analysis

In addition to homogeneous interaction graphs such as protein interaction networks, bipartite graphs occur very frequently in biological data analysis. Similarly as in the previous sections, one central problem arising in that context is dense subgraph detection. A pattern of interest would then consist of a pair of node subsets, one from each partition, such that each node is connected to a large fraction of nodes from the other set. This can be seen as a special instance of the biclustering problem, which is very prominent in gene expression analysis [55–58]. Given a data matrix (e.g., the adjacency weights of a bipartite graph), the goal is to extract subsets of rows that are similar with respect to subsets of columns; a particular pair of a row subset and a column subset (defining a submatrix) is called bicluster. This framework, which is also known as co-clustering or two-mode clustering [59], contrasts with traditional clustering approaches, which cluster either the rows according to their similarity across all columns, or the columns according to their similarity across all rows [60].

A multitude of bicluster detection methods has been developed during the last decade. First of all, one basic idea is to partition the bipartite graph that corresponds to the data matrix into distinct biclusters [61,62]. Other approaches allow for a more flexible arrangement of biclusters, including bicluster overlap, while still optimizing a global objective function taking the whole matrix into account [63,64]. In contrast, enumerative approaches use local criteria based solely on individual biclusters. Many of them are motivated by the (weighted) bipartite graph formalism and refer to some density property of the subgraph. The widely used SAMBA method [49,65] finds heavy subgraphs around each node. Sim et al. [66] fix the maximum number of missing edges tolerated per node as well as minimum size constraints. In Ref. [67], all maximal bicliques are detected and then further extended. Another work [68] searches for all bicluster patterns that satisfy homogeneity constraints with respect to the weight entries. Moreover, various other methods define specific bicluster criteria and solve the problem in a nonexhaustive way, by greedy strategies or approximation techniques [55].

10.2.5 Itemset Mining

For binary-valued data matrices, the simplest bicluster pattern of interest is a submatrix that purely consists of 1-entries. Such patterns can be exhaustively enumerated using itemset mining. This approach has been developed in the context of market basket analysis [69]. There, the data represent a set of transactions, where each transaction consists of a list of products (called items) that were purchased together. The task of frequent itemset mining is to find all sets of items that co-occur in more than m transactions. The frequent itemsets can be used to derive association rules of the following kind: "if a customer bought products A and B, he or she will also buy product C." This information can, for instance, assist in improving shop layouts. Itemset mining has also been applied in the biological domain, for example, gene expression analysis [70]. The principal algorithmic idea behind itemset mining methods is based on the observation that any subset of a frequent itemset is frequent as well. The originally proposed Apriori algorithm [69] implements this in a level-wise search strategy, where the frequent sets on one level determine the candidate sets on the next level. Since then, there has been active research on improving the efficiency by introducing additional pruning rules and investigating alternative strategies to traverse the search space [71].

10.2.6 Relational Data Mining and Higher-Order Association Analysis

Itemset mining is only suitable for analyzing binary relations such as the transaction-item association data described in the previous section. A natural extension is to consider higher-order relations, which involve more than two partners. For example, one could consider relations between purchased items, regions, and weeks of customer transactions. The generalized mining task can be formulated as follows [72–75]:

Definition 10.2 (Relational Set Mining) *Given an n-ary relation $R \subset D_1 \times \ldots \times D_n$, find all n-set patterns $(S_1, \ldots, S_n)$ such that $S_i \subset D_i$ for all $i = 1, \ldots, n$ and $S_1 \times \ldots \times S_n \subset R$.*

Generalizing the frequency criterion from itemset mining (see previous section), one can specify minimum size thresholds for $S_1, \ldots, S_n$. In the above definition, all domains D_i are assumed to be finite (i.e., categorical); their cardinality is denoted by $|D_i|$. An equivalent representation for such data is a $|D_1| \times \ldots \times |D_n|$ array A (also called data cube or tensor), where $A(d_1, \ldots, d_n) = 1$ if $(d_1, \ldots, d_n) \in R$, and $A(d_1, \ldots, d_n) = 0$ otherwise ($d_i \in \{1, \ldots, |D_i|\}$, $i = 1, \ldots, n$). Then, an n-set corresponds to a subarray that contains only 1-entries.

More generally, one can drop the constraint of binary values and consider tensors with arbitrary weight entries. Such higher-order datasets occur in different application fields such as sales analysis [72], web mining [74,76,77], neuroscience [78,79], and computational biology [75,80,81]. Therefore, methods that deal with multiway arrays receive increasing attention in the data mining community. One of the most prominent topics is tensor decomposition (see Ref. 82 for a review), which can serve as a basis for clustering or anomaly detection [83].

The goal of tensor clustering is to partition each dimension of the tensor into a predefined number of clusters such that the resulting multiway clusters are as homogeneous as possible [84]. This can be approximated by combining the results from clustering individual dimensions separately [85]. Zhao and Zaki [80] also mine for homogeneous clusters, but instead of specifying the number of clusters, they fix thresholds regarding the homogeneity of values along each dimension and detect overlapping cluster patterns (in the three-way case). Relational models [86] focus on binary-valued tensors, aiming at partitioning them into blocks that contain either mostly ones or mostly zeros. Finally, there exist approaches that deal with multiple relations or tensors at the same time, searching for clusters (communities) [84,87] or association rules [88].

10.3 DENSE CLUSTER ENUMERATION IN WEIGHTED INTERACTION NETWORKS

This section presents an enumerative approach to identify cluster patterns in undirected weighted graphs and networks [1]. In this context, a cluster is defined as a set of densely interacting nodes, also called *module* or *community*. The algorithm makes use of a paradigm called *reverse search*, which has been introduced by Avis and Fukuda [89]. Several extensions are described, including output filtering and integration of additional constraints. We first introduce some notation and give a precise mathematical formulation of the dense cluster mining problem; then, we explain the algorithm and analyze its computational complexity; finally, we outline extensions that facilitate systematic data analysis and interpretation.

10.3.1 Notation and Problem Definition

Let us consider an undirected weighted graph with node set V. For notational convenience, we assume that V is a set of consecutive indices starting from 1, that is,

$$V = \{1, \ldots, I\}, \tag{10.1}$$

where I is a positive natural number. Further, we denote by $|V|$ the size or *cardinality* of the node set, that is, the number of nodes (here, $|V| = I$). The corresponding *interaction weight matrix* is written as

$$W = (w_{ij})_{i,j \in V}\ ,\ w_{ij} = w_{ji}. \tag{10.2}$$

It contains for each pair of nodes an entry, which corresponds to the weight of the connecting edge, if existent, and has a default value of zero otherwise.[1] In the following, we assume that the weights are given relative to their maximum possible value, so the normalized weights are bounded by 1:

$$w_{ij} \leq 1 \tag{10.3}$$

Negative weights are possible.[2] Unweighted input graphs are translated into binary weight matrices with 1-entries for existing edges and 0-entries for missing edges.

A cluster is defined as a nonempty subset of nodes $U \subset V$, $|U| \geq 1$. The induced subgraph corresponding to a specific cluster U is represented by the following interaction matrix:

$$W|_U = (w_{ij})_{i,j \in U}\ . \tag{10.4}$$

The average pairwise interaction weight within a cluster is referred to as the *cluster density*.

Definition 10.3 (Cluster Density) *For a node set V with interaction weight matrix W and a cluster $U \subset V$, the density of U with respect to W is defined as*

$$\rho_W(U) = \frac{\sum\limits_{i,j \in U, i<j} w_{ij}}{|U|(|U|-1)/2}\ . \tag{10.5}$$

Here, self-interactions of nodes are not taken into account.[3] Because of the weight normalization, the largest possible density value is 1, conveniently expressed as 100%. For $|U| \leq 1$, we define $\rho_W(U)$ =100%.

Now we formulate the cluster mining problem of interest.

Definition 10.4 (Dense Cluster Enumeration) *Given a graph with node set V and interaction weight matrix W, and a minimum density threshold $\theta > 0$, find all clusters $U \subset V$ such that $\rho_W(U) \geq \theta$.*

[1]Other default values may be specified as well.

[2]However, non-negative input matrices allow for additional improvements of the search, see Sections 10.3.3 and 10.3.5.

[3]However, they can be integrated in a similar way as node weights (see Section 10.3.8).

The next section introduces an exact method to solve this problem. For unweighted input graphs, the problem is equivalent to pseudo-clique enumeration [18] (for $\theta < 100\%$) or clique search [11] (for $\theta = 100\%$).

10.3.2 Enumeration Algorithm

In the following, we describe an algorithm to solve the dense cluster enumeration problem. Being based on a reverse search strategy [89], it generalizes a pseudo-clique mining method for unweighted graphs [18].

10.3.2.1 Search Space

The core of any enumeration algorithm is the definition of a search space structure that allows for efficient traversal and pruning. A canonical search scheme for set enumeration tasks is to start with the empty set and then iteratively form larger sets by adding one element at a time. This defines a search space that is organized in multiple levels, having the empty set as its root; each time one moves a level downwards, the set obtains an additional member, that is, the set cardinality increases by 1. Figure 10.1c illustrates the search space of node sets for the example input graph with four nodes shown in Figure 10.1a. For an efficient search, it is crucial to avoid recomputations, that is, the same set should not be visited several times. This is usually achieved by defining a tree structure that spans the original graph-shaped search space, that is, each set descends from a uniquely defined parent and can have several sets as children. In pattern mining approaches, it is very common to use lexicographical set enumeration trees [90–92]. There, a predefined order on the elements is exploited such that a set can only be extended by elements that are greater than all current member elements; the extended sets form the children of the orginal set (see Figure 10.1d for an example).

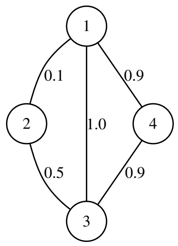
As the size of the complete set enumeration tree is exponential in the number of input elements, the practical applicability of a search procedure strongly depends on the definition of effective pruning rules, which prevent the exploration of irrelevant subtrees. To motivate our search algorithm for the dense cluster enumeration task, let us first consider the special problem of clique finding. In that case, the pruning is straightforward; if the current cluster is not a clique, we know that all supersets are noncliques as well. More generally, this property is described as *downward closure* or *antimonotonicity* [69].

Definition 10.5 (Antimonotonicity) *A function $f : 2^V \to \mathbb{R}$ is antimonotonic if $f(U') \geq f(U)$ for all U' and U with $U' \subset U \subset V$.*

Here, 2^V denotes the power set of V, so the function f assigns a score to any subset of nodes in the input graph. For clique search, we define $f(U) = 1$ if U is a clique, and $f(U) = 0$ otherwise. If the current cluster has a score of 0, all descendants will have that score; hence, we can prune the search tree as soon as the clique criterion is violated.

While the lexicographical search tree can be used for clique search, it is not suitable for solving the general dense cluster enumeration problem because the density

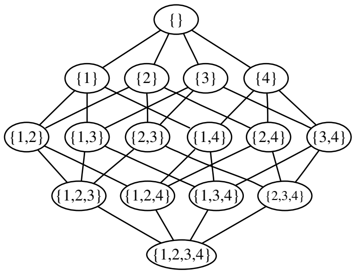
(a)



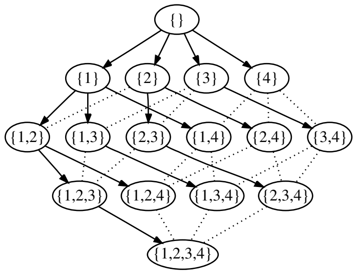
(b)

	1	2	3	4
1	0	0.1	1.0	0.9
2	0.1	0	0.5	0
3	1.0	0.5	0	0.9
4	0.9	0	0.9	0

(c)



(d)



(e)

Cluster	Density
{1,2}	0.10
{1,2,3}	0.53
{1,2,4}	0.33
{1,3}	1.00
{1,3,4}	0.93

(f)

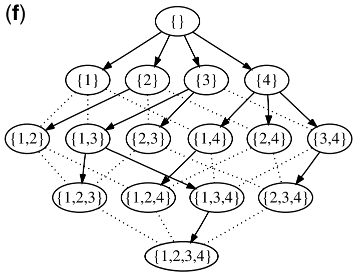


FIGURE 10.1 Motivating example for cluster enumeration strategy. While a lexicographical traversal of the search space does not yield density guarantees, the cluster density is monotonically decreasing along each path of the reverse search tree. (a) Example input graph. (b) Corresponding weight matrix. (c) Graph-shaped search space. (d) Lexicographical tree. (e) Densities of example clusters. (f) Reverse search tree.

criterion is in general not antimonotonic. For instance, while the density decreases when we go from {1, 3} to {1, 3, 4}, it increases when stepping from {1, 2} to {1, 2, 3} (see Fig. 10.1e). Thus, the lexicographical structure does not provide guarantees regarding the maximum cluster density in subtrees, which makes it impossible to define effective pruning rules. The key idea for the dense cluster enumeration approach consists in the definition of a specific search tree where the density is monotonically decreasing on each path from the root to a leaf. We call this structure an *antimonotonic set enumeration tree*.

Definition 10.6 (Antimonotonic Set Enumeration Tree) *A set enumeration tree is antimonotonic with respect to a function $f : 2^V \to \mathbb{R}$ if $f(U') \geq f(U)$ for all U' and U such that $U' \subset U$ is the parent of U.*

Note that in this definition, the antimonotonicity depends on a specific parent–child relationship between sets, whereas Definition 10.5 considers general subset–superset relations. Figure 10.1f shows a search tree that is antimonotonic with respect to the cluster density (for the weighted example graph given in Fig. 10.1a). Regarding the clique criterion, both this search tree and the lexicographical tree (Fig. 10.1d) are antimonotonic because the antimonotonicity holds for any subset–superset relation.

10.3.2.2 Reduction Scheme

Next, we explain how to construct a cluster search tree that enforces antimonotonicity of the cluster density. For that purpose, we need the definition of *degree* in weighted graphs.

Definition 10.7 (Degree) *Given a node $u \in U \subset V$, the (weighted) degree of u with respect to the cluster U is defined as*

$$\deg_U(u) = \sum_{j \in U, j \neq u} w_{uj}\,. \tag{10.6}$$

The degree obviously depends on the given weight matrix W. As W remains fixed during the whole algorithm, we omit an explicit reference to W in the notation.

The following lemma states a fundamental property of the cluster density.

Lemma 10.1 *Let $v \in U$ be a node with minimum degree in U, that is, for all $u \in U : \deg_U(u) \geq \deg_U(v)$. Then, $\rho_W(U \setminus \{v\}) \geq \rho_W(U)$.*

Proof: Using the formulae for cluster density and degree from Definitions 10.3 and 10.7, respectively, we can rewrite the following expression:

$$\begin{aligned}
&\rho_W(U \setminus \{v\}) - \rho_W(U) \\
&= \frac{\left(\sum_{i,j \in U, i<j} w_{ij}\right) - \left(\sum_{j \in U} w_{vj}\right)}{(|U|-1)(|U|-2)/2} - \frac{\sum_{i,j \in U, i<j} w_{ij}}{|U|(|U|-1)/2} \\
&= \frac{\left(\sum_{i,j \in U, i<j} w_{ij}\right)\left(1 - \frac{|U|-2}{|U|}\right) - \deg_U(v)}{(|U|-1)(|U|-2)/2} \\
&= \frac{\frac{1}{|U|}\sum_{u \in U} \deg_U(u) - \deg_U(v)}{(|U|-1)(|U|-2)/2} \\
&\geq 0
\end{aligned}$$

The inequality holds because of the assumption that v is a minimum degree node in U. In summary, it follows that $\rho(U \setminus \{v\}) \geq \rho(U)$. ■

This "weak antimonotonicity" property has been indirectly exploited as an auxiliary pruning criterion in pattern mining for unweighted graphs [93]. Furthermore, iterative removal of minimum degree instances has been used to approximate dense subgraphs [20].

In the context of dense cluster enumeration, the lemma yields the key for defining an antimonotonic search tree. Namely, we define the *parent* of a certain cluster as *the cluster that is obtained by removing a minimum degree node*. If there exist several minimum degree nodes, we apply an arbitrary predefined rule to select one among them, in order to ensure the *uniqueness of the parent*. For that purpose, we exploit a specific order on the nodes; as the nodes are represented by a set of indices (see Eq. 10.1), we here simply use the order of natural numbers. With this, the parent–child relationship between clusters is formally described as follows.

Definition 10.8 (Cluster Parent) *Given a cluster U, let $v \in U$ be the unique node having the smallest index among all minimum degree nodes, that is,*

$$\forall u \in U \setminus \{v\} : \quad \left[\deg_U(v) < \deg_U(u)\right] \vee \left[\deg_U(v) = \deg_U(u) \wedge v < u\right] .$$

Then, $U \setminus \{v\}$ is the parent of U.

As the parent is a subset of the original cluster, this parent construction rule is also called *reduction scheme*. It defines a tree structure on the search space of clusters, which has two important properties: antimonotonicity and *completeness (cluster reachability)*. Lemma 10.1 implies that each parent cluster has at least the same density as its children, so the tree is guaranteed to be antimonotonic. The second property refers to the fact that the tree covers the whole search space: by iterative application of the reduction scheme, any cluster is transformed into the empty set, hence, it is reachable on a path from the root.

10.3.2.3 Search Procedure

Given a search tree with these properties, the enumeration procedure is straightforward. We traverse the tree, for example, by depth-first search, starting from the empty set and recursively generating children on demand as long as the density threshold is satisfied. However, there remains one complicating fact; while the reduction scheme allows to go from a cluster to its parent (i.e., bottom-up), it is not possible to directly derive the children of a given cluster in a top-down search. Therefore, we have to generate all direct supersets of the current cluster U and check whether they have U as a parent. This paradigm is known as *reverse search principle* [89]. For clarity, we reformulate the conditions of the parent–child relationship (Definition 10.8) from a top-down perspective, that is, as it is used during the search process.

TABLE 10.1 Dense Cluster Enumeration Algorithm (DCE) for a Network with Node Set V, Interaction Weight Matrix W, and Minimum Density Threshold θ

```
1: DCE $(V, W, \theta, U)$ :
2:   for each $v \in V \setminus U$ do
3:     if $\rho_W(U \cup \{v\}) \geq \theta$ and $U \cup \{v\}$ is child of $U$ then
4:       DCE $(V, W, \theta, U \cup \{v\})$
5:     end if
6:   end for
7:   output $U$
```

U represents the current cluster; in the initial method call, U is the empty set.

Definition 10.9 (Cluster Children) *Let U be a cluster and $v \in V \setminus U$. The extended cluster $U^* := U \cup \{v\}$ is a child of U if and only if*

$$\forall u \in U : \quad \left[\deg_{U^*}(v) < \deg_{U^*}(u)\right] \;\vee\; \left[\deg_{U^*}(v) = \deg_{U^*}(u) \;\wedge\; v < u\right] .$$

With this, the dense cluster enumeration method boils down to the simple pseudocode shown in Table 10.1. In the beginning, the cluster U is set to the empty set. Then, the algorithm builds extended cluster candidates with every node that is not yet contained in U. If a candidate satisfies the density criterion and is indeed a child of U, the search is continued recursively.[4] Note that although the same set can be considered several times as a candidate, it occurs only once as a valid child, namely as an extension of its true parent. Hence, it will be processed only once; there is no replication of subsearches (i.e., a specific set is used at most once during the whole search as input of a DCE call). For an illustration of this fact, see Figure 10.1f; dotted lines indicate unsuccessful candidates, whereas solid edges indicate true parent–child relationships.

EXAMPLE 10.1 We go step by step through an example run of DCE for the graph given in Figure 10.1a, with the density threshold θ set to 0.9. First, we extend the empty set by element 1; $\{1\}$ is trivially dense and a child of the empty set, so we try to further extend it. Candidate $\{1,2\}$ is not sufficiently dense; the subsequent candidates $\{1,3\}$ and $\{1,4\}$ satisfy the density threshold, but are not children of $\{1\}$. Next candidate and true child of the empty set is $\{2\}$. Candidates $\{1,2\}$, $\{2,3\}$, and $\{2,4\}$ fail the density criterion, so we go to set $\{3\}$. $\{1,3\}$ is dense and a true child, so we next consider $\{1,2,3\}$, but it does not satisfy the density threshold. $\{1,3,4\}$ does and is a true child, but $\{1,2,3,4\}$ is not a dense cluster, and we continue with the next candidate of $\{3\}$, which is $\{2,3\}$, and so forth.

[4]The density check comes first because it is computationally cheaper, see next section.

The correctness of the algorithm follows directly from the antimonotonicity and completeness properties discussed above; that is, it finds all clusters that satisfy the given density threshold. Apart from the presented depth-first search approach, other tree traversal strategies are applicable as well, such as breadth-first search, top-k search, or iterative deepening variants [94].

10.3.3 Efficient Generation of Children

The central step of a reverse search algorithm is the generation of children for the current solution. Therefore, engineering of this process is important for the efficiency of the method. Here, we describe some additional details for the implementation of the dense cluster enumeration approach.

First of all, the density of a cluster candidate and the degree values of its nodes can be calculated incrementally. For that purpose, we maintain an array d of length $|V|$ where we store the degree of each node with respect to the current cluster U; more precisely, it contains for the nodes in U the degree value with respect to U, and for all nodes $v \in V \setminus U$ the degree value looking one potential extension step ahead, that is,

$$d_U(v) = \begin{cases} \deg_U(v) & \text{if } v \in U \ , \\ \deg_{U \cup \{v\}}(v) & \text{if } v \in V \setminus U \ . \end{cases} \tag{10.7}$$

In addition, we keep track of the *total weight* of the current cluster, which is equivalent to the following expressions:

$$\text{totalWeight}(U) = \sum_{i,j \in U, i<j} w_{ij} = \tfrac{1}{2} \sum_{u \in U} \deg_U(u) \tag{10.8}$$

This allows to check in constant time whether the candidate $U \cup \{v\}$ satisfies the density criterion:

$$\rho_W(U \cup \{v\}) \geq \theta \iff \text{totalWeight}(U) + d_U(v) \geq \theta \, |U|(|U|+1)/2 \tag{10.9}$$

For each successful candidate $U \cup \{v\}$, we further investigate whether it actually is a child of the cluster U. For this, we have to test the conditions given in Definition 10.9, which requires $O(|U|)$ operations (assuming constant access to the entries in W); we have to determine the values of $\deg_{U \cup \{v\}}(u) = d_U(u) + w_{uv}$ for all $u \in U$ and compare them with $\deg_{U \cup \{v\}}(v) = d_U(v)$. In several cases, however, it is possible to skip these computations and decide in constant time whether $U \cup \{v\}$ is a child or not. The following two lemmata describe such speed-up rules.

Lemma 10.2 *Given a cluster U with respect to the interaction weight matrix W, let $u^* \in U$ be the previously added node. Let us consider a node $v \in V \setminus U$. If W is non-negative, the following rule holds:*

$$[d_U(v) < d_U(u^*)] \vee [d_U(v) = d_U(u^*) \wedge v < u^*] \implies U \cup \{v\} \text{ is a child of } U.$$

Proof: By definition, u^* is the smallest among the minimum degree nodes in U. As W contains only non-negative entries, we obtain for $u \in U$ $\deg_{U\cup\{v\}}(u) = d_U(u) + w_{uv} \geq d_U(u) \geq d_U(u^*)$. Further, $\deg_{U\cup\{v\}}(v)$ is equal to $d_U(v)$ by definition. In the case of $d_U(v) < d_U(u^*)$, it follows that $\deg_{U\cup\{v\}}(v) = d_U(v) < d_U(u^*) \leq \deg_{U\cup\{v\}}(u)$ for $u \in U$, so v is the unique node with minimum degree in $U \cup \{v\}$. In the second case, an analogous derivation shows that v is the node with smallest index in the set of minimum degree nodes in $U \cup \{v\}$. ■

Lemma 10.3 *Given a (normalized) weight matrix W and a cluster U, let $u^* \in U$ be the previously added node. For $v \in V \setminus U$, the following rule holds:*

$$[d_U(v) > d_U(u^*) + 1] \vee [d_U(v) = d_U(u^*) + 1 \wedge v > u^*]$$
$$\implies U \cup \{v\} \textit{ is not a child of } U.$$

Proof: By assumption, $w_{ij} \leq 1$ for all $i, j \in V$, so $\deg_{U\cup\{v\}}(u^*) = d_U(u^*) + w_{u^*v} \leq d_U(u^*) + 1$. If $d_U(v) > d_U(u^*) + 1$, it follows directly that $\deg_{U\cup\{v\}}(v) = d_U(v) > \deg_{U\cup\{v\}}(u^*)$, so v cannot be a minimum degree node. A similar argument holds for the case $d_U(v) = d_U(u^*) + 1 \wedge v > u^*$. ■

Further efficiency improvements can be achieved by keeping candidate nodes sorted according to their degree values, see Ref. [18].

10.3.4 Complexity

In combinatorial enumeration problems, the output size (i.e., the number of solution patterns) can be exponential in the input size. As an extreme example, a fully connected input graph with node set V contains $2^{|V|}$ cliques; in other words, each subset of nodes appears in the output. Therefore, rather than the total running time, a conventional complexity measure for enumeration methods is the time between two consecutive solution patterns, which is called *delay* [95–97]. In the following, we show that the reverse search algorithm for dense cluster enumeration has polynomial delay.

Using the degree array implementation described in the previous section, each recursion step of Algorithm 10.1 needs $O(|V| + |V \setminus U| \cdot |U|)$ operations, for updating the degree array and generating the children of the current cluster U. In the worst case, we perform for each candidate node a temporary update for the degree values of the nodes in U and compare them to its own degree value. However, in practice, the number of operations is typically much smaller because many candidates already fail at the density check, and the rules from Lemmas 10.2 and 10.3 often allow to circumvent the temporary update step. To estimate the delay to the subsequent solution pattern, we consider a small modification of the algorithm. If the recursion depth is odd, we output the cluster before the recursive calls, and otherwise afterwards. Thereby, any three consecutive iterations of the code yield at least one

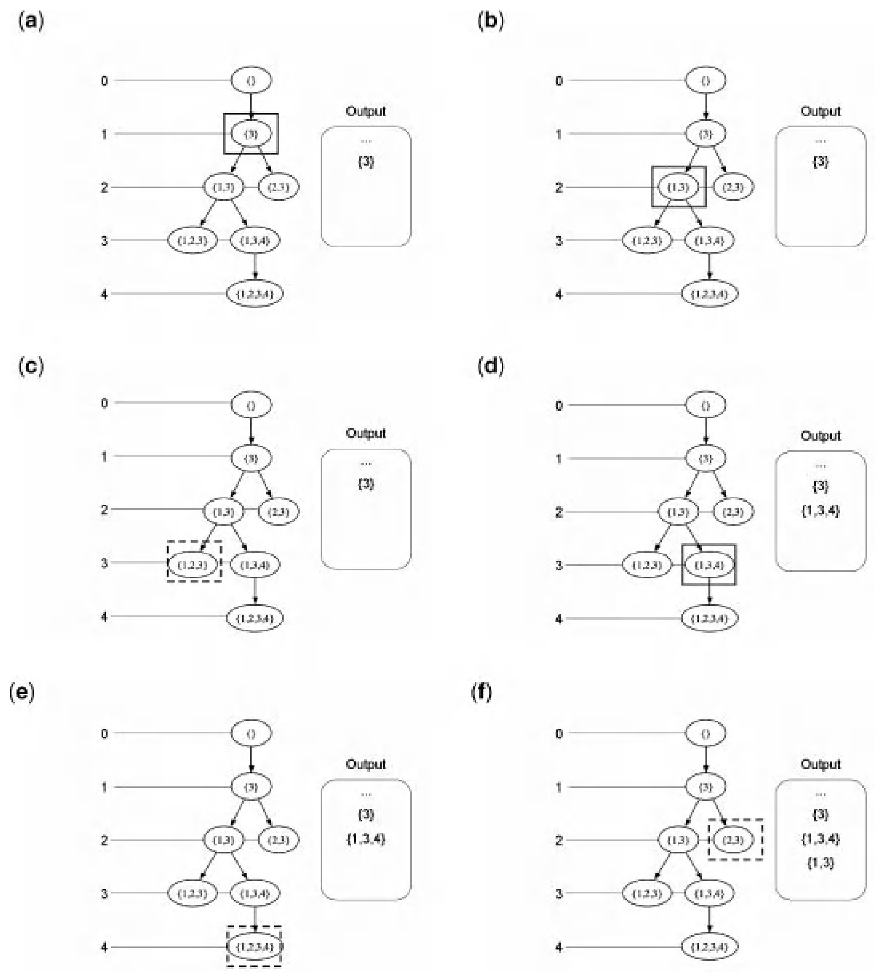


FIGURE 10.2 Illustration of reverse search with the odd–even output method. Sequence of traversal steps for a subtree of the example in Figure 10.1. The boxes indicate the cluster that is currently investigated. The solid boxes correspond to solution clusters; the dashed boxes correspond to candidates that are pruned. For that, we assume a minimum density threshold of 0.9. Note that each solution cluster gives rise to a new recursive call. The numbered lines indicate the levels of recursion depth. After three recursive calls, two of which are completely executed ({1, 3} and {1, 3, 4}), the output contains three solution patterns.

output.[5] This computational trick is known as the odd–even output method [18,98]. For illustration, we show in Figure 10.2 an example execution of the algorithm. Now, the delay has the same complexity as the execution of one recursion step, that is,

[5]Note that without this modification, the algorithm would output solutions only after having reached leaf clusters of the search tree (i.e., after a sequence of recursive calls).

$O(|V| + |V \setminus U| \cdot |U|)$, where U is the current solution. More generally, let the $U_{\max}$ be the largest solution pattern; then, the delay is bounded by $O(|V| \cdot |U_{\max}|)$, so it is at most quadratic in the number of nodes in the network (in practice $|U_{\max}| \ll |V|$). We summarize our (worst-case) analysis in the following theorem.

Theorem 10.1 *For a given input graph with node set V, the dense cluster enumeration problem can be solved by a reverse search algorithm that has a delay of $O(|V|^2)$.*

In contrast to that, straightforward branch-and-bound strategies might need exponential time between two outputs because they have to solve an NP-complete problem in each recursive step [18]. Moreover, the reverse search approach is directly compatible with distributed computation because different branches of the search tree can be investigated in parallel. Finally, the memory requirements of the recursive implementation are also polynomial in the input size. For each recursive call, we store the current cluster U and the degree array of length $|V|$. Hence, the space complexity depends on the maximum recursion depth, $|U_{\max}|$, and is given by $O(|U_{\max}| \cdot |V|)$ plus the space needed for the input matrix. Using a simple implementation with a full matrix representation, the total space complexity of the algorithm amounts to $O(|V|^2)$, but improvements for sparse settings are conceivable.

10.3.5 Output Representation

As enumerative approaches potentially return a large solution set, we discuss in this section how to obtain a user-friendly representation of the dense cluster enumeration output. In particular, subcluster solutions can be efficiently eliminated, and the remaining clusters are ranked according to their statistical significance.

10.3.5.1 Locally Maximal Clusters

Usually, the user is not interested in clusters that are subsets of other cluster solutions; rather, the most comprehensive clusters are most relevant for further analyses. Therefore, the concept of *maximality* is widely used in pattern mining approaches [2,92,99–101]. In the context of dense cluster mining, it can be formulated as follows.

Definition 10.10 (Maximal Dense Cluster) *A dense cluster is called maximal if it is not contained in any other dense cluster.*

A straightforward approach to obtain the set of maximal solutions would be to go for each newly detected cluster through all previous solutions, checking for inclusions. However, the structure of our reverse search algorithm allows us to reduce the number of solutions in the output in a meaningful way without any additional costs. We simply set a flag that indicates whether there exists a direct supercluster (i.e., a cluster with one additional node) that also satisfies the minimum density threshold (see algorithm in Table 10.2). If that is the case, we do not output the current cluster, otherwise we do. This yields us the set of all *locally maximal* dense clusters.

TABLE 10.2 Enumeration of Locally Maximal Dense Clusters for a Network with Node set V, Interaction Weight Matrix W, and Minimum Density Threshold θ

```
1: **DCE_lmax** $(V, W, \theta, U)$ :
2:     locallyMaximal = true
3:     **for each** $v \in V \setminus U$ **do**
4:         **if** $\rho_W(U \cup \{v\}) \geq \theta$ **then**
5:             locallyMaximal = false
6:             **if** $U \cup \{v\}$ is child of $U$ **then**
7:                 **DCE_lmax** $(V, W, \theta, U \cup \{v\})$
8:             **end if**
9:         **end if**
10:    **end for**
11:    **if** locallyMaximal **then**
12:        **output** $U$
13:    **end if**
```

U represents the current module; in the initial method call, U is the empty set.

Definition 10.11 (Locally Maximal Dense Cluster) *A dense cluster U is called locally maximal if for all $v \in V \setminus U$, $U \cup \{v\}$ does not satisfy the minimum density threshold.*

10.3.5.2 Cluster Ranking

Even after filtering the results for local maximality or other predefined criteria, the solution set might be large. Therefore, it is important to provide a meaningful ranking criterion for the discovered clusters. A widely used concept to measure the uncommonness or statistical significance of patterns are p-values. In general, the p-value of a certain pattern is defined as the probability that a randomly selected pattern of equal size is at least as "good" as the given pattern [102]. In our case, the statistics to measure the quality of an outcome U is its density. Regarding the model for random selection, the simplest choice is to draw $|U|$ nodes without replacement from the network at hand. The probability that this produces a cluster with at least the same density as the given pattern U is calculated by the following expression:

$$p_W(U) = \left|\{U' \subset V : |U'| = |U| \wedge \rho_W(U') \geq \rho_W(U)\}\right| \Big/ \binom{|V|}{|U|} \qquad (10.10)$$

The completeness of our dense cluster enumeration approach enables us to determine the numerator exactly, that is, we can compute for each detected cluster an *exact p-value* [102]. For that purpose, we have to keep track of the densities and the sizes of all solutions we encounter during the search. If we maintain for each cluster size a sorted list of densities, one pass is sufficient to obtain the p-values for all output clusters of that size. Lower p-values indicate more remarkable or surprising patterns, so the results are sorted in increasing order of their p-values. This ranking scheme captures the intuition that the importance of a cluster increases with its size and density, which is used in other ranking measures as well. For instance, the product of cluster size and density has been proposed as a possible criterion [25]. Furthermore, this p-value specifically refers to the network at hand, in contrast to significance measures that are based on reference network models [103]. This property can be advantageous if real networks violate the assumptions of network models.

Other cluster finding approaches calculate p-values by assessing the number of interactions within the cluster relative to the number of interactions between cluster nodes and the remaining network [104]; in that case, interaction weights are ignored. Finally, it is very common to estimate *empirical p-values* by generating multiple random networks with the same degree distribution as the given input network [105]. The p-value criterion formulated above does not take the node degrees into account. However, more sophisticated approaches respecting degree distributions are conceivable.

10.3.6 Degree-Based Cluster Criteria

So far, our primary criterion of interest has been the density of interactions across the whole cluster. Here, we discuss more specific criteria that refer to individual cluster nodes.

10.3.6.1 Minimum Degree

The cluster density criterion is very flexible, allowing for missing or weak edges to a certain extent; in particular, for a fix density threshold, the flexibility increases with growing cluster size. While this property is advantageous in many situations (e.g., for finding clusters supported by a large number of weak edges as well as clusters containing few, but very strong edges), there can also occur undesired artifacts, a large dense cluster might tolerate the addition of several loosely connected nodes without violating the density threshold. This effect can blow up the number of solutions considerably, even after selection for (local) maximality, because the same core cluster can appear in many different variants, each time augmented by a few loosely attached, possibly irrelevant nodes. Of course, an obvious remedy would be to choose a stricter density threshold, but this could lead to the loss of other interesting solutions. Therefore, we introduce an optional filtering criterion for clusters, which fixes a minimum degree threshold for cluster nodes.

Definition 10.12 (Minimum Degree Threshold) *A cluster U satisfies the minimum degree threshold t if* $\deg_U(u) > t$ *for all* $u \in U$.

By default, we set $t = 0$, that is, clusters containing isolated nodes or nodes with negative degree are not considered as solutions. If the interaction matrix W contains only non-negative entries, it is straightforward to search for clusters that are locally maximal with respect to the new combined criterion, consisting of a minimum density threshold and a minimum degree threshold; here, a solution cluster U is locally maximal if and only if $\forall v \in V \setminus U : \rho_W(U \cup \{v\}) < \theta \vee \min_{u \in U \cup \{v\}} \deg_{U \cup \{v\}}(u) \leq 0$. For mixed-sign input data, the situation is more complicated because the degree of a node does not monotonically increase with the extension of the cluster. Therefore, one either needs additional costs to check the local maximality, or resort to considering maximality with respect to own descendants only.

10.3.7 Minimum Relative Degree and Quasi-Cliques

A drawback of the minimum degree criterion is that it specifies the threshold irrespective of the cluster size. That means, for low thresholds we will get potentially undesired weakly connected extensions of large clusters, whereas high thresholds *a priori* exclude small clusters. Therefore, it can be meaningful to replace the combination of density and minimum degree criteria by the following *minimum relative degree criterion.*

Definition 10.13 (Minimum Relative Degree Threshold) *The minimum relative degree threshold γ is satisfied for a cluster U if* $\deg_U(u)/(|U| - 1) \geq \gamma$ *for all $u \in U$.*

This condition considers the density with respect to each individual cluster node and thereby guarantees a certain balance in the distribution of the weight among the nodes in a cluster, which is a reasonable requirement in many applications. For unweighted graphs, this kind of pattern is known as *γ-quasi-clique* [14–17].

Definition 10.14 (γ-Quasi-Clique) *A node set U is a γ-quasi-clique if each node has edges to at least $\lceil \gamma(|U| - 1) \rceil$ other nodes in U.*

It is easy to verify that a cluster satisfying the minimum relative degree threshold γ has a density of at least γ. On the other hand, for a cluster with density $\geq \gamma$ the minimum relative degree is not necessarily greater than or equal to γ. Figure 10.3 illustrates this relationship between cluster density and minimum relative degree. Unfortunately, we cannot mine directly for clusters that satisfy the minimum relative degree criterion, because it inherently does not allow to define an antimonotonic reduction scheme.

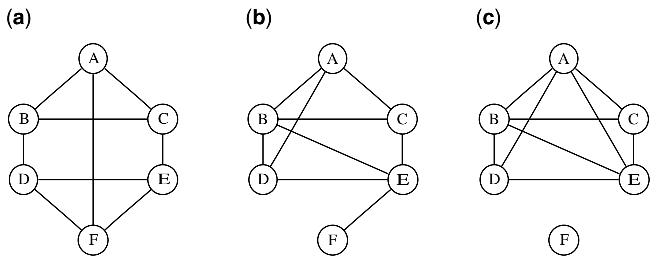


FIGURE 10.3 Minimum relative degree versus density. While the clusters (a–c) all have the same density (3/5), the minimum relative degree varies: (a) 3/5, (b) 1/5, (c) 0.

For instance, let us consider the cluster in Figure 10.3a. The minimum relative degree is 3/5. However, no matter which node we remove, the resulting cluster will have a minimum relative degree of 2/4. Therefore, we have to use a more general criterion during the search and perform a filtering step to select the actual solutions. One possible approach is to enumerate by reverse search all clusters with density $\geq \gamma$ and use similar strategies as with minimum degree thresholds to filter the clusters. For unweighted input graphs, alternative approaches have been developed, which are described in the following section.

10.3.7.1 Approaches to Quasi-Clique Mining

We briefly review the major techniques for γ-quasi-clique enumeration used in existing work [14–17]. The basic search strategy is depth-first search in a lexicographical set enumeration tree. As it lacks an antimonotonicity property with respect to the quasi-clique criterion, the pruning is based on other characteristics. The first rule is based on the diameter of γ-quasi-cliques. In general, the diameter of a subgraph is defined as the maximum shortest path length between any pair of nodes. For example, the graph in Figure 10.3a has diameter 2. It turns out that the diameter of a γ-quasi-clique U can be bounded in dependence of the minimum relative degree threshold γ and the number of nodes $|U|$.

Lemma 10.4 *Let U be a γ-quasi-clique ($|U| > 1$). Further, let* diam(U) *denote the diameter of U. Then,*

$$\text{diam}(U) \begin{cases} = 1 & \textit{if } 1 \geq \gamma > \frac{|U|-2}{|U|-1} \\ \leq 2 & \textit{if } \frac{|U|-2}{|U|-1} \geq \gamma \geq \frac{1}{2} \end{cases} .$$

For the proof and upper bounds of the diameter for smaller γ, we refer to Ref. [17]. With this lemma, we can restrict the set of candidate nodes for extending a given set of nodes:

Lemma 10.5 *Let $U \subset V$ be the current set of nodes and $u \in U$. Further, we denote by $N_V^b(u)$ the b-step neighborhood of u with respect to V, that is, all nodes in V that are reachable from u by a path of length $\leq b$. If there exists a γ-quasi-clique Q with $U \subset Q \subset V$, each node $v \in Q \setminus U$ satisfies*

$$v \in \bigcap_{u \in U} N_V^b(u) ,$$

where b is the upper bound of the diameter of a γ-quasi-clique.

Consequently, we can discard candidate nodes that are not included in the intersection of the b-neighborhoods of the current nodes. Beyond that, many additional refinements of γ-quasi-clique mining have been proposed, including degree-dependent pruning, look-ahead strategies, and candidate sorting (see Refs. 15–17 for details).

One problem of the existing quasi-clique mining approaches is that the underlying search tree is not antimonotonic. Consequently, indirect criteria have to be used to decide where pruning is possible. Often, one single criterion is not sufficient to achieve an efficient search. For instance, the diameter-based pruning can be problematic for input graphs that contain highly connected nodes ("hubs"), because the b-step neighborhoods of nodes and their intersection becomes very large. Therefore, it is crucial for the feasibility of the approach to combine several such criteria, as suggested in the literature [14–17].

As the cluster density criterion is a direct generalization of the γ-quasi-clique criterion and allows to construct an antimonotonic search tree, an interesting alternative could be to exploit the dense cluster enumeration strategy for quasi-clique mining. This is particularly promising considering the fact that pruning rules from previous quasi-clique mining methods can also be integrated into the framework, thereby combining the advantageous properties of both approaches to achieve a high pruning potential. Furthermore, it would be interesting to extend the quasi-clique pruning concepts to the minimum relative degree criterion for weighted input graphs. Finally, a combination of topology-based and weight-based criteria could also be fruitful for cluster detection in weighted graphs. Here, we focus on search criteria that consider exclusively the interactions within a cluster. However, depending on the application scenario, it might be desirable to take outgoing edges into account as well. In the literature, a number of different approaches to combine these two aspects have been studied (see, e.g., Refs. 37,39,106,107).

10.3.8 Integration of Node Weights

So far, the density criterion for clusters takes only interaction weights into account. However, node weights also play a role in many biological applications; they are commonly used to indicate the (measured or estimated) relevance of a node for the biological problem at hand. In this section, we discuss how to integrate node weights into the cluster mining process. In contrast to cluster finding approaches that use node weights to preprocess the input graph (i.e., remove low-weight nodes) [105], we directly incorporate them into the search criterion.

Let us consider again an input network with node set V. We assume that each node $i \in V$ has an assigned *node weight*, denoted by o_i. With this, we define the *node density* of a cluster.

Definition 10.15 (Node Density) *Given a cluster $U \subset V$ and node weights $o = (o_i)_{i \in V}$, the node density is defined as*

$$\rho_o(U) = \frac{1}{|U|} \sum_{i \in U} o_i \,. \tag{10.11}$$

That means, the node density corresponds to the average node weight within a cluster. For clarity, we use the term *interaction density* to refer to the previous definition of cluster density, the average pairwise interaction weight (see Definition 10.3). Now we introduce a *combined density* criterion, which includes interaction weights, node weights, and a calibration parameter α.

Definition 10.16 (Combined Density) *Given a cluster $U \subset V$, the combined density is defined as*

$$\rho_{\alpha,W,o}(U) = \tfrac{1}{1+\alpha}\left(\rho_W(U) + \alpha\,\rho_o(U)\right)\,, \tag{10.12}$$

where $\alpha \geq 0$.

In other words, the combined density is a weighted sum of the interaction density and the node density, where the node contribution obtains the α-fold weight of the interaction contribution. A straightforward method for enumerating all clusters with $\rho_{\alpha,W,o}(U) \geq \theta$ can be obtained by transforming the task to the basic cluster enumeration problem involving only interaction weights:

1. Construct a transformed interaction weight matrix W^{new}:

$$w_{ij}^{\text{new}} = \tfrac{1}{1+\alpha}\left(w_{ij} + \alpha\tfrac{(o_i+o_j)}{2}\right) \tag{10.13}$$

2. Use the standard DCE algorithm to enumerate all clusters U that satisfy the following criterion:

$$\rho_{W^{\text{new}}}(U) \geq \theta \tag{10.14}$$

The equivalence of $\rho_{W^{\text{new}}}(U)$ and $\rho_{\alpha,W,o}(U)$ follows directly from the definitions of interaction and node density. The transformed interaction weight matrix W^{new} is typically less sparse than the original interaction weight matrix W, that is, it has more nonzero values. Alternatively to the combined density criterion, it is conceivable to consider the two criteria separately and define individual thresholds for each of them; more generally, one could consider various types of criteria at the same time, also relating to several different networks. However, even if enumeration tasks for individual criteria can be solved with antimonotonic search trees, it is nontrivial to design efficient methods to search for patterns that satisfy all criteria. In particular, a given cluster does not necessarily have a subcluster that guarantees antimonotonicity with respect to all criteria and could serve as a parent in a global antimonotonic search tree [107]. One obvious solution strategy is to construct an efficiently prunable search tree with respect to the strictest criteria and filter the results afterwards according to the remaining criteria.

10.3.9 Constraint Integration

Often, it is desirable to restrict the dense cluster search by some additional criteria. The described enumeration framework allows us to incorporate and systematically

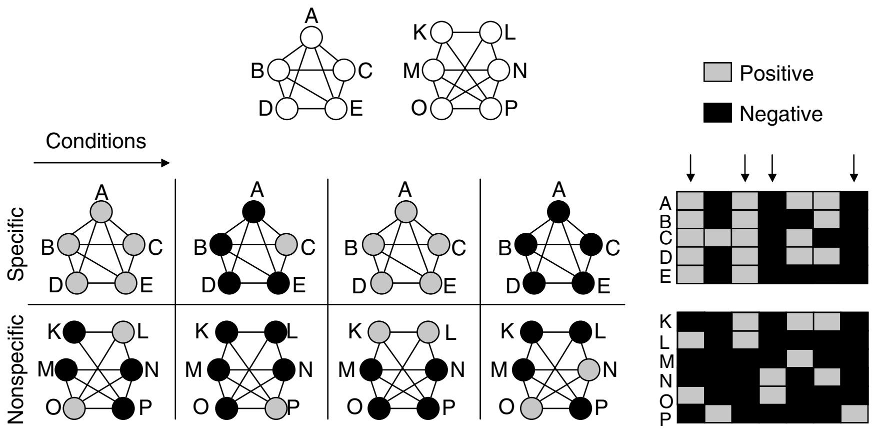


FIGURE 10.4 Integration of node profile data into network analysis. The combination of the interaction density criterion and node profiles allows to focus on clusters with consistent states of all nodes across a subset of conditions.

exploit various types of constraints defining further the cluster characteristics or involving external datasets. In the following subsections, we exemplarily present some constraints that actively contribute to pruning during the search; constraint-based output filtering has been discussed in Section 10.3.6.

10.3.9.1 Constraints from External Data Sources

Often, integration of background knowledge or other data sources in the mining process is crucial to the enhancement of the relevance of patterns. This particularly applies to systems biology studies. One useful criterion are consistency constraints with respect to node profile data [1], as illustrated in Figure 10.4. Given a discrete-valued profile for each node, we call a cluster consistent if all its nodes share a common subprofile. In the context of protein interaction data, we could for instance consider profiles that indicate presence or absence of proteins in different types of cells or subcellular compartments. In that case, clusters correspond to putative protein complexes that are "realizable" in living cells and therefore more plausible than clusters without protein co-occurrence. Formally, we define the concept of consistency as follows.

Definition 10.17 (Consistency) *Let V be the node set of the input network and $U \subset V$ a cluster. Let $X = (x_{ij})_{i \in V, j \in C}$ be an auxiliary profile matrix, where C denotes the set of columns (representing different conditions) and the entries x_{ij} take values from a set of discrete states S. Given a state $s \in S$, a particular column $c \in C$ is said to be s-consistent with respect to U if*

$$x_{uc} = s \text{ for all } u \in U\,.$$

The number of s-consistent columns with respect to U is denoted by $f_s(U)$. The cluster U is consistent with respect to s if

$$f_s(U) \geq n_s\,,$$

where n_s is a prespecified (non-negative) integer.

To control the consistency of cluster profiles, we fix minimum thresholds n_s for the frequencies $f_s(U)$ of the different states s. Then, we can formulate the constrained dense cluster enumeration task.

Definition 10.18 (Dense Cluster Enumeration with Consistency Constraints) *Given a graph with node set V and weight matrix W, a density threshold $\theta > 0$, an auxiliary profile matrix $(x_{ij})_{i \in V, j \in C}$ with $x_{ij} \in S$, and integers n_s for all $s \in S$, find all clusters $U \subset V$ such that $\rho_W(U) \geq \theta$ and $f_s(U) \geq n_s$ for all $s \in S$.*

If we are only interested in consistency with respect to one specific state, the thresholds for the other states are simply set to 0. For example, one might look for protein clusters with consistent presence in a cellular compartment, ignoring patterns of coordinated absence. Such constraints can lead to a considerable speed-up of the search procedure. Namely, we can exploit pruning techniques from traditional frequent itemset mining [69], based on the observation of antimonotonicity (see Definition 10.5).

Lemma 10.6 *The consistency criterion satisfies the antimonotonicity property, that is,*

$$U' \subset U \implies f_s(U') \geq f_s(U)\,.$$

The proof is obvious. In other words, cluster extension cannot increase the frequency of consistent columns, see Figure 10.5 for illustration. When adding node F to the cluster {A,B,C,D,E}, the number of consistent 1-columns shrinks, whereas the number of consistent 0-columns is left unchanged. To determine the frequencies of consistent columns for {A,B,C,D,E,F}, only the columns of F that were consistent

Cluster U	$f_1(U)$	$f_0(U)$
{A,B,C,D,E}	2	2
{A,B,C,D,E,F}	1	2

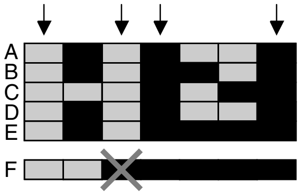


FIGURE 10.5 Antimonotonicity of consistency constraints. Given the example profile on the right, the table shows the frequencies of consistent columns for the 1-state (gray) and for the 0-state (black) before and after cluster extension.

with respect to {A,B,C,D,E} need to be considered. The lemma implies that we can prune the search tree whenever a cluster does not satisfy the frequency requirements.

The described framework can be applied to incorporate any kind of antimonotonic constraints. Furthermore, one can use arbitrarily many of these constraints at the same time, potentially involving several different data sources. Beside profile consistency, another prominent example of antimonotonic constraints on external data are *node pair constraints*. Assume we have an additional weight matrix containing an entry for each pair of nodes. This could be a similarity or distance matrix that is based on another criterion or comes from another data source. Then, constraints that define minimum or maximum weight thresholds for all node pairs in a cluster are antimonotonic, if a cluster contains a pair that violates the threshold, all its superclusters will not satisfy the constraint. Finally, not all constraints that are potentially interesting have the antimonotonicity property. This applies for instance to requirements specifying properties for a minimum percentage of cluster nodes. Sometimes it might be possible to derive bounds for pruning; otherwise, they are only used for output filtering, without accelerating the search.

10.3.9.2 Connectivity Constraints

The density criterion for clusters does not necessarily guarantee the connectivity of the corresponding subgraph. A subgraph is connected if every node is reachable from every other node via a path, that is, a series of (nonzero) edges. For simplicity, we here focus on interaction matrices with non-negative weights, although extensions to mixed-sign weights are possible. As already mentioned in Section 10.3.6, a cluster might tolerate a certain fraction of weakly connected or even disconnected nodes; also, it might fall apart into separate components. These problems arise in particular for large clusters or low density thresholds.

However, in many applications it is meaningful to consider only connected clusters as results. Unfortunately, we nevertheless have to visit disconnected clusters during the search because the connectivity property is not antimonotonic with respect to the density-based reverse search tree; that is, connected clusters may descend from disconnected ones, as the example in Figure 10.6 shows. An easy way to check whether a cluster is connected is a depth-first search traversal of the induced subgraph.

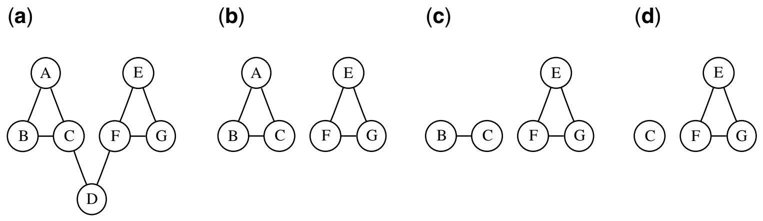


FIGURE 10.6 Example reduction of a cluster. It shows that a connected cluster (a) can have disconnected ancestors (b–d). In particular, one ancestor has an isolated node (d). For the uniqueness of parents, a lexicographical order on the nodes is assumed.

However, the connectivity criterion can also contribute to the pruning of the density-based search tree.

Definition 10.19 (Isolated Node) *Given a cluster U, a node $u \in U$ is called isolated if* $\deg_U(u) = 0$.

Lemma 10.7 *Let $W = (w_{ij})_{i,j \in V}$ be a weight matrix representing the input network such that $w_{ij} > 0$ if the nodes i and j are connected by an edge, and $w_{ij} = 0$ otherwise. Then, a cluster with two isolated nodes cannot have any connected descendant.*

Proof: Let us consider a connected cluster, on which we iteratively apply the reduction scheme, that is, in each step we transform the current cluster into its direct ancestor in the search tree (Definition 10.8). During this process, we might encounter clusters that include an isolated node (see Fig. 10.6). Whenever this happens, the isolated node(s) will be removed in the next reduction step(s) because they are the minimum degree nodes; as removals of isolated instances leave the degrees of other nodes unchanged, they cannot produce new isolated nodes. Hence, isolated nodes cannot be accumulated during the reduction. It remains to show that a single reduction step on a cluster without isolated nodes cannot produce two or more isolated nodes. Let us assume this would be possible. We denote by U the original cluster, that is, $\deg_U(u) > 0$ for all $u \in U$. Further, let $u^* \in U$ be the node that is removed, and let $u_1, u_2 \in U \setminus \{u^*\}$ be the isolated nodes in the resulting cluster, that is, $\deg_{U\setminus\{u^*\}}(u_1) = \deg_{U\setminus\{u^*\}}(u_2) = 0$. As $\deg_U(u_1) > 0$ and $\deg_U(u_2) > 0$, there have to exist edges $\{u_1, u^*\}$ and $\{u_2, u^*\}$ with $w_{u_1u^*} > 0$ and $w_{u_2u^*} > 0$. This implies $\deg_U(u^*) \geq w_{u_1u^*} + w_{u_2u^*} > w_{u_1u^*} = \deg_U(u_1)$ and analogously $\deg_U(u^*) > \deg_U(u_2)$, which is a contradiction to $U \setminus \{u^*\}$ being the parent of U. ■

This implies that we can refrain from extending clusters that have two isolated nodes. By additionally activating the minimum degree filtering step explained in Section 10.3.6, one can make sure that any cluster with isolated nodes is eliminated from the output. Clearly, the absence of isolated nodes does not guarantee yet that the cluster is indeed connected. Other connectivity-based pruning rules are possible. In fact, there exists a reverse search tree to enumerate connected node sets [89], but it is different from the density-based reverse search tree, and an effective combination of two search trees is generally difficult (see Section 10.3.8). However, for non-negative input data and density-cutoffs greater than 0.5, one can construct antimonotonic search schemes to enumerate patterns that are both dense and connected [108,109].

10.3.9.3 Cardinality and Branching Restrictions

Sometimes it is possible to specify in advance a size range for the clusters of interest. As our search strategy extends the clusters by exactly one node in each step, it can naturally respect thresholds for the maximum number of nodes in a cluster. Minimum cardinality constraints, on the contrary, are a popular means to eliminate insignificant results. Being nonantimonotonic, they do not explicitly contribute to speed up the search procedure. However, in the case of additional minimum (relative) degree constraints (Section 10.3.6), we can exploit minimum cardinality constraints to *a priori* remove low-degree nodes from consideration.

Due to the completeness of the dense cluster enumeration, the method might visit a large number of overlapping clusters, in particular, it will consider all dense subclusters of a large solution cluster. A simple way to control the generation of similar clusters are *maximum branching constraints*. While maximum cardinality thresholds constrain the depth of a search tree, branching criteria constrain the width of subtrees by restricting the maximum number of children per cluster to k. Consequently, the number of clusters sharing the same ancestor is limited. Of course, this heuristic strategy leads to the loss of the completeness guarantee; in particular, the exact p-values introduced in Section 10.3.5.2 cannot be determined anymore. But if we select the most "promising" children in each step, the method is likely to find a substantial fraction of the most significant solutions. This idea is related to the concept of beam search [110]. We propose the following procedure for the selection of children, among all nodes v that produce children $U \cup \{v\}$ of the current cluster U, we choose the k nodes with the largest degree within $U \cup \{v\}$ (leading to child clusters with the largest density). The motivation behind this is that they are most likely to have dense descendants. Among nodes with equal degree, we prefer those with the largest indices because according to our reduction scheme, they are the last to be removed. In other words, we use an ordering of candidate nodes that is reverse to the ordering defined for the reduction scheme (Definition 10.8).

10.4 DENSE CLUSTER ENUMERATION IN HIGHER-ORDER ASSOCIATION DATA

So far, we have assumed undirected weighted networks as input data for the dense cluster enumeration approach. Figure 10.7 illustrates generalizations of this setting that are addressed in this section [3,4]. While the basic network case deals with symmetric weight matrices and extracts submatrix patterns of large average weight, a similar task can be defined for asymmetric two-way weight matrices with a different set of entities in each dimension. It can be considered as an instance of bicluster mining approaches that focus on the strength or density of associations between entity subsets (see Section 10.2.4, Refs. 49,65,67,111). Beyond that, we look at higher-order input data, which can be represented as n-way weight arrays, also known as tensors. A higher-order or n-way cluster is then a subtensor described by specifying a nonempty subset of indices in each dimension. From a graph-theoretic point of view, an n-way tensor corresponds to a weighted n-partite hypergraph, where each hyperedge connects n nodes, exactly one node from each partition. If different dimensions of the tensor share the same entity set, the corresponding hypergraph has less partitions, and inherent symmetry relationships can exist. After introducing the problem of n-way dense cluster enumeration and a generalized reverse search algorithm that solves it, we consider its extension to symmetric scenarios.

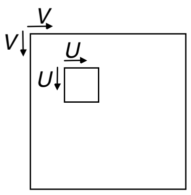
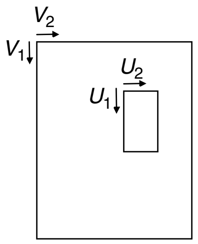
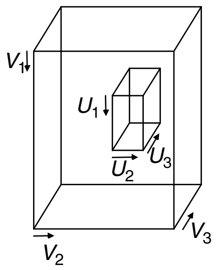
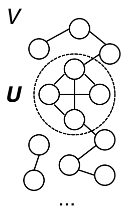
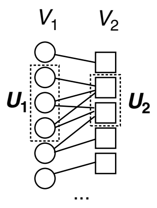
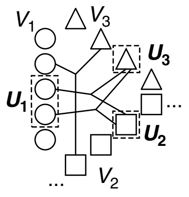
	Network Mining	Bicluster Mining	Higher-Order mining
Data	Symmetric weight matrix: $W: V \times V \to \mathbb{R}$, $W(u, v) = W(v, u)$	Two-way weight matrix: $V_1 \times V_2 \to \mathbb{R}$	Multiway weight tensor: $V_1 \times ... \times V_n \to \mathbb{R}$
Cluster definition	U $U \subset V$	(U_1, U_2) $U_i \subset V_i$	$(U_1, ..., U_n)$ $U_i \subset V_i$
Array representation			
Graph representation			

FIGURE 10.7 Generalization of cluster mining in networks to two-way and higher-order data. For visualization purposes, the sets U and U_i are shown as coherent blocks; however, they can be arbitrary subsets.

10.4.1 Motivation

A popular approach to investigate higher-order data is relational data mining (see Section 10.2.6). There, the focus is on binary-valued multiway relationships, also called n-ary relations. They can be represented as an n-way tensor where an entry is 1 if the corresponding n-way relationship has been observed, and 0 otherwise. In that framework, relational mining is equivalent to extracting subtensors (clusters) that contain only 1-entries; different clusters may overlap. In the relational data mining terminology, these patterns are called n-sets. Our approach to higher-order cluster detection extends the definition of n-set to numerical data, that means, the tensor may contain arbitrary weights. We consider a cluster or n-set pattern as a solution if the average value of the entries in the corresponding subtensor exceeds a given threshold; in particular, 0-entries are tolerated to a certain extent, whereas relational mining approaches require that all entries are 1. The generalization can be advantageous for detecting associations between sets of instances in data with missing or noisy observations. The crucial difference to previous methods for cluster discovery in multiway weight tensors (see Section 10.2.6) lies in the quality criterion for cluster patterns, which considers the average association strength within a cluster instead of the homogeneity of weights. Furthermore, overlapping clusters are detected, whereas many previous techniques are based on partitioning. The approach is generic in the sense that it can in principle deal with tensor data of an arbitrary number of dimensions, binary values or real weights, and partial symmetries.

10.4.2 Problem Definition

Our goal is to extract all dense clusters from a multidimensional data array (tensor). To formalize the problem, we first introduce some notation, generalizing the definitions from Section 10.3.1. Let $n > 0$ be the number of *dimensions* in the given data array (also called *ways* or *modes*). Then, we write the input in the following form:

$$W = (w_{k_1,\dots,k_n})_{k_i \in V_i,\ i=1,\dots,n} \tag{10.15}$$

The index k_i is used to access the ith dimension and takes values from a finite index set $V_i = \{1, \dots I_i\}$, where I_i is a natural number that can differ from dimension to dimension. V_i is also called the *instance set* or *range* for the ith dimension; the cardinality of the set is denoted by $|V_i|$ and equals I_i. The elements (entries) of W are real-valued weights indicating the association strength between the n instances. For convenience, we again normalize the array such that

$$w_{k_1,\dots,k_n} \leq 1 \quad \forall\, k_i \in V_i,\ i = 1, \dots, n\,. \tag{10.16}$$

An n-way *cluster* U is defined by specifying for each dimension a nonempty subset of the corresponding index set,

$$U = (U_1, \dots, U_n), \quad U_i \subset V_i,\ |U_i| \geq 1 \quad \forall i = 1, \dots, n\,. \tag{10.17}$$

The induced subarray is given by

$$W|_U = (w_{k_1,\dots,k_n})_{k_i \in U_i,\ i=1,\dots,n}. \tag{10.18}$$

Let us define the *cardinality* of a cluster as the sum of the cardinalities of the index subsets in all n dimensions, that is, the total number of instances included in the cluster:

$$\mathrm{card}(U) = \sum_{i=1}^{n} |U_i|. \tag{10.19}$$

This is not to be confused with the *cluster size*, which corresponds to the number of entries in the induced subarray,

$$\mathrm{size}(U) = \prod_{i=1}^{n} |U_i|. \tag{10.20}$$

Our cluster definition implies that size$(U) \geq 1$. The *density* of the cluster U is defined as the average value of the weight entries in the induced subarray.

Definition 10.20 (Cluster Density) *The density of an n-way cluster U with respect to the n-dimensional weight array W is given by*

$$\rho_W(U) = \frac{1}{\text{size}(U)} \sum_{k_i \in U_i} w_{k_1,\ldots,k_n} \,. \tag{10.21}$$

Due to the normalization of the data array W, the largest possible cluster density is 1. Using the above definitions, we state the problem of dense cluster enumeration as follows.

Definition 10.21 (Higher-Order Dense Cluster Enumeration) *Given a weight-normalized n-dimensional data array W and a minimum density threshold θ with $0 < \theta \leq 1$, find all n-way clusters U such that $\rho_W(U) \geq \theta$.*

Note that different clusters are allowed to overlap. For $\theta = 1$, the problem is equivalent to n-set or hyperclique enumeration [72–75].

10.4.3 Enumeration Approach

In order to solve the dense cluster enumeration task in higher-order data, we again use a reverse search algorithm, extending the network-based enumeration strategy from Section 10.3. For that purpose, we first introduce an index mapping scheme that facilitates the algorithmic description; then, we specify the level-wise search space, establish a reduction scheme, and present the overall search procedure. Finally, we consider some implementation details and analyze the complexity of the method.

10.4.3.1 Global Index Representation

As defined in Equation 10.15, an n-way array is represented using dimension-specific index sets $V_1, \ldots, V_n$; each of them consists of successive indices starting from 1. Now we build a *global index* set across all dimensions:

$$\mathcal{V} = \{1, \ldots, \sum_{i=1}^{n} |V_i|\} \,. \tag{10.22}$$

The conversion of an element $v \in V_i$ to a global index is carried out according to the following scheme:

$$\mathcal{C}(v, i) = v + \sum_{j=1}^{i-1} |V_j| \tag{10.23}$$

For $i = 1$, the summation term is zero, that is, $\mathcal{C}(v, 1) = v$. Accordingly, we determine the array dimension to which an element $v \in \mathcal{V}$ belongs as

$$\dim(v) = \max\{i = 1, \ldots, n : \sum_{j=1}^{i-1} |V_j| < v\} . \tag{10.24}$$

Then a cluster $U = (U_1, \ldots, U_n)$ can also be represented as a subset of $\mathcal{V}$,

$$\mathcal{U} = \bigcup_{i=1}^{n} \bigcup_{u \in U_i} \{\mathcal{C}(u, i)\} . \tag{10.25}$$

Note that U and $\mathcal{U}$ are alternative representations of a uniquely determined cluster and can easily be transformed into each other. In the following, we will use the representation that is more convenient in the particular context.

10.4.3.2 Search Space

The search space for the dense cluster enumeration problem is the set of all possible n-way clusters. As in the module enumeration setting, it can be organized in the form of a lattice, that is, in multiple levels. Here, the root level consists of all *trivial* clusters.

Definition 10.22 (Trivial Cluster) *A cluster $U = (U_1, \ldots, U_n)$ is called trivial if $|U_i| = 1$ for $i = 1, \ldots, n$. Consequently,* $\text{size}(U) = 1$ *and* $\text{card}(U) = n$ *for each trivial cluster U.*

A trivial cluster corresponds to exactly one entry of the multidimensional array. By adding exactly one index to one particular set U_i, we obtain the clusters on the subsequent level of the search lattice. In this way, the clusters are iteratively expanded from level to level; at each level, the cluster index set $\mathcal{U}$ grows by one element and the cluster cardinality increases by 1. To traverse the search lattice in an efficient way, we define a search tree for each trivial cluster such that the resulting set of trees is a spanning forest of the search space. The next section describes a reduction scheme to construct search trees that allow for effective pruning based on the density criterion.

10.4.3.3 Reduction Scheme

The core component of a reverse search algorithm is the definition of a reduction scheme (i.e., the rule for parent construction) that guarantees antimonotonicity and completeness of the search. For the reduction of dense multiway clusters, we extend the Definition 10.8. First, we define the degree of an instance in a multidimensional array.

Definition 10.23 (Degree) *Given a cluster U, the degree of $v \in U_j$ with respect to U is defined as*

$$\deg_U(v, j) = \sum_{k_i \in U_i, i \neq j} w_{k_1, \ldots, k_{j-1}, v, k_{j+1}, \ldots, k_n} . \tag{10.26}$$

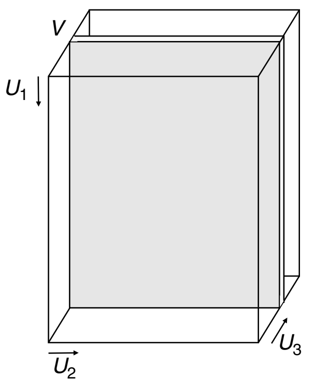


FIGURE 10.8 Visualization of a three-way cluster. In each reduction step, we remove one slice of the cluster, specified by a particular index element v.

In the global index representation, there is no ambiguity for instances of different dimensions, so we simply write $\deg_{\mathcal{U}}(v)$ for $v \in \mathcal{U}$. With that, we specify the following reduction scheme.

Definition 10.24 (Reduction Scheme) *Let $\mathcal{U}$ be a cluster. If v is the instance with the smallest index among the minimum degree elements in $\mathcal{U}$, the parent of $\mathcal{U}$ is given by $\mathcal{U} \setminus \{v\}$.*

Reducing the cluster $\mathcal{U}$ by one instance corresponds to removing a *slice* of the respective subarray, namely all entries involving the specified instance (see Fig. 10.8). Here, we select an instance such that the sum of the entries in the corresponding slice (i.e., the degree) is minimal. It remains to show that this parent–child relationship satisfies the cluster reachability and antimonotonicity requirements.

Lemma 10.8 *Let $\mathcal{U}$ be a nontrivial cluster and $\rho_W(\mathcal{U}) > 0$. Then, for all $v \in \mathcal{U}$ with minimum degree, that is,*

$$v \in \operatorname*{argmin}_{u \in \mathcal{U}} \deg_{\mathcal{U}}(u),$$

the following properties hold: $\mathrm{size}(\mathcal{U} \setminus \{v\}) \geq 1 \wedge \rho_W(\mathcal{U} \setminus \{v\}) \geq \rho_W(\mathcal{U})$.

The proof is straightforward (see Ref. 4). The first statement of the lemma refers to the cluster reachability; it ensures that, by iterative application of the reduction scheme in Definition 10.24, any cluster with positive density shrinks to a trivial cluster, that is, a root of the search space; that means, degenerate constructs do not occur where some dimensions-specific instance sets are empty. The second statement implies antimonotonicity, that is, a parent cluster is at least as dense as any child cluster.[6] Note that

[6]However, the parent is not necessarily the densest direct subcluster of a given child cluster.

TABLE 10.3 Dense Cluster Enumeration for an n-Dimensional Data Array with Global Index Set $\mathcal{V}$ (and Corresponding Mapping $\mathcal{C}$); θ Denotes the Minimum Density Threshold

```
1: DCE ($\mathcal{V}$, $\mathcal{C}$, $W$, $\theta$) :
2:     for each $(k_1, \ldots, k_n)$ with $w_{k_1,\ldots,k_n} \geq \theta$ do
3:         DCE_Rec($\mathcal{V}$, $W$, $\theta$, $\bigcup_{i=1}^{n}\{\mathcal{C}(k_i, i)\}$)
4:     end for

1: DCE_Rec ($\mathcal{V}$, $W$, $\theta$, $\mathcal{U}$) :
2:     for each $v \in \mathcal{V} \setminus \mathcal{U}$ do
3:         if $\rho_W(\mathcal{U} \cup \{v\}) \geq \theta$ and $\mathcal{U} \cup \{v\}$ is child of $\mathcal{U}$ then
4:             DCE_Rec ($\mathcal{V}$, $W$, $\theta$, $\mathcal{U} \cup \{v\}$)
5:         end if
6:     end for
7:     output $\mathcal{U}$
```

these properties hold for any minimum degree instance; however, to avoid duplicate investigation of subspaces, each cluster should have a unique parent, that is, the reduction scheme has to specify which of the minimum degree instances is selected (in our case, the instance with the smallest global index).

10.4.3.4 Search Algorithm

The defined reduction scheme directly suggests the algorithm in Table 10.3. The first step consists in finding all entries in the array that are greater than or equal to θ. These trivial clusters are then further expanded by a depth-first strategy producing descendants of increasing cardinality and pruning low-density branches, in analogy to the network cluster enumeration (Table 10.1). To be able to deal with an arbitrary number of dimensions, the input is conveniently represented in a sparse format. For each nonzero entry, we create a data object that contains the n-dimensional index vector and the corresponding value. To facilitate the access to entries during the search, we generate for each $v \in \mathcal{V}$ a list of pointers to the objects containing v (also called *adjacency list* of v). For efficient checking of the density and the child conditions during the search, we maintain an array of length $|\mathcal{V}|$ for storing degree values, equivalently to the network case (Section 10.3.3). One of the rules for quick parent checking has to be adjusted to the higher-order setting, which is as follows.

Lemma 10.9 *Given a cluster $\mathcal{U}$ in the data array W, let $u^* \in \mathcal{U}$ be the previously added instance and $v \in \mathcal{V} \setminus \mathcal{U}$. Let $g_{\mathcal{U}}(u^*, v)$ be the number of elements that the u^*-slice gains by adding v to the cluster $\mathcal{U}$. If $\deg_{\mathcal{U}\cup\{v\}}(v) > \deg_{\mathcal{U}}(u^*) + g_{\mathcal{U}}(u^*, v)$ then $\mathcal{U} \cup \{v\}$ is not a child of $\mathcal{U}$.*

Here, the notion of v-slice simply refers to the set of entries in the cluster subarray that include the instance v (see Fig. 10.8).

The complexity of the algorithm is analyzed in a similar way as for the network case described in Section 10.3.4. Depending on the application at hand, either a sparse input representation as described above or a full multidimensional array representation with constant-time access to specific entries is more suitable. Irrespective of the chosen data representation, the following theorem holds [4].

Theorem 10.2 *Using the reverse search algorithm for dense cluster enumeration in a given n-way tensor, the delay between two consecutive solutions is linear in the size of the data structure that represents the input.*

To facilitate a carefully directed analysis of result patterns, we can employ similar techniques as proposed for cluster enumeration in networks (Section 10.3). This includes for instance output filtering steps such as checks for local maximality and minimum degree thresholds. Relative degree values are equivalent to the average of the entries in the corresponding slice of the cluster (i.e., the density of the slice); constraining them ensures to a certain extent the balance of weights across the whole cluster. Such balance constraints can also be defined at finer granularity levels, that is, lower-order slices or fibers of the cluster. Furthermore, exact probabilities for ranking and pruning rules based on isolated instances can be directly generalized from the network case; equivalently, external constraints and branching restrictions can be handled (see Ref. 4 for details).

10.4.3.5 Symmetry Adaptations

The previous subsections considered the search for multiway cluster patterns with an individual subset specification for each dimension. However, multiway data might represent homogeneous or partially homogeneous associations, that is, different dimensions of the array can refer to the same set of entities, sometimes making a symmetric cluster analysis more appealing [4]. Here, we briefly discuss how to deal with full or partial symmetry scenarios. In particular, this extension of the higher-order cluster enumeration formalism restores the dense subgraph enumeration task from Section 10.3 as a special case. An example for a partially symmetric dataset is a set of weighted undirected networks that share the same set of nodes; they can be stored in a three-dimensional array where an entry w_{ijk} corresponds to the weight of the edge between the ith and the jth node in the kth network and the entries w_{ijk} and w_{jik} are equivalent (we say that the array is *symmetric* with respect to the first two dimensions). This has several consequences for the cluster enumeration. First, equivalence of entries must be respected when computing the cluster density; in particular, it is sufficient to store only one of them. Second, different dimensions of the input data might share the same set of instances (identified by their global indices); in that case, a cluster pattern defines a common instance subset for these dimensions instead of using separate subsets in each dimension. So, in the above example, a cluster would simply specify a subset of network nodes and a subset of networks, corresponding to node interaction patterns across several networks. Apart from that, various combinations of symmetry relationships in the data or requirements for the cluster can be handled simultaneously.

10.5 DISCUSSION

This chapter presented a general framework for the systematic extraction of dense patterns from simple or higher-order edge-weighted network data. It extends conventional relational set mining approaches [72,74,75] and clique-related network analysis [13,15,17]. The proposed reverse search algorithm allows for an effective, antimonotonicity-based pruning of the search space without missing any solutions; the complexity of the delay between two consecutive solutions is in the order of the input size. This property allows to apply it in cases where straightforward set enumeration algorithms are infeasible [4]. However, for large datasets or low density thresholds, the number of solutions can be prohibitive, making the computation slow in spite of the favorable runtime per solution.

There are several remedies for this problem. The first possibility is to maintain the enumerative search, but add further constraints based on additional criteria, prior knowledge, or external data [1], as described in Section 10.3.9. If relevant subsets are prespecified for some dimensions (for instance, windows of consecutive time intervals), reverse search with respect to the other dimensions can be performed for each of these subsets individually. On the other hand, one can use the reverse search strategy and additionally apply heuristic criteria or sampling techniques to control the number, overlap, and relevance of solutions; this allows to directly trade off the runtime and the completeness of the solution set, as exemplified in Section 10.3.9 with a simple branching heuristic [4]; similarly, heuristic pruning rules could be specified by appropriate thresholding of (relative) degree values. Even if it is not used for exhaustive exploration, the definition of the antimonotonic search space has a value by itself, as valid solutions are visited with short delay.

Finally, instead of applying the method to the whole dataset at once, it can be combined with different strategies of prepartitioning or preaggregation of the data [8,82]. Also, more effective exploitation of minimum size or connectivity constraints is conceivable. Furthermore, the reverse search strategy is compatible with distributed computation, and its efficiency can be enhanced by adapting data structures and pruning techniques to the specific task at hand.

REFERENCES

1. E. Georgii, S. Dietmann, T. Uno, P. Pagel, K. Tsuda, Enumeration of condition-dependent dense modules in protein interaction networks. *Bioinformatics* **25**(7), 933–940 (2009).
2. E.A. Akkoyunlu, The enumeration of maximal cliques of large graphs. *SIAM J. Comput.* **2**(1), 1–6 (1973).
3. E. Georgii, K. Tsuda, B. Schölkopf, Multi-way set enumeration in real-valued tensors, in *DMMT '09: Proceedings of the 2nd Workshop on Data Mining using Matrices and Tensors*, ACM, pp. 32–41 (Article No. 4), 2009.
4. E. Georgii, K. Tsuda, B. Schölkopf, Multi-way set enumeration in weight tensors. *Mach. Learn.* **82**, 123–155 (2011).

5. X. Yan, J. Han, gSpan: graph-based substructure pattern mining, in *ICDM '02: Proceedings of the 2nd IEEE International Conference on Data Mining*, IEEE Computer Society, pp. 721–724, 2002.
6. M. Kuramochi, G. Karypis, An efficient algorithm for discovering frequent subgraphs. *IEEE Trans. Knowl. Data Eng.* **16**(9), 1038–1051 (2004).
7. S. Kramer, L. De Raedt, C. Helma, Molecular feature mining in HIV data, in *KDD '01: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp. 136–143, 2001.
8. H. Hu, X. Yan, Y. Huang, J. Han, X.J. Zhou, Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics* **21**(Suppl. 1), i213–i221 (2005).
9. X. Yan, X.J. Zhou, J. Han, Mining closed relational graphs with connectivity constraints, in *KDD '05: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp. 324–333, 2005.
10. M. Kuramochi, G. Karypis, Finding frequent patterns in a large sparse graph. *Data Min. Knowl. Discov.* **11**(3), 243–271 (2005).
11. R.M. Karp, Reducibility among combinatorial problems, in *Complexity of Computer Computations*, Plenum Press, pp. 85–103, 1972.
12. V. Spirin, L.A. Mirny, Protein complexes and functional modules in molecular networks. *Proc. Natl. Acad. Sci. U.S.A.* **100**(21), 12123–12128 (2003).
13. 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).
14. J. Pei, D. Jiang, A. Zhang, Mining cross-graph quasi-cliques in gene expression and protein interaction data, in *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, IEEE Computer Society, pp. 353–354, 2005.
15. Z. Zeng, J. Wang, L. Zhou, G. Karypis, Coherent closed quasi-clique discovery from large dense graph databases, in *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp. 797–802, 2006.
16. G. Liu, L. Wong, Effective pruning techniques for mining quasi-cliques, in *ECML PKDD '08: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases – Part II*, Springer, pp. 33–49, 2008.
17. D. Jiang, J. Pei, Mining frequent cross-graph quasi-cliques. *ACM Trans. Knowl. Discov. Data* **2**(4), 1–42 (2009).
18. T. Uno, An efficient algorithm for enumerating pseudo cliques, in *Algorithms and Computation, Proceedings of the 18th International Symposium (ISAAC 2007)*, pp. 402–414, 2007.
19. Y. Asahiro, R. Hassin, K. Iwama, Complexity of finding dense subgraphs. *Discrete Appl. Math.* **121**(1–3), 15–26 (2002).
20. Y. Asahiro, K. Iwama, H. Tamaki, T. Tokuyama, Greedily finding a dense subgraph. *J. Algorithms* **34**(2), 203–221 (2000).
21. M. Charikar, Greedy approximation algorithms for finding dense components in a graph, in *APPROX '00: Proceedings of the 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization*, Springer, pp. 84–95, 2000.
22. U. Feige, G. Kortsarz, D. Peleg, The dense k-subgraph problem. *Algorithmica* **29**(3), 59–78 (2001).

23. G.F. Georgakopoulos, K. Politopoulos, MAX-DENSITY revisited: a generalization and a more efficient algorithm. *Comput. J.* **50**(3), 348–356 (2007).
24. L. Everett, L.S. Wang, S. Hannenhalli, Dense subgraph computation via stochastic search: application to detect transcriptional modules. *Bioinformatics* **22**(14), e117–e123 (2006).
25. G.D. Bader, C.W. Hogue, An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinform.* **4**, 2 (2003).
26. M.T. Dittrich, G.W. Klau, A. Rosenwald, T. Dandekar, T.Müller, Identifying functional modules in protein–protein interaction networks: an integrated exact approach. *Bioinformatics* **24**(13), i223–i231 (2008).
27. C. Faloutsos, K.S. McCurley, A. Tomkins, Fast discovery of connection subgraphs, in *KDD '04: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp. 118–127, 2004.
28. S.E. Schaeffer, Graph clustering. *Comput. Sci. Rev.* **1**(1), 27–64 (2007).
29. B.W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.* **49**(1), 291–307 (1970).
30. A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks. *Phys. Rev. E* **70**(6), 066111 (2004).
31. J. Hopcroft, O. Khan, B. Kulis, B. Selman, Natural communities in large linked networks, in *KDD '03: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp. 541–546, 2003.
32. A.W. Rives, T. Galitski, Modular organization of cellular networks. *Proc. Natl. Acad. Sci. U.S.A.* **100**(3), 1128–1133 (2003).
33. T. Yamada, M. Kanehisa, S. Goto, Extraction of phylogenetic network modules from the metabolic network. *BMC Bioinform.* **7**, 130 (2006).
34. E. Hartuv, R. Shamir, A clustering algorithm based on graph connectivity. *Inform. Process. Lett.* **76**, 175–181 (1999).
35. M. Girvan, M.E. Newman, Community structure in social and biological networks. *Proc. Natl. Acad. Sci. U.S.A.* **99**(12), 7821–7826 (2002).
36. J. Chen, B. Yuan, Detecting functional modules in the yeast protein–protein interaction network. *Bioinformatics* **22**(18), 2283–2290 (2006).
37. F. Luo, Y. Yang, C.-F. Chen, R. Chang, J. Zhou, R.H. Scheuermann, Modular organization of protein interaction networks. *Bioinformatics* **23**(2), 207–214 (2007).
38. J.W. Pinney, D.R. Westhead, Betweenness-based decomposition methods for social and biological networks, in *Interdisciplinary Statistics and Bioinformatics*, Leeds University Press, Leeds, pp. 87–90, 2006.
39. M.E. Newman, Modularity and community structure in networks. *Proc. Natl. Acad. Sci. U.S.A* **103**(23), 8577–8582 (2006).
40. G. Karypis, V. Kumar, Multilevel k-way partitioning scheme for irregular graphs. *J. Parallel Distrib. Comput.* **48**, 96–129 (1998).
41. D. Medini, A. Covacci, C. Donati, Protein homology network families reveal step-wise diversification of type III and type IV secretion systems. *PLoS Comput. Biol.* **2**(12) (2006).
42. U. von Luxburg, A tutorial on spectral clustering. *Stat. Comput.* **17**(4), 395–416 (2007).
43. S. van Dongen, *Graph Clustering by Flow Simulation*, PhD thesis, University of Utrecht, 2000.

44. A.J. Enright, S. van Dongen, C.A. Ouzounis, An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res.* **30**(7), 1575–1584 (2002).
45. E. Segal, H. Wang, D. Koller, Discovering molecular pathways from protein interaction and gene expression data. *Bioinformatics* **19**(Suppl. 1), i264–i271 (2003).
46. W. Chu, Z. Ghahramani, R. Krause, D.L. Wild, Identifying protein complexes in high-throughput protein interaction screens using an infinite latent feature model, in *Proceedings of the Pacific Symposium on Biocomputing*, pp. 231–242, 2006.
47. J.A. Parkkinen, S. Kaski, Searching for functional gene modules with interaction component models. *BMC Syst. Biol.* **4**, 4 (2010).
48. D. Hanisch, A. Zien, R. Zimmer, T. Lengauer, Co-clustering of biological networks and gene expression data. *Bioinformatics* **18**(Suppl. 1), S145–S154 (2002).
49. A. Tanay, R. Sharan, M. Kupiec, R. Shamir, Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. *Proc. Natl. Acad. Sci. U.S.A.* **101**(9), 2981–2986 (2004).
50. I. Ulitsky, R. Shamir, Identification of functional modules using network topology and high-throughput data. *BMC Syst. Biol.* **1**, 8 (2007).
51. I. Ulitsky, R. Shamir, Identifying functional modules using expression profiles and confidence-scored protein interactions. *Bioinformatics* **25**(9), 1158–1164 (2009).
52. T. Ideker, O. Ozier, B. Schwikowski, A.F. Siegel, Discovering regulatory and signalling circuits in molecular interaction networks. *Bioinformatics* **18**(Suppl. 1), S233–S240 (2002).
53. Y. Huang, H. Li, H. Hu, X. Yan, M.S. Waterman, H. Huang, X.J. Zhou, Systematic discovery of functional modules and context-specific functional annotation of human genome. *Bioinformatics* **23**(13), i222–i229 (2007).
54. X. Yan, M.R. Mehan, Y. Huang, M.S. Waterman, P.S. Yu, X.J. Zhou, A graph-based approach to systematically reconstruct human transcriptional regulatory modules. *Bioinformatics* **23**(13), i577–i586 (2007).
55. S.C. Madeira, A.L. Oliveira, Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **1**(1), 24–45 (2004).
56. A. Prelić, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, E. Zitzler, A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics* **22**(9), 1122–1129 (2006).
57. A. Tanay, R. Sharan, R. Shamir, Biclustering algorithms: a survey, in *Handbook of Computational Molecular Biology*, Chapman and Hall, 2005.
58. G. Li, Q. Ma, H. Tang, A.H. Paterson, Y. Xu, QUBIC: a qualitative biclustering algorithm for analyses of gene expression data. *Nucl. Acids Res.* **37**(15), e101 (2009).
59. I. Van Mechelen, H.H. Bock, P. De Boeck, Two-mode clustering methods: a structured overview. *Stat. Methods Med. Res.* **13**(5), 363–394 (2004).
60. D. Jiang, C. Tang, A. Zhang, Cluster analysis for gene expression data: a survey. *IEEE Trans. Knowl. Data Eng.* **16**(11), 1370–1386 (2004).
61. I.S. Dhillon, Co-clustering documents and words using bipartite spectral graph partitioning, in *KDD '01: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp. 269–274, 2001.
62. Y. Kluger, R. Basri, J.T. Chang, M. Gerstein, Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Res.* **13**(4), 703–716 (2003).

63. L. Lazzeroni, A. Owen, Plaid models for gene expression data. *Stat. Sin.* **12**(1), 61–86 (2002).
64. P. Wang, C. Domeniconi, K.B. Laskey, Latent dirichlet bayesian co-clustering, in *ECML PKDD '09: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, pp. 522–537, 2009.
65. A. Tanay, R. Sharan, R. Shamir, Discovering statistically significant biclusters in gene expression data. *Bioinformatics* **18**(Suppl. 1), S136–S144 (2002).
66. K. Sim, J. Li, V. Gopalkrishnan, G. Liu, Mining maximal quasi-bicliques to co-cluster stocks and financial ratios for value investment, in *ICDM '06: Proceedings of the 6th International Conference on Data Mining*, pp. 1059–1063, 2006.
67. C. Yan, J.G. Burleigh, O. Eulenstein, Identifying optimal incomplete phylogenetic data sets from sequence databases. *Mol. Phylogenet. Evol.* **35**(3), 528–535 (2005).
68. J. Besson, C. Robardet, L. De Raedt, J.-F. Boulicaut, Mining bi-sets in numerical data, in *KDID '06: Knowledge Discovery in Inductive Databases, 5th International Workshop*, Vol. 4747, *Lecture Notes in Computer Science*, Springer, pp. 11–23, 2006.
69. R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, Morgan Kaufmann, pp. 487–499, 1994.
70. C. Creighton, S. Hanash, Mining gene expression databases for association rules. *Bioinformatics* **19**(1), 79–86 (2003).
71. T. Uno, M. Kiyomi, H. Arimura, LCM ver. 2: efficient mining algorithms for frequent/closed/maximal itemsets, in *FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, 2004.
72. L. Cerf, J. Besson, C. Robardet, J.-F. Boulicaut, Data Peeler: contraint-based closed pattern mining in n-ary relations, in *SDM '08: Proceedings of the 8th SIAM International Conference on Data Mining*, pp. 37–48, 2008.
73. L. Cerf, J. Besson, C. Robardet, J.-F. Boulicaut, Closed patterns meet n-ary relations. *ACM Trans. Knowl. Discov. Data* **3**(1), 1–36 (2009).
74. R. Jäschke, A. Hotho, C. Schmitz, B. Ganter, G. Stumme, TRIAS: an algorithm for mining iceberg tri-lattices, in *ICDM '06: Proceedings of the 6th International Conference on Data Mining*, IEEE Computer Society, pp. 907–911, 2006.
75. L. Ji, K.-L. Tan, A.K.H, Tung, Mining frequent closed cubes in 3D datasets, in *VLDB '06: Proceedings of the 32nd International Conference on Very Large Data Bases*, VLDB Endowment, pp. 811–822, 2006.
76. T.G. Kolda, B.W. Bader, J.P. Kenny, Higher-order web link analysis using multilinear algebra, in *ICDM '05: Proceedings of the 5th IEEE International Conference on Data Mining*, IEEE Computer Society, pp. 242–249, 2005.
77. E. Acar, S. Çamtepe, B. Yener, Collective sampling and analysis of high order tensors for chatroom communications, in *Intelligence and Security Informatics*, Springer, pp. 213–224, 2006.
78. C.F. Beckmann, S.M. Smith, Tensorial extensions of independent component analysis for multisubject FMRI analysis. *Neuroimage* **25**(1), 294–311 (2005).
79. E. Acar, C. Aykut-Bingol, H. Bingol, R. Bro, B. Yener, Multiway analysis of epilepsy tensors. *Bioinformatics* **23**(13), i10–i18 (2007).

80. L. Zhao, M.J. Zaki, TRICLUSTER: an effective algorithm for mining coherent clusters in 3D microarray data, in *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, ACM, pp. 694–705, 2005.
81. S.E. Baranzini, P. Mousavi, J. Rio, S.J. Caillier, A. Stillman, P. Villoslada, M.M. Wyatt, M. Comabella, L.D. Greller, R. Somogyi, X. Montalban, J.R. Oksenberg, Transcription-based prediction of response to IFNβ using supervised computational methods. *PLoS Biol.* **3**(1), e2 (2004).
82. T.G. Kolda, B.W. Bader, Tensor decompositions and applications. Technical Report SAND2007-6702, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, November 2007.
83. T.G. Kolda, J.Sun, Scalable tensor decompositions for multi-aspect data mining, in *ICDM*, 2008.
84. A. Banerjee, S. Basu, S. Merugu, Multi-way clustering on relation graphs, in *SDM '07: Proceedings of the 7th SIAM International Conference on Data Mining*, 2007.
85. S. Jegelka, S. Sra, A. Banerjee, Approximation algorithms for tensor clustering, in *Algorithmic Learning Theory*, pp. 368–383, 2009.
86. C. Kemp, J.B. Tenenbaum, T.L. Griffiths, T. Yamada, N. Ueda, Learning systems of concepts with an infinite relational model, in *AAAI '06: Proceedings of the 21st National Conference on Artificial Intelligence*, AAAI Press, pp. 381–388, 2006.
87. Y.-R. Lin, J. Sun, P. Castro, R. Konuru, H. Sundaram, A. Kelliher, MetaFac: community discovery via relational hypergraph factorization, in *KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp. 527–536, 2009.
88. S. Džeroski, Multi-relational data mining: an introduction. *SIGKDD Explor. Newsl.* **5**(1), 1–16 (2003).
89. D. Avis, K. Fukuda, Reverse search for enumeration. *Discrete Appl. Math.* **65**, 21–46 (1996).
90. R. Rymon, Search through systematic set enumeration, in *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, pp. 539–550, 1992.
91. M.J. Zaki, S. Parthasarathy, M. Ogihara, W. Li, New algorithms for fast discovery of association rules, in *KDD '97: Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAI Press, pp. 283–286, 1997.
92. R.J. Bayardo, Jr., Efficiently mining long patterns from databases, in *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, ACM, pp. 85–93, 1998.
93. F. Zhu, X. Yan, J. Han, P.S. Yu, gPrune: a constraint pushing framework for graph pattern mining, in *PAKDD '07: Proceedings of the 11th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, Springer, pp. 388–400, 2007.
94. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, 3rd edn., The MIT Press, 2009.
95. L.A. Goldberg, Efficient algorithms for listing unlabeled graphs. *J. Algorithms* **13**(1), 128–143 (1992).
96. L.A. Goldberg, Polynomial space polynomial delay algorithms for listing families of graphs, in *STOC '93: Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, ACM, pp. 218–225, 1993.

97. J. Ramon, S. Nijssen, Polynomial-delay enumeration of monotonic graph classes. *J. Mach. Learn. Res.* **10**, 907–929 (2009).
98. S.-I. Nakano, T. Uno, Constant time generation of trees with specified diameter, in *Graph-Theoretic Concepts in Computer Science, 30th International Workshop*, Vol. 3353, *Lecture Notes in Computer Science*, Springer, pp. 33–45, 2004.
99. D. Gunopulos, H. Mannila, S. Saluja, Discovering all most specific sentences by randomized algorithms, in *ICDT '97: Proceedings of the 6th International Conference on Database Theory*, Springer, pp. 215–229, 1997.
100. D.-I. Lin, Z.M. Kedem, Pincer search: a new algorithm for discovering the maximum frequent sets, in *EDBT '98: Proceedings of the 6th International Conference on Extending Database Technology*, Springer, pp. 105–119, 1998.
101. K. Gouda, M.J. Zaki, Efficiently mining maximal frequent itemsets, in *ICDM '01: Proceedings of the IEEE International Conference on Data Mining*, IEEE Computer Society, pp. 163–170, 2001.
102. G. Bejerano, N. Friedman, N. Tishby, Efficient exact p-value computation for small sample, sparse, and surprising categorical data. *J. Comput. Biol.* **11**(5), 867–886 (2004).
103. M. Koyutürk, W. Szpankowski, A. Grama, Assessing significance of connectivity and conservation in protein interaction networks. *J. Comput. Biol.* **14**(6), 747–764 (2007).
104. H.C.M. Leung, Q. Xiang, S.M. Yiu, F.Y.L. Chin, Predicting protein complexes from PPI data: a core-attachment approach. *J. Comput. Biol.* **16**(2), 133–144 (2009).
105. R. Sharan, T. Ideker, B. Kelley, R. Shamir, R.M. Karp, Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. *J. Comput. Biol.* **12**(6), 835–846 (2005).
106. M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks. *Phys. Rev. E* **69**(2), 026113 (2004).
107. C. Robardet, Constraint-based pattern mining in dynamic graphs, in *ICDM '09: Proceedings of the 9th IEEE International Conference on Data Mining*, IEEE Computer Society, pp. 950–955, 2009.
108. J. Long, C. Hartman, ODES: an overlapping dense sub-graph algorithm. *Bioinformatics* **26**(21), 2788–2789 (2010).
109. P. Dao, R. Colak, R. Salari, F. Moser, E. Davicioni, A. Schönhuth, M. Ester, Inferring cancer subnetwork markers using density-constrained biclustering. *Bioinformatics* **26**(18), i625–i631 (2010).
110. R. Bisiani, Beam search, in *Encyclopedia of Articial Intelligence*, Wiley, pp. 56–58, 1987.
111. N. Mishra, D. Ron, R. Swaminathan, A new conceptual clustering framework. *Mach. Learn.* **56**(1–3), 115–151 (2004).

11

HYPONYM EXTRACTION EMPLOYING A WEIGHTED GRAPH KERNEL

Tim vor der Brück

Hyponyms are required for many applications in the area of natural language processing. Constructing a knowledge base with hyponyms manually requires a lot of work. Thus, many approaches were developed to harvest them automatically. However, most of them do not make use of deep semantic information but instead are based on surface representations. In this paper, we present a purely semantic approach, which is based on semantic networks. In the first step, hyponym hypotheses are extracted by application of deep semantic patterns. In the second step, the extracted hypotheses are validated employing a combined feature and graph kernel. Furthermore, a weighting scheme is described to weight the edges of the compared graphs. The graph kernel calculation is implemented in such a way that intermediate results are reused as much as possible to allow for a reasonable runtime. The evaluation shows that the graph kernel improves the evaluation results in contrast to a purely feature-based kernel. We expect that by optimizing the combination parameters for graph and feature kernels, a further improvement is possible.

11.1 INTRODUCTION

For many years, machine learning approaches were predominantly dealing with features, that is, attribute value pairs [1]. Such an approach is attractive for several reasons. First, feature values can easily be stored in a relational database. Second, operations like the scalar product or the radial basis function for the similarity calculation of two feature vectors are very efficient and can therefore be applied to mass-data. However,

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

with the feature-based approach it is not possible to express relationships between variables. Furthermore, the original data is often directly represented as graph and has no natural feature representation. A conversion of a graph representation into features can seriously degrade precision and recall. Therefore, graph-based machine learning methods (also called statistical relational learning) are becoming more and more popular in the past years. Often employed graph-based machine learning methods are graph kernels, conditional random fields, and graph substructure learning.

A graph kernel is a similarity function for two graphs where the matrix of kernel values is positive-semidefinite and symmetric. Graph kernels are often used together with support vector machines and nearest neighbor methods for classification of graphs. Applications are image classification based on recognized structures [2], semantic relation extraction from texts [3–7] or estimating certain properties (e.g.,toxicity) of molecules [8].

Conditional random fields (CRFs) are employed to automatically label graph nodes based on observations which are associated to the graph nodes. According to Lafferty et al. [9], a conditional random field is *a sequence modeling framework that has all the advantages of maximum entropy models but also solves the label bias problem in a principled way*. The label bias problem is a problem often occurring with maximum entropy models. It consists of the fact that the next chosen label can become almost or even fully independent from the observation if the number of possible succeeding states is small. Conditional random fields are for instance employed for part-of-speech taggers or named entity recognition. Named entity recognition is the problem of identifying named entities in a text and assigning them a semantic type such as organization, person, geographic object, and so on. In contrast to maximum entropy models or hidden Markov models, conditional random fields can consider long distance relationships. An example of such a relationship is that the same named entity must always be assigned the same semantic type independent of its location of occurrence [10]. Conditional random fields are also applied in the area of image processing, for instance to label image pixels with their content type (e.g.,animal, syk, water, and so on) [11].

A further popular task in graph-based machine learning is to identify often occurring substructures in graphs. The approach of Cook and Holder [12] is based on the minimum description length principle [13] . This principle states *that the best theory to describe a set of data is that theory which minimizes the description length of the entire data set*. Applied to subgraph discovery, it states that the optimal subgraph is the one that compresses the entire graph the most. An example application of this method is the prediction of the toxicity of molecules depending on their structure activity representations (SARs) [14]. For that often occurring SAR, substructures are extracted and compared with previously unseen molecules. An alternative method for substructure learning that is used for relation extraction from dependency trees was proposed by Snow et al. [15].

In this work, we are dealing with graph-based hyponym extraction. Hyponymy extraction can be divided into kernel- and pattern-based approaches. A kernel-based approach requires an annotated training set with dependency trees/DAGs and the associated annotation (hyponym: yes or no). For a new training instance, first a word pair is chosen for which the correct relation is required. The sentence is parsed by a dependency parser and the tree structure of the new sentence is compared with the tree structures of the annotated training set by means of a tree kernel. A support vector machine then distributes the word pair in the correct relation class.

A pattern-based approach applies a collection of patterns to the dependency tree of the sentence or directly to the surface structure. If a pattern is applicable, the placeholder variables in the pattern can be instantiated and a new relation is created.

Our method combines the kernel-based approach with the pattern-based approach. First, hyponymy hypotheses are created by applying a set of patterns. Instead of usual surface type or syntactic patterns we employ truly semantic patterns in the form of semantic networks. After the hyponymy hypotheses[1] are extracted, they are validated by a support vector machine [16] employing different kernels. We use a feature kernel based on a radial basis function and two graph kernels, which compare the semantic networks the hypotheses were extracted from.

This paper is organized as follows. First, we introduce related work in the area of hyponym harvesting in Section 11.2, especially concerning the use of patterns and kernel methods and discuss their drawbacks (see Section 11.3). Afterwards, we describe the MultiNet semantic network formalism, which is the basis of our work in Section 11.4. We then introduce support vector machines and optimization with kernel functions (see Section 11.5). The architecture of our hyponym extraction system is specified in Section 11.6. In Section 11.7, the graph kernel based on common walks, which is the heart of our extraction method, is described in detail. We then introduce several enhancements of this method (see Sections 11.8 and 11.9), which were required to make the graph kernel suitable for our task and introduce a weighting scheme. Afterwards, the linguistic feature kernel is described, which is employed in addition to the graph kernel (see Section 11.10). Finally, we show our evaluation results (see Section 11.11), give a conclusion and an outlook on possible further work (see Section 11.12).

11.2 RELATED WORK

Hyponym harvesting has attracted a lot of interest. A large number of approaches were developed so far. The approaches can be divided into pattern-based, kernel-based, and document clustering-based [17] methods. A typical text pattern consists of a sequence of words or sentence marks and two placeholder variables labeled with *hypernym* or *hyponym*. An example of such a pattern is the Hearst pattern [18], "*hyponym* and other *hypernym*." If matched to the sentence *The secretary and other politicians criticized the law*, the placeholder variable *hyponym* would be assigned to *secretary*, the variable *hypernym* to *politician*, and therefore the correct hyponymy relation "*secretary* is a hyponym of *politician*" is extracted. Such a surface-based approach is easy to realize and also quite limited. It fails for instance if an additional subclause is inserted, for example, *The secretary and, according to our information, a lot of other politicians criticized the law.* In this case, the given surface pattern can no longer be used for the extraction of the above-mentioned relation. This problem can

[1]Note that our approach is also used to extract instance-of relations.

be overcome by employing graph-based representations such as dependency trees. The patterns of such an approach are given by dependency subtrees. An approach to learn these patterns automatically was devised by Snow et al. [15]. For that, the path in the dependency tree is extracted, which connects the corresponding nouns with each other. To account for certain keywords indicating a hyponym relation like *such* (see first Hearst pattern) they added the links to the word on either side of the two nouns (if not yet contained) to the path too. Frequently occurring paths are then learned as patterns for indicating a hyponymy relation.

Another often used approach is the use of structure kernels, usually tree and sequence kernels. Assume that a relation $R_1 = R(a1, a2)$ should be compared with a relation $R_2 = R(a1', a2')$. Then the following kernels might be applied to determine the estimated similarity of the surrounding tree structures:

- *Argument Kernel*. $a1$ is compared with $a1'$, and $a2$ with $a2'$ for the conformity of their surface representations, lexical heads and entity types (e.g.,person or organization) [3].
- *N-Gram*. Bigrams, trigrams, and so on that show up in the surface representation between $a1$ and $a2$ are compared with the corresponding n-grams between $a1'$ and $a2'$ [3].
- *Link Sequence*. The conformity of the tokens (surface representation/lexical head/type/word class) in the dependency path from argument $a1$ to $a2$ with that from $a1'$ to $a2'$ is determined, that is, the ith token of the first path is compared with the ith token of the second [3–5].
- *Labels of the Dependency Paths*. All labels in the dependency path from $a1$ to $a2$ matching the labels of the path from $a1'$ to $a2'$ are determined, independently of their order (bag of words) [3].
- *Local Dependency*. The conformity of the dependency relations directly connected with the arguments $a1$ and $a1'$ ($a2$ and $a2'$, respectively) is determined. This measure can also be computed if no connection can be established between the two relation arguments for R_1 or/and R_2 [3].
- *Subtree Comparison*. The minimum subtrees, which contain both arguments are compared [5,19].

Hyponym extraction approaches based on document clustering usually try to convert the document hierarchy, as determined by a hierarchical clustering method, into a taxonomy [17,20].

11.3 DRAWBACKS OF CURRENT APPROACHES

Pattern-based approaches are currently very popular for hyponymy extraction. But there is the problem that either recall or precision is poor. Consider a pattern collection containing among other things the pattern *hyponym is a hypernym*, which is very often applicable, but the precision of the extracted hypotheses is rather low. If this pattern is removed then the recall of the entire extraction process is seriously degraded.

However, the precision probably increases. A second drawback is that the possibility of a pattern application is always a binary decision. A pattern is either applicable or not and therefore the pattern provides no quality estimate based on the semantic or syntactic sentence structure. Kernel-based approaches do not suffer these problems. However, for kernel approaches it is difficult to figure out, which sentence elements should be set as anchor points for the kernel. To compare all nouns with all other nouns would result in a very long runtime. In this work, the advantages of both approaches are combined. We extract hyponym hypotheses with quite general patterns and afterwards validate them with a graph kernel. In this way, the hypotheses extraction is very fast, and also the anchor points for the graph kernel are already determined and finally, we also get structure-based quality estimates.

Another drawback of current approaches is that they are usually based on words instead of concepts or word readings. Furthermore, a syntactic or surface representation is less suitable for semantic relation extraction. In MultiNet for instance, anthroponyms (person names) are already identified and represented as feature value structures. Moreover, the semantic network representation of several different syntactic representations often coincide if identical meaning is expressed. Thus, semantic patterns are more generally usable than patterns based on surface or syntactic representations.

11.4 SEMANTIC NETWORKS FOLLOWING THE MULTINET FORMALISM

The relation extraction described here is based on semantic networks that follow the MultiNet formalism, which was devised by Hermann Helbig [21]. The semantic networks are automatically created by applying a deep syntactico-semantic parser to a text corpus. A semantic network is a graph where the nodes represent concepts and the arcs represent relations between the concepts (or functions involving concepts). Lexicalized concepts correspond to word readings. A lexicalized concept name is given by the lemma and two numbers specifying the homograph and the sememe. For details on the numbering scheme see Ref. [21].

An example of such a semantic network is given in Figure 11.1. The central sentence node, which represents the meaning of the entire sentence, is labeled $c1$. The actor or agens (MultiNet relation: AGT) of the situation expressed by the sentence is a certain man, which is an instance (MultiNet relation SUB) of the generic concept *man.1.1*. The *ITMS function combines several concepts in a conjunction, in this case the concepts representing the two constituents: *cello* and *other instruments*.

Sample relations of MultiNet are given below:

- AGT. Conceptual role: agent
- ATTR. Specification of an attribute
- *ITMS. Function enumerating a set
- OBJ. Neutral object
- PRED. Predicative concept characterizing a plurality

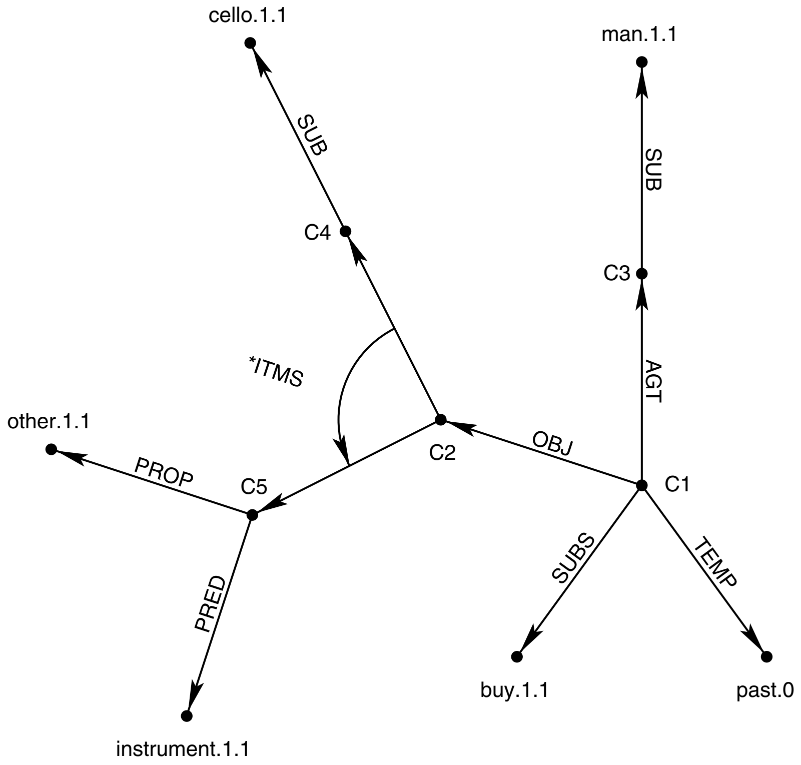


FIGURE 11.1 Example semantic network for the sentence *Der Mann kaufte ein Cello und andere Instrumente* / "*The man bought a cello and other instruments.*" For better readability, English concept names are used in the figure.

- PROP. Relation between object and property
- SUB. Relation of conceptual subordination (hyponymy and instance-of)
- SUBO. Relation of generalized subordination (hyponymy and instance-of), super-relation of SUB, SUBR, and SUBS
- SUBS. Relation of conceptual subordination for situations (hyponymy, instance-of, and troponymy)
- SUBR. Relation of conceptual subordination for relations
- TEMP. Relation specifying the temporal embedding of a situation
- VAL. Relation between a specific attribute and its value

11.5 SUPPORT VECTOR MACHINES AND KERNELS

A support vector machine [16] is a supervised machine learning method that distributes instances into two classes (extensions with more than two classes are also available). Using a set of training examples, a hyperplane is calculated that separates data of the two different classes from each other and maximizes the margin of the hyperplane. This margin is defined by the instances that are located closest to the hyperplane. Usually, a complete separation is not possible. Therefore, vectors on the wrong side of the hyperplane are allowed but penalized. The support vector optimization problem is given by [22]

$$\underset{\mathbf{w},\mathbf{b},\xi_i}{\operatorname{argmin}}\{0.5||\mathbf{w}||_2^2 + C\sum_{i=1}^{m}\xi_i\} \tag{11.1}$$

with the constraints

$$\begin{aligned} y_i(\langle \mathbf{w}, \mathbf{x_i}\rangle + b) \geq 1 - \xi_i,\ 1 \leq i \leq m \\ \xi_i \geq 0,\ \text{ with } 1 \leq i \leq m \end{aligned} \tag{11.2}$$

where $\mathbf{w} \in \mathbb{R}^n$, vector, orthographic to the hyperplane; $b \in \mathbb{R}$, parameter; $\mathbf{x_i} \in \mathbb{R}^n$, feature vectors; $y_i \in \{-1, +1\}$, class labels; $\langle ., .\rangle$, the scalar product; ξ_i, slack variables; $C \in \mathbb{R}^+$, a positive constant.

This optimization problem stated in Formulas 11.1 and 11.2 is given in the so-called primal representation. It can be converted into the equivalent dual representation

$$\underset{\alpha}{\operatorname{argmax}} \sum_{i=1}^{m}\left(\alpha_i - \frac{1}{2}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \langle \mathbf{x_i}, \mathbf{x_j}\rangle\right) \tag{11.3}$$

with the constraints

$$\begin{aligned} 0 \leq \alpha_i \leq C \\ \sum_{i=1}^{m}\alpha_i y_i = 0 \end{aligned} \tag{11.4}$$

where α, the vector to be determined by the optimization; $\mathbf{w} = \sum_{i=1}^{m}\alpha_i y_i \mathbf{x_i}$; $C \in \mathbb{R}^+$, a positive constant.

In recent implementations of support vector machines, the scalar product $\langle \mathbf{x_i}, \mathbf{x_j}\rangle$ can be replaced by an arbitrary user-provided kernel function K. The only conditions on such a function are that the matrix of kernel values is positive-semidefinite and symmetric. The dual representation allows it to apply a support vector machine on problems where the vector representations $\mathbf{x_i}$ are unknown or represented in an infinite space. Only the kernel function K has to be defined. Thus, the dual representation makes it possible to compare graphs and trees that have no natural vector representation. In the next sections, we demonstrate how support vector machines and a graph kernel can be employed for hyponymy harvesting.

11.6 ARCHITECTURE

In this section, we shortly give an overview of the total hyponymy extraction process.

1. In the first step the corpus containing the hyponyms (here the German Wikipedia) is parsed by the deep linguistic parser WOCADI[2] [23]. For that

[2]WOCADI is the abbreviation of *wo*rd-*c*l*a*ss *di*sambiguating parser.

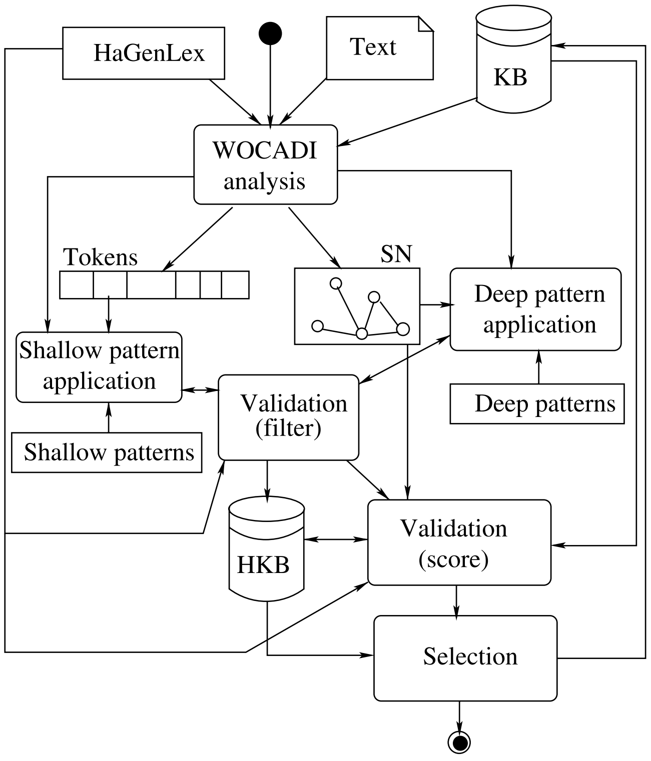


FIGURE 11.2 Activity diagram for the hyponym acquisition process.

WOCADI makes use of the semantic lexicon HaGenLex[3] [24] and a given knowledge base KB. The output of the WOCADI analysis for a single sentence is a token list, a dependency tree, and a semantic network.

2. Shallow extraction rules (similar to Hearst patterns) are applied on the token list.
3. Deep extraction rules are applied on the semantic network representation.
4. A validation module is applied that filters out incorrect hypotheses by looking on the semantic properties of these hypotheses [25].
5. Not all of the hypotheses that pass this filter are actually correct. Therefore, a support vector machine is additionally applied to validate the accepted hypotheses. Validation scores are calculated for all hypotheses and stored with them together in the hypotheses knowledge base HKB.
6. The best hypotheses of HKB, according to the scores, are stored in the knowledge base KB after manual inspection.

The entire validation process is illustrated in Figure 11.2.

[3]HaGenLex is the abbreviation of *Ha*gen *Ge*rma*n* *Lex*icon.

TABLE 11.1 A Selection of Deep Patterns

Definition	Example
$\textsc{subo}(a1, a2) \leftarrow \textsc{subo}(c, a1) \wedge$ $\textsc{pred}(e, a2) \wedge F_{*\textsc{itms}}(d, c) \wedge$ $F_{*\textsc{itms}}(d, e) \wedge H_{*\textsc{itms}}(c, e) \wedge$ $\textsc{prop}(e, \mathit{ander.1.1}(\mathit{other.1.1}))$	The secretary ($a1$) and other politicians ($a2$) criticized the law.
$\textsc{subo}(a1, a2) \leftarrow \textsc{pred}(c, a2) \wedge$ $\textsc{sub}(e, a1) \wedge F_{*\textsc{altn1}}(d, c) \wedge$ $F_{*\textsc{altn1}}(d, e) \wedge \textsc{prop}(c, \mathit{ander.1.1}(\mathit{other.1.1}))$	Do you have a cello ($a1$) or another string instrument ($a2$)?
$\textsc{subo}(a1, a2) \leftarrow \textsc{scar}(c, d) \wedge$ $\textsc{subo}(d, a1) \wedge \textsc{obj}(c, e) \wedge$ $\textsc{subo}(e, a2) \wedge$ $\textsc{subs}(c, \mathit{bezeichnen.1.1}(\mathit{denote.1.1}))$	A skyscraper ($a1$) denotes a very high building ($a2$).
$\textsc{subo}(a1, a2) \leftarrow \textsc{arg1}(d, e) \wedge$ $\textsc{arg2}(d, f) \wedge \textsc{subr}(d, \mathit{equ.0}) \wedge$ $\textsc{subo}(e, a1) \wedge \textsc{subo}(f, a2)$	This car ($a1$) is the best vehicle ($a2$) they offer.
$\textsc{subo}(a1, a2) \leftarrow \textsc{sub}(f, a2) \wedge$ $\textsc{temp}(e, \mathit{present.0}) \wedge \textsc{subr}(e, \mathit{sub.0}) \wedge$ $\textsc{sub}(d, a1) \wedge \textsc{arg2}(e, f) \wedge$ $\textsc{arg1}(e, d)$	The Morton number ($a1$) is a dimensionless indicator ($a2$).

$F_r(a_1, a_2)$: a_1 is the first argument of function r and precedes a_2 in the argument list. $H_r(a_1, a_2)$: a_1 and a_2 are arguments of function r and a_1 directly precedes a_2 in the argument list.

A deep extraction rule consists of a conclusion $\textsc{subo}(a1, a2)$ ($\textsc{subo}$: hyponymy/instance-of/troponymy relation) and a premise that is a semantic network where two of the nodes are labeled with the variables $a1$ and $a2$. The pattern network is tried to be matched to the sentence network. The variables can be bound to arbitrary concepts. The instantiated conclusion, where $a1$ and $a2$ are replaced by the concepts they were bound to, is the extracted hypothesis. Several example rules are given in Table 11.1. The extraction rules are in part manually specified and in part learn automatically from a collection of annotated semantic networks employing the Minimum Description Length Principle [26], basically following the approach of Cook and Holder [12].

11.7 GRAPH KERNEL

A graph kernel is a kernel function that compares different graphs. We use such a graph kernel to compare two hyponymy hypotheses with each other. In particular, we compare by means of a graph kernel the SNs with each other from which the hypotheses were extracted from. There exist a lot of different types of graph kernels.

The graph kernel of Borgwardt and Kriegel [27] compares the shortest paths in both graphs. These paths are determined for each graph separately. Two comparison functions were tested to compare the graph lengths: the product of the lengths and the function that takes the value of one if both lengths are identical and zero otherwise. The graph kernel value is then given by the sum of all such comparison function values iterating over all possible pairs of nodes in the two graphs. Kashima et al. [28] generate randomly common walks on the two graphs and count the number of walks that both graphs have in common. Another approach of Kashima, which he devised together with Hido, is to compare special kind of hash values computed for each graph [29].

Our work is based on the graph kernel as proposed by Gärtner et al. Like the approach of Kashima et al., the idea of this graph kernel is that graphs are similar if they share a lot of walks. Let us review some graph definitions before further explanations. A directed nonuniquely labeled graph is given by a set of nodes and arcs: $G = (V, E)$. Each node and arc is assigned a label: $label : G \cup E \to A^*$ where A is a given alphabet. This function is not injective, which means that different nodes or arcs can be mapped to the same label. A path in this graph is defined as a subgraph with a set of nodes $V = \{v_0, \ldots, v_k\}$ and arcs $E = \{e_0, \ldots, e_k\}$ with $e_i = (v_i, v_{i+1})$ and where all v_is must be distinct [30]. In contrast to a path, the same arc can be visited twice in a walk, which means that E is no longer a set but a sequence. Gärtner et al. use common walks instead of common paths because his proposed common walk kernel can easily be computed by matrix multiplications. For our hyponymy extraction scenario, we allow to follow an arc against the direction of the arc too if the associated arc in the other graph is also followed against its direction.

An example of a common walk in two SNs is given in Figure 11.3. Before calculating the common walks, the hyponym and hypernym candidates are replaced by fixed variables (here $a1$ and $a2$), which allows it to identify common walks involving hypernym and hyponym candidate nodes between SNs for sentences even if the

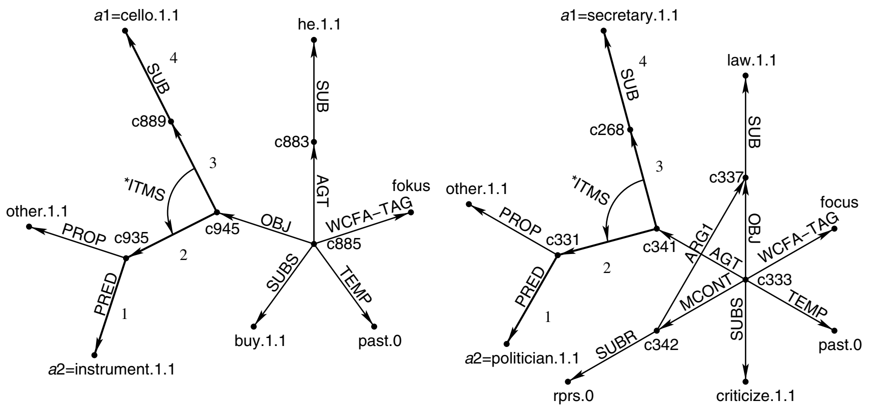


FIGURE 11.3 A common walk/path of length 4 in the disordered graphs.

hypernyms and hyponym candidates of the two sentences are different. In this way, the generality is increased.

The calculation of the kernel value is based on the product graph (*PG*) representation of the two graphs G_1 and G_2. These graphs are also called factor graphs of the product graph. The node set PV of the product graph consists of pairs of nodes of the two factor graphs. The first component of such a pair is a node of the first factor graph, the second component a node of the second graph, and both nodes must be assigned the same label. For our semantic network scenario, the label of a node is assigned the concept name if the concept is lexicalized (e.g., *school.1.1*) or *anon* (e.g.,if the concept is named *c935*) otherwise. Therefore, two nonlexicalized nodes always show up together as a node pair in the product graph.

$$\begin{aligned}
G_1 &= (V_1, E_1) \\
G_2 &= (V_2, E_2) \\
PG &= G_1 \times G_2 \\
PV &\subseteq V_1 \times V_2 \\
u \in PV &:\Leftrightarrow u = (s_1, s_2), s_1 \in G_1, s_2 \in G_2 \wedge \\
label(s_1) &= label(s_2)
\end{aligned}$$

In contrast to Gärtner et al., we also identify nodes with each other if the labels are not identical, but the associated concepts are synonymous.

An arc $e = (u_1, u_2)$ belongs to the product graph, if there exist an arc between the first node components of u_1 and u_2 in the first factor graph and an arc between the second node components in the second factor graph.

$$\begin{aligned}
(u_1, u_2) \in PE &:\Leftrightarrow u_1 = (s_1, s_2), u_2 = (t_1, t_2) \wedge \\
(s_1, t_1) &\in E_1, (s_2, t_2) \in E_2 \wedge \\
label(u_1) &= label(u_2)
\end{aligned}$$

The adjacency matrix $\mathbf{A}$ of the product graph is given by

$$\mathbf{A}_{PG} = (a_{xy}) \text{ with } a_{xy} = 1 :\Leftrightarrow (u_x, u_y) \in PE \vee (u_y, u_x) \in PE$$

where n is the number of nodes in the product graph. An entry (x, y) in the adjacency matrix of the product graph is one iff there exists one common walk from node u_x to u_y or from node u_y to u_x of length one. Thus, basically an undirected graph model is used in the product graph with the exception that the orientations of the two identified arcs in the two compared graphs have to be identical. More generally, there are k common walks of length i from node x to node y iff $a^i_{xy} = k$ where $(a^i_{xy}) = \mathbf{A}^i_{PG}$ is the adjacency matrix taken to the ith power.

The total graph kernel value is then given by[4]

$$K(G_1, G_2) = \iota^T \sum_{i=0}^{\infty} \lambda^i \mathbf{A}_{PG}^i \iota$$

where $\lambda \in \mathbb{R}$ and $\lambda < 1$. λ is a discounting factor and causes long common walks to be weighted less than short ones. This sounds quite irritating at first since a long common walk is a stronger indication of similarity than a short one. But a long common walk always implies a lot of shorter common walks. In this way, a long common is overall weighted more than a short one. ι is a vector consisting of only ones. $\iota^T \mathbf{M} \iota$ (ι^T denotes the transposed vector) causes the entries of the surrounded matrix to be summated. $\mathbf{A}_{PG}^0$ is defined to be the identity matrix $\mathbf{I}$. There is an analytical solution of this infinite sum if this sum converges. This becomes obvious after several transformations

$$\begin{aligned} \left(\sum_{i=0}^{\infty} \lambda^i \mathbf{A}_{PG}^i\right) - \left(\sum_{i=1}^{\infty} \lambda^i \mathbf{A}_{PG}^i\right) &= \mathbf{I} \\ \left(\sum_{i=0}^{\infty} \lambda^i \mathbf{A}_{PG}^i\right) - \mathbf{A}_{PG}\lambda \left(\sum_{i=0}^{\infty} \lambda^i \mathbf{A}_{PG}^i\right) &= \mathbf{I} \end{aligned} \tag{11.5}$$

Let S be $(\sum_{i=0}^{\infty} \lambda^i \mathbf{A}_{PG}^i)$. Then Equation 11.5 can be rewritten as

$$\begin{aligned} (\mathbf{S} - \mathbf{A}_{PG}\lambda\mathbf{S}) &= \mathbf{I} \\ \mathbf{S}(\mathbf{I} - \mathbf{A}_{PG}\lambda) &= \mathbf{I} \\ \mathbf{S} &= (\mathbf{I} - \mathbf{A}_{PG}\lambda)^{-1} \end{aligned} \tag{11.6}$$

Since matrix inversion is cubic in runtime, the kernel can be computed in $\mathcal{O}(n^3)$, where n is the number of nodes in the product graph.

11.8 GRAPH KERNEL EXTENSIONS

The kernel as described so far disregards the position of hyponym and hypernym candidates in the product graph. However, nodes directly connected to hyponym and hypernym candidates or nodes that are located nearby a direct path from the hyponym to the hypernym candidate are usually more important for validating a hyponymy hypothesis. Nodes far away from both nodes could belong to an embedded subclause or a second main clause that is not semantically related to the hyponym and hypernym candidates. Therefore, instead of considering all common walks, two special graph kernels are used, which are extensions of the original approach of Gärtner et al. The first graph kernel counts weighted common walks that pass both hyponym and

[4]Note that in our evaluation, the index i was started at one instead of zero in order to exploit the entire value range for the normalized kernel value from zero to one.

hypernym candidates. The second graph kernel counts the weighted common walks that pass at least one of the hyponym and hypernym candidates. Both kernels cannot be directly calculated by this approach but obtained by subtracting different numbers of common walks from each other. The weighted number of common walks that pass at least one of the hyponym or hypernym candidate can be obtained by subtracting the number of common walks that pass neither hyponym and hypernym candidate from the number of all common walks.

$$K_{a_1 \vee a_2} = K - K_{\neg a1 \wedge \neg a2} \tag{11.7}$$

The weighted number of common walks that do not pass neither $a1$ nor $a2$ is fairly easy to determine. It can be done by just cancelling out (which means setting to zero) all entries of the adjacency matrix in the row and column belonging to either $a1$ or $a2$ and calculating the ordinary common walk kernel afterwards.

Similarly, the weighted number of common walks that pass both hypernym and hyponym candidates can be obtained by subtracting the number of common walks that do not pass the hypernym candidate node and the number of common walks that do not pass the hyponym candidate node from the number of all common walks. Since the number of common walks that pass neither the hyponym nor the hypernym candidate node are subtracted twice, they have to be added one time to the difference.

$$K_{a_1 \wedge a_2} = K - K_{\neg a1} - K_{\neg a2} + K_{\neg a1 \wedge \neg a2} \tag{11.8}$$

One possibility to calculate the common walk kernel is to apply Formula 11.6. An alternative, which we discuss in more detail in the remainder of this paper, is to use an approximation

$$\sum_{i=0}^{\infty} \iota^T (\lambda^i \mathbf{A}_{PG}^i)\iota \approx \sum_{i=0}^{k} \iota^T (\lambda^i \mathbf{A}_{PG}^i)\iota \tag{11.9}$$

for a sufficient high k. The advantage of this approach is that matrix multiplications are often hardware accelerated due to their importance for 3D-computer games. Although the runtime is cubic, the performance can be much faster that one would expect. A quite simple optimization is to reuse the $i-1$th entry of the sum for the calculation of the ith entry

$$\mathbf{A}_{PG}^{i+1}\lambda^i = \mathbf{A}_{PG}\lambda\mathbf{A}_{PG}^i\lambda^i \tag{11.10}$$

A second optimization is to exploit the fact that the weighted number of all common walks K as well as the weighted number of all common walks that pass neither $a1$ nor $a2$ is used in both kernels $K_{a1\vee a2}$ and $K_{a1\wedge a2}$ so these two intermediate results have to be calculated only once.

Note that the discount factor λ might be different for the kernels $K_{a1\vee a2}$ and $K_{a1\wedge a2}$. In this case, a direct use of those two intermediate results is not possible but instead some reformulations are required. Consider the case that K for a given λ is known and should be determined for λ'.

$$\begin{aligned}
&\iota^T \sum_{i=0}^{k} \left(\lambda'^i \mathbf{A}_{PG}^i\right)\iota = \\
&\iota^T \sum_{i=0}^{k} \left(\lambda^i \left(\frac{\lambda'}{\lambda}\right)^i \mathbf{A}_{PG}^i\right)\iota = \\
&\sum_{i=0}^{k} \left(\iota^T \lambda^i \left(\frac{\lambda'}{\lambda}\right)^i \mathbf{A}_{PG}^i \iota\right) = \\
&\left(\left(\frac{\lambda'}{\lambda}\right)^0, \ldots, \left(\frac{\lambda'}{\lambda}\right)^k\right) \cdot \left(\lambda^0 \mathbf{A}_{PG}^0, \ldots, \lambda^k \mathbf{A}_{PG}^k\right) = \\
&\mathbf{u} \cdot \mathbf{v}
\end{aligned} \tag{11.11}$$

Thus, the kernel function for a different value of λs can be determined by a simple scalar product. Finally, the two kernels are normalized with [22, p. 413]

$$\begin{aligned}
K_{a1\wedge a2,\text{norm}}(G_1, G_2) &= K_{a1\wedge a2}(G_1, G_2)/\sqrt{K_{a1\wedge a2}(G_1, G_1)K_{a1\wedge a2}(G_2, G_2)} \\
K_{a1\vee a2,\text{norm}}(G_1, G_2) &= K_{a1\vee a2}(G_1, G_2)/\sqrt{K_{a1\vee a2}(G_1, G_1)K_{a1\vee a2}(G_2, G_2)}
\end{aligned} \tag{11.12}$$

The graph kernel is used to compare the SNs the hypotheses were extracted from. If a hypothesis was extracted from several semantic networks then the maximum kernel values of all combinations is used but considering at most two semantic networks per hypothesis. Note that the maximum value function does not generally lead to positive-semidefinite kernel matrices, which means that the solution found by the SVM may only be a local optimum [31,32].

11.9 DISTANCE WEIGHTING

The kernels introduced so far have the advantage that the hypernym and hyponym hypotheses are reflected by the calculation of the graph kernel. A further possible improvement is to assign weights to the product graph nodes depending on the distance of the associated nodes to the hyponym and hypernym candidates. Such a weighting is suggested by the fact that edges located nearby the hypernym and hyponym candidates are expected to be more important for estimating the correctness of the hypothesis. It will be shown that a distance calculation is possible with minimum overhead just by using the intermediate results of the kernel computations. Let us define the matrix function $B : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ the function that sets an entry in a matrix to one, if the associated component of the argument matrix $\mathbf{M} = (m_{xy})$ is greater than zero and to zero otherwise. Let $(h_{xy}) = B(\mathbf{M})$, then

$$h_{xy} = \begin{cases} 1, \; m_{xy} > 0 \\ 0, \; \text{otherwise} \end{cases} \tag{11.13}$$

Let $\mathbf{A}_{a1 \vee a2}$ be the adjacency matrix where all entries are set to zero, if either the column or row of the entry is associated to $a1$ or $a2$. Then a matrix entry of $B(\mathbf{A}_{a1\vee a2}\mathbf{A}_{PG})$ is one, if there exists a common walk of length two between the two nodes and one of them is $a1$ and $a2$. Let us define $\mathbf{C}^i$ as $B(\mathbf{A}_{a1\vee a2}\mathbf{A}_{PG}^{i-1})$. An entry of $\mathbf{C}^i$ is one if there exists a common walk of length i between two nodes and one of them is $a1$ or $a2$. Then the distance of node u with matrix index $v := index(u)$ from node $a1$ or $a2$ is $i \; :\Leftrightarrow \exists e : f_{e,v} = 1$ with $(f_{x,y}) = (\mathbf{C}^i - \sum_{j=1}^{i-1} \mathbf{C}^j)$. The matrix differences need not to be calculated if the distances are determined iteratively starting at $i = 0$. Let $\text{dist}_i(u) : V \to \mathbb{R}$ be the distance of node u from $a1$ or $a2$ after the ith step. $\text{dist}_i(u)$ is a partial function, which means there are eventually nodes that are not yet assigned a distance value. Distances that are once assigned are immutable, that is, they do not change in succeeding steps, that is, $\text{dist}_i(u) = a \Rightarrow \text{dist}_{i+1}(u) = a$. $\text{dist}_i(u)$ for $i \geq 1$ is defined as

$$\begin{aligned} &\text{dist}_i(u) = a \Leftrightarrow \exists e : c_{e,\text{index}(u)} = 1 \wedge \\ &(\text{dist}_{i-1}(u) = a \vee \text{dist}_{i-1}(u) = \text{undef}) \wedge \\ &(c_{xy}) = \mathbf{C}^i \end{aligned} \tag{11.14}$$

Furthermore, $\text{dist}_0(u)$ is defined as zero if u is a node containing the hypernym or hyponym candidates. $\mathbf{C}^i$ can easily be obtained from $\mathbf{A}_{PG}^j$, $j = 0, \ldots, i-1$, which are intermediate results of the kernel computations. But the calculation can still be further simplified. $B(\mathbf{A}_{a1\vee a2}\mathbf{A}_{PG}^{i-1})$ can also be written as $\mathbf{A}_{a1\vee a2} \wedge B(\mathbf{A}_{PG}^{i-1})$ ($\mathbf{M_1} \wedge \mathbf{M_2}$ is build analogously to the matrix product, where the multiplication of matrix elements is replaced by the conjunction and the addition by the disjunction). The equivalence is stated in the following theorem.

Theorem 11.1 *If* $\mathbf{A}$ *and* $\mathbf{C}$ *are matrices with non-negative entries, then* $B(\mathbf{AC}) = B(\mathbf{A}) \wedge B(\mathbf{C})$.

Proof: Let b_{xy} be the matrix entries of $B(\mathbf{AC})$, $(a_{xy}) = \mathbf{A}$ and $(c_{xy}) = \mathbf{C}$. A matrix entry b_{xy} of $B(\mathbf{AC})$ is of the form $B(\sum_{s=1}^{m}(a_{xs}c_{sy}))$. Assume

$$\begin{aligned} b_{xy} = 1 &\Leftrightarrow \\ \exists j : (a_{xj}c_{jy} > 0) &\Leftrightarrow \\ \exists j : (a_{xj} > 0 \wedge c_{jy} > 0) &\Leftrightarrow \\ \exists j : (B(a_{xj}) \wedge B(c_{jy}) = 1) &\Leftrightarrow \\ \bigvee_{s=1}^{m} (B(a_{xs}) \wedge B(c_{sy})) = 1 & \end{aligned} \tag{11.15}$$

Let us now assume

$$\begin{aligned} b_{xy} = 0 \Leftrightarrow \\ \forall j : (a_{xj} c_{jy} = 0) \Leftrightarrow \\ \forall j : (a_{xj} = 0 \vee c_{jy} = 0) \Leftrightarrow \\ \forall j : (B(a_{xj}) \wedge B(c_{jy}) = 0) \Leftrightarrow \\ \bigvee_{s=1}^{m} (B(a_{xs}) \wedge B(c_{sy})) = 0 \\ q.e.d \end{aligned} \tag{11.16}$$

■

$\mathbf{A}_{a1 \vee a2}$ is a sparse matrix with nonzero entries only in the rows and columns of the hyponym and hypernym candidate indices. The matrix $\mathbf{A}_{a1 \vee a2}$ can be split up into two components $\mathbf{A}_{\text{row},\, a1 \vee a2}$ where only the entries in rows index($a1$) and index($a2$) are nonzero and into the matrix $\mathbf{A}_{\text{col},\, a1 \vee a2}$ where only the entries in columns index($a1$) and index($a2$) are nonzero.

The matrix conjunction $\mathbf{A}_{a1 \vee a2} \wedge B(\mathbf{A}_{PG}^{i-1})$ can then be written as

$$\begin{aligned} &\mathbf{A}_{a1 \vee a2} \wedge B(\mathbf{A}_{PG}^{i-1}) = \\ &(\mathbf{A}_{\text{row},\, a1 \vee a2} \vee \mathbf{A}_{\text{col},\, a1 \vee a2}) \wedge B(\mathbf{A}_{PG}^{i-1}) = \\ &\mathbf{A}_{\text{row},\, a1 \vee a2} \wedge B(\mathbf{A}_{PG}^{i-1}) \vee \mathbf{A}_{\text{col},\, a1 \vee a2} \wedge B(\mathbf{A}_{PG}^{i-1}) \end{aligned} \tag{11.17}$$

Similarly, $\mathbf{A}_{\text{row},\, a1 \vee a2}$ can be further splitted up into

$$\begin{aligned} \mathbf{A}_{\text{row(hypo)},\, a1 \vee a2} \text{ and} \\ \mathbf{A}_{\text{row(hyper)},\, a1 \vee a2} \end{aligned} \tag{11.18}$$

This conjunction contains the nonzero entries for the hyponym and hypernym row (analogously for the columns). The conjunction
$\mathbf{A}_{\text{hypo/hyper(row)},\, \mathbf{a1} \vee \mathbf{a2}} \wedge B(\mathbf{A}_{PG}^{i-1})$ is given in Figure 11.4, the conjunction
$\mathbf{A}_{\text{hypo/hyper(col)},\, a1 \vee a2} \wedge B(\mathbf{A}_{PG}^{i-1})$ in Figure 11.5. The first factor matrix as well as the result of the matrix conjunction are sparse matrices where only the nonzero entries are

$$\begin{array}{c} \\ d \end{array} \begin{array}{c} \begin{array}{cc} p & q \end{array} \\ \begin{pmatrix} & & \\ & 1 \quad 1 & \\ & & \end{pmatrix} \end{array} \wedge \begin{array}{c} B(\mathbf{A}_{PG}^{i-1}) \\ \begin{pmatrix} B(\mathbf{A}_1^{i-1}) \\ \\ \cdots \\ \\ B(\mathbf{A}_n^{i-1}) \end{pmatrix} \end{array} = \begin{pmatrix} \\ B(\mathbf{A}_p^{i-1}) \vee B(\mathbf{A}_q^{i-1}) \\ \\ \end{pmatrix}$$

FIGURE 11.4 Matrix conjunction of $\mathbf{A}_{\text{hyper/hypo(row)} a1 \vee a2}$ and $B(\mathbf{A}_{PG}^{i-1})$where $\mathbf{A}_{\text{hyper/hypo(row)} a1 \vee a2}$ is a sparse matrix with nonzero entries only in one row (called: d) and in the two columns p and q.

$$\begin{array}{c} \\ g \\ \\ h \\ \\ \end{array} \begin{array}{c} p \\ \left(\begin{array}{c} 1 \\ \\ 1 \end{array}\right) \\ \end{array} \wedge \begin{array}{c} B(\mathbf{A}_{PG}^{i-1}) \\ \left(\begin{array}{c} B(\mathbf{A}_1^{i-1}) \\ \cdots \\ B(\mathbf{A}_n^{i-1}) \end{array}\right) \end{array} = \left(\begin{array}{c} B(\mathbf{A}_p^{i-1}) \\ \\ B(\mathbf{A}_p^{i-1}) \end{array}\right)$$

FIGURE 11.5 Matrix conjunction of $\mathbf{A}_{\text{hyper/hypo(col)}a1\vee a2}$ and $B(\mathbf{A}_{PG}^{i-1})$ where $\mathbf{A}_{\text{hyper/hypo(col)}a1\vee a2}$ is a sparse matrix with nonzero entries only in one column (called: p) and in the two rows g and h.

given in the figures. $\mathbf{A}_r$ denotes the rth row vector of matrix $\mathbf{A}$. Usually the hyponym and hypernym nodes are only directly connected to one or two other nodes. Therefore, only a constant number of rows has to be checked for nonzero entries in each step. Thus, with the given intermediate results of $\mathbf{A}_{PG}^{j}$ $(j = 1, \ldots, i-1)$ with different exponents j, the distance computation can be done in time $\mathcal{O}(n)$.

After the distances are calculated, an $n \times n$ (n: number of nodes in the product graph) weight matrix $\mathbf{W}$ is constructed in the following way:

$$\begin{aligned} \mathbf{W} &:= (w_{xy}) \text{ and} \\ w_{xy} &:= g_w(a) \text{ with} \\ g_w^*(a) &:= \begin{cases} 1.0 & a \leq c \\ \cos(b(a-c)) & 0 < b(a-c) \leq \pi/2 \\ 0.0 & b(a-c) > \pi/2 \end{cases} \qquad (11.19) \\ g_w(a) &:= \max\{0.1, g_w^*(a)\} \text{ and} \\ a &:= \min\{\text{dist}_k(x), \text{dist}_k(y)\} \end{aligned}$$

The cosine function is used to realize a smooth transition from 1.0 to 0.1. b, c are fixed positive constants, which can be determined by a parameter optimization (for instance by a grid search). They are currently manually set to $b = \pi/5$, $c = 2$. The application of the weight matrix is done by a component-wise (*) multiplication. Let $\mathbf{M_1}$ and $\mathbf{M_2}$ be two $m \times n$ matrices. Then the component-wise product $\mathbf{P} = \mathbf{M_1} * \mathbf{M_2}$ with $\mathbf{P} = (p_{xy})$, $\mathbf{M_1} = (m_{1,xy})$, $\mathbf{M_2} = (m_{2,xy})$ is defined as $p_{xy} = m_{1,xy} \cdot m_{2,xy}$ for $1 \leq x \leq m, 1 \leq y \leq n$.

An alternative method to derive the weighted matrix, directly determines weights for the edges instead for the nodes. The matrix of distance values for every edge is given by $\mathbf{D} = (d_{xy})$ with $d_{xy} = i - 1 :\Leftrightarrow f_{xy} = 1$ and $(f_{xy}) = B(\mathbf{C}^i - \sum_{j=1}^{i-1} \mathbf{C}^j)$. The weight matrix $\mathbf{W}$=(w_{xy}) is then given by $w_{xy} = g_w(d_{xy})$. This method also benefits from the proposed matrix decomposition, which speeds up the matrix multiplication enormously. We opted for the first possibility of explicitly deriving distance values for all graph nodes, which allows a more compact representation.

The component-wise multiplication can be done in $\mathcal{O}(n^2)$ and is therefore much faster than the ordinary matrix multiplication, which is cubic in runtime. The component-wise multiplication follows the distributive and the commutative laws, which can easily be seen. Thus,

$$\iota^T \mathbf{W} * \left(\sum_{i=0}^{k} (\lambda^i \mathbf{A}^i_{PG}) \right) \iota = \iota^T \left(\sum_{i=0}^{k} (\lambda^i \mathbf{W} * \mathbf{A}^i_{PG}) \right) \iota. \tag{11.20}$$

Therefore, the Formula 11.11 changes to

$$\begin{aligned}
&\iota^T \mathbf{W} * \sum_{i=0}^{k} (\lambda'^i \mathbf{A}^i_{PG}) \iota = \\
&\iota^T \sum_{i=0}^{k} \left(\lambda^i \left(\frac{\lambda'}{\lambda} \right)^i \mathbf{W} * \mathbf{A}^i_{PG} \right) \iota = \\
&\sum_{i=0}^{k} \left(\left(\frac{\lambda'}{\lambda} \right)^i \iota^T \lambda^i \mathbf{W} * \mathbf{A}^i_{PG} \iota \right) = \\
&\left(\left(\frac{\lambda'}{\lambda} \right)^0, \ldots, \left(\frac{\lambda'}{\lambda} \right)^k \right) \cdot (\lambda^0 \mathbf{W} * \mathbf{A}^0_{PG}, \ldots, \lambda^k \mathbf{W} * \mathbf{A}^k_{PG}) = \\
&\mathbf{u} \cdot \mathbf{v}(\mathbf{W})
\end{aligned} \tag{11.21}$$

This shows that the transformation method given in Formula 11.11 where the sum is converted from one value of λ to another is still possible.

11.10 FEATURES FOR HYPONYMY EXTRACTION

Beside the graph kernel approach we also estimated the hypernymy hypothesis correctness by a set of features. The following feature are used:

- *Pattern Application*. A set of binary features. A pattern feature is set to one, if the hypothesis was extracted by this pattern, to zero otherwise.
- *Correctness*. In many cases the correctness can be estimated by looking on the hyponymy and hypernymy candidate alone. An automatic approach was devised that calculates a correctness estimation based on this assumption [25].
- *Lexicon*. The lexicon features determines a score based on the fact that if both hypernym and hyponym candidates (or the concepts associated to their base words) were contained in the lexicon or only one of them. This procedure is based on the fact that a lexicon-based hyponymy hypotheses validation is only fully possible, if both concepts are contained in the deep lexicon.
- *Context*. The context features investigates if both hyponym and hypernym candidates are connected in the semantic network to similar properties.
- *Deep/Shallow*. This binary feature is set to one if a hypotheses is only extracted by either deep or shallow extraction rules (0) or by both together (1).

11.11 EVALUATION

We applied the patterns on the German Wikipedia corpus from November 2006, which contains about 500,000 articles. In total, we extracted 391,153 different hyponymy relations employing 22 deep and 19 shallow patterns. The deep patterns were matched to the SN representation, the shallow patterns to the tokens. Concept pairs that were also recognized by the compound analysis were excluded from the results, since such pairs can be recognized on the fly and need not be stored in the knowledge base. Thus, these concept pairs are disregarded for the evaluation. Otherwise, recall and precision would increase considerably. 149,900 of the extracted relations were only determined by the deep but not by the shallow patterns. If relations extracted by one rather unreliable pattern are disregarded, this number is reduced to 100,342. The other way around, 217,548 of the relations were determined by the shallow but not by the deep patterns. 23,705 of the relations were recognized by both deep and shallow patterns. Naturally, only a small fraction of the relations were checked for correctness. In total, 6932 relations originating from the application of shallow patterns were annotated, 4727 were specified as correct. In contrast, 5626 relations originating from the application of deep patterns were annotated and 2705 were specified as correct.

We evaluated our hyponymy extraction approach called SemQuire (SemQuire for *acquiring knowledge semantic-based*) on a set of selected 1500 hypotheses, which were annotated by test persons. The annotation was either $\{+1\}$ for hyponymy relation actually present and $\{-1\}$ for hyponymy relation not present. We conducted on this set a 10-fold cross-validation. Positive and negative examples were chosen equally, which avoids that the evaluation values fluctuate depending on the used patterns. Two methods were evaluated, the feature-based validation and the validation where a graph kernel is used in addition. The confusion matrix is given in Table 11.2, the evaluation measures in Table 11.3. The evaluated measures are accuracy (relative frequency with which a decision for hypothesis correctness or noncorrectness is correct), precision (relative frequency with which a predicted hyponym is indeed one), and recall (relative frequency with which correct hyponym hypotheses were predicted as correct). Note that the given recall is the recall of the validation and not of the hypothesis extraction component. The use of the graph kernel leads to an increase in

TABLE 11.2 Confusion Matrix for the Validation of Hypernyms

	Cimiano		GK−		GK+		Σ
	PNH	PH	PNH	PH	PNH	PH	
NH	585	165	637	113	611	139	750
H	473	277	187	563	149	601	750
Σ	1058	442	824	676	760	740	

Cimiano: context-based method from Cimiano et al., GK−=without graph kernel (only feature kernel), GK+=with graph and feature kernel. NH: no hyponym, H: hyponym, PNH: predicted nonhyponym, PH: predicted hyponym.

TABLE 11.3 Accuracy, *F*-Measure, Precision, and Recall for the Validation of Hyponyms for the GermaNet Classifier, the Context-Based Method of Cimiano et al. and SemQuire

Measure	GermaNet	Cimiano	GK−	GK+
Accuracy	0.52	0.57	0.80	0.81
F-measure	0.07	0.46	0.79	0.81
Precision	1.00	0.63	0.83	0.81
Recall	0.04	0.37	0.75	0.80

TABLE 11.4 Confusion Matrix for the Validation of Hypernyms for the Unweighted and the Weighted Graph Kernel

	Weighted−		Weighted+		Σ
	PNH	PH	PNH	PH	
NH	348	152	349	151	500
H	191	309	190	310	500
Σ	539	461	539	461	

TABLE 11.5 Accuracy, *F*-Measure, Precision, and Recall for the Validation of Hyponyms for the Unweighted and the Weighted Graph Kernel

Measure	Weighted−	Weighted+
Accuracy	0.657	0.659
F-measure	0.643	0.645
Precision	0.670	0.672
Recall	0.618	0.620

accuracy, *F*-measure and recall where the increase of the recall is significant with a level of 5%. Furthermore, we compared our system SemQuire with a GermaNet classifier[5] that opts for hypernymy if the hypernymy relation can be looked up in the GermaNet knowledge base and/or can be inferred by the use of synonymy and/or the transitive closure of the hypernymy relation. Furthermore, we reimplemented the context-based hypernymy validation method as proposed by Cimiano et al. [33]. Both methods were clearly outperformed by SemQuire (significance level: 1%).

Furthermore, a preliminary evaluation of the weighting method was done on 1000 instances (see confusion matrix in Table 11.4 and evaluation measures in Table 11.5). At first one might think that the evaluation can be done in such a way that the non-weighted graph kernel is replaced by the weighted graph kernel and the original *F*-measure is compared with that one obtained with the weighted graph kernel. But

[5]GermaNet synsets were mapped to HaGenLex concepts.

since the weights are always less than one, the total weight of the graph kernel in comparison with the feature-based kernel would be reduced. Thus, in this experiment the feature kernel was not used at all. Note that for some instances an SN was not available, which degraded the F-measure of the graph kernel. The evaluation showed that the F-measure of the weighted graph kernel is only slightly better than that one of the unweighted graph kernel and the improvement is not significant. We plan to test with other discount factors as well as other decay functions than cosine. Also, a larger training corpus should be employed for reliable results.

11.12 CONCLUSION AND OUTLOOK

This paper described the automatic extraction of hyponyms from the Wikipedia corpus based on several deep and shallow patterns. The shallow patterns are designed on the basis of tokens and the deep patterns as semantic networks. Both types of patterns were applied to the German Wikipedia. The extracted hypotheses were afterwards validated with a support vector machine and a graph kernel. The use of a graph kernel leads to an improvement in F-measure, accuracy, and recall where the increase in recall is significant. A preliminary evaluation was done for the weighted graph kernel where only a very slight (but not significant) improvement was reached. Furthermore, we compared our method SemQuire to a GermaNet classifier and to the context feature of Cimiano where both of them were clearly outperformed. We plan to optimize the weights of the individual kernels (feature and graph kernel) by a grid search, which is expected to further improve the results.

Currently the weighting is done only on the basis of the distance measured in number of edges. Other factors such as the edge labels are not taken into account. So future work could be to develop a more sophisticated weighting scheme.

ACKNOWLEDGMENTS

We thank all members of the department of the Goethe University that contributed to this work, especially Prof. Alexander Mehler. Furthermore, I want to thank Vincent Esche, Dr. Ingo Glöckner, and Armin Hoehnen for proof-reading this document. Also, I am indebted to Prof. Hermann Helbig and Dr. Sven Hartrumpf for letting me use the WOCADI parse of the Wikipedia.

REFERENCES

1. L. Getoor, B. Taskar, Introduction, in *Introduction to Statistical Relational Learning*, (L. Getoor, B. Taskar, eds.), MIT Press, Cambridge, Massachusetts, pp. 1–8, 2007.
2. Z. Harchaoui, F. Bach, Image classification with segmentation graph kernels, in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, Minnesota, 2007.

3. R. Grishman, S. Zhao, Extracting relations with integrated information using kernel methods, in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, Ann Arbor, Michigan, pp. 419–426, 2005.
4. R.C. Bunescu, R.J. Mooney, A shortest path dependency kernel for relation extraction, in *Proceedings of the Human Language Technology Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, Vancouver, Canada, pp. 724–731, 2005.
5. F. Reichartz, H. Korte, G. Paass, Dependency tree kernels for relation extraction from natural language text, in *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, Bled, Slovenia, pp. 270–285, 2009.
6. S. Ghosh, P. Mitra, Combining content and structure similarity for xml document classification using composite svm kernels, in *19th International Conference on Pattern Recognition (ICPR)*, Tampa, Florida, pp. 1–4, 2008.
7. A. Moschitti, R. Basili, A tree kernel approach to question and answer classification in question answering systems, in *In Proceedings of the Conference on Language Resources and Evaluation (LREC)*, Genova, Italy, 2006.
8. P. Mahé, J.-P. Vert, Graph kernels based on tree patterns for molecules. *Mach. Learn.* **75**(1), 3–35 (2008).
9. J. Lafferty, A. McCallum, F. Pereira, Conditional random fields: probabilistic models for segmenting and labeling sequence data, in *Proceedings of the International Conference on Machine Learning (ICML)*, Pittsburgh, Pennsylvania, pp. 282–289, 2001.
10. C. Sutton, A. McCallum, An introduction to conditional random fields for relational learning, in *Statistical Relational Learning*, (L. Getoor, B. Taskar, eds.), MIT Press, Cambridge, Massachusetts, USA, pp. 93–127, 2007.
11. X. He, R.S. Zemel, M.Á. Carreira-Perpi nán, Multiscale conditional random fields for image labeling, in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)*, Washington, D.C., Vol. 2, pp. 695–702, 2004.
12. D.J. Cook, L.B. Holder, Substructure discovery using minimum description length and background knowledge. *J. Artif. Intell. Res.* **1**, 231–255 (1994).
13. J. Rissanen, *Stochastic Complexity in Statistical Inquiry*, World Scientific Publishing Company, Hackensack, New Jersey, 1989.
14. R.N. Chittimoori, L.B. Holder, D.J. Cook, Applying the subdue substructure discovery system to the chemical toxicity domain, in *Proceedings of the 12th International Florida AI Research Society Conference (FLAIRS)*, Orlando, Florida, pp. 90–94, 1999.
15. R. Snow, D. Jurafsky, A.Y. Ng, Learning syntactic patterns for automatic hypernym discovery, in *Advances in Neural Information Processing Systems 17*, MIT Press, Cambridge, Massachusetts, pp. 1297–1304, 2005.
16. V. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, New York, New York, 1998.
17. T.T. Quan, S.C. Hui, A.C.M. Fong, T.H. Cao, Automatic generation of ontology for scholarly semantic web, in *The Semantic Web: ISWC 2004*, Springer, Heidelberg, Germany, Vol. 4061 *LNCS*, pp. 726–740, 2004.
18. M. Hearst, Automatic acquisition of hyponyms from large text corpora, in *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, Nantes, France, pp. 539–545, 1992.

19. A. Culotta, J. Sorensen, Dependency tree kernels for relation extraction, in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, Barcelona, Spain, pp. 423–429, 2004.
20. V. Kashyap, C. Ramakrishnan, C. Thomas, A.P. Sheth, Texaminer: an experimentation framework for automated taxonomy bootstrapping. *Int. J. Web Grid Serv.* **1**(2) (2005).
21. H. Helbig, *Knowledge Representation and the Semantics of Natural Language*, Springer, Heidelberg, Germany, 2006.
22. B. Schölkopf, A.J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*, MIT Press, Cambridge, Massachusetts, 2002.
23. S. Hartrumpf, *Hybrid Disambiguation in Natural Language Analysis*. PhD thesis, FernUniversität in Hagen, Fachbereich Informatik, Hagen, Germany, 2002.
24. S. Hartrumpf, H. Helbig, R. Osswald, The semantically based computer lexicon HaGenLex: structure and technological environment. *Traitement Automatique des Langues* **44**(2), 81–105 (2003).
25. T. vor der Brück, Hypernymy extraction using a semantic network representation. *Int. J. Comput. Linguist. Appl.* **1**(1), 105–119 (2010).
26. T. vor der Brück, Learning semantic network patterns for hypernymy extraction, in *Proceedings of the 6th Workshop on Ontologies and Lexical Resources (OntoLex)*, Beijing, China, pp. 38–47, 2010.
27. K.M. Borgwardt, H.-P. Kriegel, Shortest-path kernels on graphs, in *International Conference on Data Mining*, Houston, Texas, pp. 74–81, 2005.
28. H. Kashima, K. Tsuda, A. Inokuchi, Marginalized kernels between labeled graphs, in *Proceedings of the International Conference on Machine Learning (ICML)*, Washington, D.C., pp. 321–328, 2003.
29. S. Hido, H. Kashima, A linear-time graph kernel, in *9th IEEE International Conference on Data Mining*, Miami, Floria, 2009.
30. R. Diestel, *Graph Theory*, Springer, Heidelberg, Germany, 2010.
31. B. Haasdonk, Feature space interpretation of SVM with indefinite kernels. *IEEE Trans. Pattern Anal. Mach. Intell.* **27**(4), 482–492 (2005).
32. F. Fleuret, S. Boughorbel, J.-P. Tarel, Non-mercer kernel for SVM object recognition, in *Proceedings of the British Machine Vision Conference (BMVS)*, London, UK, pp. 137–146, 2004.
33. P. Cimiano, A. Pivk, L. Schmidt-Thieme, S. Staab, Learning taxonomic relations from heterogeneous sources of evidence, in *Ontology Learning from Text: Methods, Evaluation and Applications*, (P. Buitelaar, P. Cimiano, B. Magnini, eds.), IOS Press, Amsterdam, The Netherlands, pp. 59–73, 2005.

INDEX

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