Birkhäuser

Matthias Dehmer
Editor

Structural Analysis of Complex Networks

Birkhäuser

Editor
Ao. Prof. Dr. habil. Matthias Dehmer
Institute for Bioinformatics and Translational Research
The Health and Life Sciences University
UMIT-Private Universität für Gesundheitswissenschaften
Eduard Wallnöfer-Zentrum 1
A-6060 Hall in Tirol, Austria

and

Institute of Discrete Mathematics and Geometry
Vienna University of Technology
Wiedner Hauptstrasse 8-10
A-1040 Vienna, Austria

ISBN 978-0-8176-4788-9 e-ISBN 978-0-8176-4789-6
DOI 10.1007/978-0-8176-4789-6
Springer Dordrecht Heidelberg London New York

Library of Congress Control Number: 2010938359

Mathematics Subject Classification: 05C05, 05C12, 05C75, 05C80, 05C85, 05D40, 68R10, 90B10, 92E10, 94C15

© Springer Science+Business Media, LLC 2011
All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.
The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed on acid-free paper

www.birkhauser-science.com

Preface

Because of the increasing complexity and growth of real-world networks, their analysis by using classical graph-theoretic methods is oftentimes a difficult procedure. Thus, there is a strong need to combine graph-theoretic methods with mathematical techniques from other scientific disciplines, such as machine learning, statistics, and information theory, for analyzing complex networks more adequately.

The book *Structural Analysis of Complex Networks* presents theoretical as well as practice-oriented results for structurally exploring complex networks. Hence, the book does not only focus on classical graph-theoretical methods, it also shows the usefulness and potential of structural graph theory as a tool for solving interdisciplinary problems. Special emphasis is given to methods and areas which can be roughly summarized as follows:

- Graph-theoretical applications in, e.g., structural biology, computational biology, mathematical chemistry, and computational linguistics;
- Graph classes;
- General structural properties of networks;
- Graph colorings;
- Graph polynomials;
- Information measures for graphs, e.g., graph entropies;
- Metrical properties of graphs;
- Partitions and decompositions;
- Quantitative graph measures.

This book is intended for an interdisciplinary audience, covering topics from artificial intelligence, computer science, computational and systems biology, cognitive science, computational linguistics, discrete mathematics, machine learning, mathematical chemistry, and statistics, and it contains nineteen chapters that have been peer-reviewed according to the standards of international journals in applied mathematics. The chapters and some of their interrelations can be briefly described as follows.

Emmert-Streib starts the volume by surveying basic structural properties of complex networks, important graph classes, and graph measures used when performing network analysis quantitatively. The latter relates to determining the structural similarity between graphs and their structural complexity using entropic measures.

Further concepts used to explore networks structurally are provided by the next chapters authored by *Borowiecki*, *Goddard* et al., and *Ananchuen* et al. In particular, these chapters present techniques of graph partitioning, distances in graphs, and domination in graphs, respectively. The chapter written by *Fujii* also discusses entropy measures, but for infinite directed graphs. However, these measures are obtained by using operator theory and, hence, are differently defined than the ones presented in the chapter by Emmert-Streib; those are derived based on Shannon's entropy and can be interpreted as the structural information content of a graph. Then, the chapters authored by *Matsumoto*, *Kovář*, and *Brešar* et al. investigate multifaceted problems, like exploring infinite labeled graphs to study presentations of symbolic dynamical systems, special graph decompositions, and the examination of geodetic sets in graphs, which represents an important problem using metrical properties of graphs. *Ellis-Monaghan* et al. provide two chapters in this volume on graph polynomials: The first one emphasizes the Tutte polynomial and some closely related graph polynomials. The second chapter by *Ellis-Monaghan* et al. sheds light on interpretations of concrete polynomials and on interrelations between other graph polynomials and the Tutte polynomial. The problem of reconstructing graphs by examining specific properties of polynomials, here, the zeros of Krawtchouk polynomials, is tackled in the next chapter by *Stoll*. Quantitative methods to calculate the structural similarity or distance between two graphs have already been mentioned in Emmert-Streib's chapter. In particular, classical measures based on determining isomorphic subgraphs have already been mentioned there. The chapter written by *Lauri* treats the graph similarity problem in a similar manner, namely based on the number of common vertex-deleted subgraphs, and examines aspects of the computational complexity for calculating the mentioned graph similarity measure. The idea of defining structural distances between graphs is tackled in the next chapter, written by *Benadé*. More precisely, a chromatic metric is defined, and, remarkably, by applying this metric, the maximum distance between any two graphs is at most three.

The last six chapters of this volume use graph-theoretic techniques to solve challenging problems in, e.g., applied mathematics, computer science, quantum chemistry, electrical engineering, computational linguistics, structural biology and RNA structure analysis, computational biology, and mathematical chemistry. The chapter authored by *Cioabă* gives a broad overview on results for relating important structural properties of a graph to its eigenvalues. Also, *Cioabă* surveys important applications of graph spectra in subfields of the just-mentioned disciplines. To demonstrate the great potential of novel graph classes within computational linguistics, *Mehler* introduces a graph class consisting of hierarchical graphs called Minimum Spanning Markovian Trees and shows its usefulness by outlining concrete applications within semiotic network analysis. The chapter contributed by *Scripps* et al. starts by reviewing techniques to mine general complex networks, but mainly focuses on link-based classification, which often appears as an important problem in Web mining. The next two chapters, authored by *Washietl* et al. and *Mason* et al., explore graph-based problems in structural and computational biology, respectively. In particular, *Washietl* et al. investigate RNA structures represented by graphs and review graph-theoretical methods for describing and comparing such structures. A problem that is currently of considerable interest in biological network analysis is addressed by *Mason* et al. and deals with surveying methods for predicting protein function based on complex interaction networks. An area in which graph-theoretical models and techniques have been intensely applied so far is mathematical chemistry. The volume concludes by presenting a chapter about a graph class that is meaningful in mathematical chemistry: *Vukičević* presents techniques for determining the existence and enumeration of what are called perfect matchings that correspond to Kekulé structures, which are well known in mathematical chemistry.

Many colleagues, whether consciously or unconsciously, provided input, help, and support before and during the formation of this book. In particular, I would like to thank Hamid Arabnia, Alireza Ashrafi, Alexandru T. Balaban, Subhash Basak, Igor Bass, Agnieszka Bergel, David Bialy, Danail Bonchev, Stefan Borgert, Mieczysław Borowiecki, Monique Borusiak, Ulrike Brandt, Mathieu Dutour, Michael Drmota, Abdol-Hossein Esfahanian, Maria Fonoberova, Bernhard Gittenberger, Arno Homburg, Jürgen Kilian, Elena Konstantinova, Reinhard Kutzelnigg, Dmitrii Lozovanu, Alexander Mehler, Tomás Madaras, Abbe Mowshowitz, Marina Popovscaia, Fred Sobik, Stefan Shetschew, Doru Stefanescu, Thomas Stoll, Kurt Varmuza, Ilona Wesarg, Bohdan Zelinka, Dongxiao Zhu, and all authors and co-authors of this book. I apologize to any who inadvertently have not been named.

I am deeply grateful to Armin Graber from UMIT for his strong support and for providing such a stimulating working atmosphere. Many thanks to Isabella Fritz, Bernd Haas, Gerd Lorünser, Brigitte Senn-Kircher, and Klaus Weinberger for their help and fruitful discussions. Moreover, I thank Frank Emmert-Streib for the extremely fruitful collaboration and many stimulating discussions we had over several years. Frank also provided the figures used to design the front cover of this book.

In addition, I would like to thank editors Tom Grasso, Rebecca Biega, and Regina Gorenshteyn from Birkhäuser Publishing (Boston), who have always been available and helpful. Last but not least, I would like to thank my wife Jana and my family — Marion Dehmer-Sehn and Werner Dehmer — for their unfailing support and encouragement.

Finally, I hope that this book will help to extend the enthusiasm and joy that I feel for this field to others, and that it will inspire people to apply graph theory to different scientific areas for the solution of challenging and interdisciplinary problems.

Hall in Tirol, April 2010 Matthias Dehmer

Contents

Contributors

Nawarat Ananchuen Department of Mathematics, Faculty of Science, Silpakorn University, Nakorn Pathom 73000, Thailand, nawarat@su.ac.th

Watcharaphong Ananchuen School of Liberal Arts, Sukhothai Thammathirat Open University, Nonthaburi 11120, Thailand, laasawat@stou.ac.th

Gerhard Benadé School of Computer Science, Statistics and Mathematics, North-West University, Potchefstroom, South Africa, gerhard.benade@nwu.ac.za

Mieczysław Borowiecki Faculty of Mathematics, Computer Science and Econometrics, University of Zielona Góra, Podgórna 50, 65-246 Zielona Góra, Poland, M.Borowiecki@wmie.uz.zgora.pl

Boštjan Brešar Faculty of Natural Sciences and Mathematics, University of Maribor, Koroška 160, 2000 Maribor, Slovenia, bostjan.bresar@uni-mb.si

Sebastian M. Cioabă Department of Mathematical Sciences, University of Delaware, 501 Ewing Hall, Newark, DE 19716-2553, USA, cioaba@math.udel.edu

Peter Clifford Hamilton Institute, NUI Maynooth, Maynooth, Ireland, p@pclifford.net

Joanna A. Ellis-Monaghan Department of Mathematics, Saint Michael's College, One Winooski Park, Colchester, VT 05439, USA
and
Department of Mathematics and Statistics, University of Vermont, 16 Colchester Avenue, Burlington, VT 05405, USA, jellis-monaghan@smcvt.edu

Frank Emmert-Streib Computational Biology and Machine Learning, Center for Cancer Research and Cell Biology, School of Medicine, Dentistry and Biomedical Sciences, Queen's University Belfast, 97 Lisburn Road, Belfast BT9 7BL, UK, v@bio-complexity.com

Abdol-Hossein Esfahanian Computer Science and Engineering Department, 3115 Engineering Building, Michigan State University, East Lansing, MI 48824-1226, USA, esfahanian@cse.msu.edu

Jun Ichi Fujii Department of Arts and Sciences (Information Science), Osaka Kyoiku University, Asahigaoka, Kashiwara, Osaka 582-8582, Japan, fujii@cc.osaka-kyoiku.ac.jp

Tanja Gesell Center for Integrative Bioinformatics Vienna, Max F. Perutz Laboratories, Dr. Bohr-Gasse 9, 1030 Vienna, Austria
and
University of Vienna, Medical University of Vienna, and University of Veterinary Medicine, Vienna, Austria, tanja.gesell@univie.ac.at

Wayne Goddard School of Computing and Department of Mathematical Sciences, Clemson University, Clemson, SC 29634-1906, USA, goddard@clemson.edu

Petr Kovář Department of Applied Mathematics, Technical University Ostrava, 17. listopadu, 708 33 Ostrava–Poruba, Czech Republic, petr.kovar@vsb.cz

Matjaž Kovše Faculty of Natural Sciences and Mathematics, University of Maribor, Koroška 160 2000 Maribor, Slovenia matjaz.kovse@uni-mb.si

Josef Lauri Department of Mathematics, University of Malta, Tal-Qroqq, Malta, josef.lauri@um.edu.mt

Oliver Mason Hamilton Institute, NUI Maynooth, Maynooth, Ireland, oliver.mason@nuim.ie

Kengo Matsumoto Department of Mathematics, Joetsu University of Education, Joetsu 943-8512, Japan, kengo@juen.ac.jp

Alexander Mehler Goethe-University Frankfurt am Main, Senckenberganlage 31, 60325 Frankfurt am Main, Germany, Mehler@em.uni-frankfurt.de

Criel Merino Instituto de Matemáticas, Universidad Nacional Autónoma de México, Area de la Investigación Científica, Circuito Exterior, C.U., Coyoacán, 04510 México D.F., México, merino@matem.unam.mx

Ronald Nussbaum Computer Science and Engineering Department, 3115 Engineering Building, Michigan State University, East Lansing, MI 48824-1226, USA, ronald@cse.msu.edu

Ortrud R. Oellermann Department of Mathematics and Statistics, University of Winnipeg, Winnipeg, MB R3B 2E9, Canada o.oellermann@uwinnipeg.ca

Michael D. Plummer Department of Mathematics, Vanderbilt University, Nashville, TN 37240, USA, michael.d.plummer@vanderbilt.edu

Jerry Scripps School of Computing and Information Systems, 1 Campus Drive, Grand Valley State University, Allendale, MI 49401, USA, scrippsj@gvsu.edu

Thomas Stoll Faculty of Mathematics, School of Computer Science, University of Waterloo, Waterloo, ON, Canada, tstoll@cs.uwaterloo.ca

Pang-Ning Tan Computer Science and Engineering Department, 3115 Engineering Building, Michigan State University, East Lansing, MI 48824-1226, USA, ptan@cse.msu.edu

Aleksandra Tepeh University of Maribor, FEECS, Smetanova 17, 2000 Maribor, Slovenia, aleksandra.tepeh@uni-mb.si

Mark Verwoerd Hamilton Institute, NUI Maynooth, Maynooth, Ireland, mark.verwoerd@nuim.ie

Damir Vukičević Faculty of Mathematics and Natural Sciences, University of Split, Nikole Tesle 12, HR-21000 Split, Croatia, vukicevi@pmfst.hr

Stefan Washietl EMBL-European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, UK
and
Institute for Theoretical Chemistry, University of Vienna, Währingerstraße 17, A-1090 Vienna, Austria, washietl@ebi.ac.uk

Chapter 1
A Brief Introduction to Complex Networks and Their Analysis

Frank Emmert-Streib

Abstract In this chapter we present a brief introduction to complex networks and their analysis. We review important network classes and properties thereof as well as general analysis methods. The focus of this chapter is on the structural analysis of networks, however, information-theoretic methods are also discussed.

Keywords Complex networks · Centrality measures · Comparative network analysis · Module detection · Information-theoretic measures

MSC2000: Primary 05C90; Secondary 65C60, 46N60, 94A17

1.1 Introduction

Discrete objects representing graphs have been studied for a long time. Among the first who studied graphs are Euler [56] and Cayley [30]. Interestingly, the origin of the term *graph* dates back to König in the 1930s [81], less than 100 years ago. The interest in graphs and their analysis is manifold. From a theoretical point of view the categorization and the analysis of properties of graphs [20, 44, 54, 70] as well as the development of graph algorithms [37, 57] are important problems that have been studied extensively. From an applied point of view it has been realized that graphs can represent physical [70], biological [51, 78, 101], or sociological objects [68, 104], e.g., a crystal or protein structure or the acquaintance network among a group of people. Recently, networks have been also employed in data analysis and machine learning [3, 51, 99]. In the following we use the terms *graph* and *network* interchangeably although they do not mean precisely the same thing. Usually,

F. Emmert-Streib (✉)
Computational Biology and Machine Learning, Center for Cancer Research and Cell Biology, School of Medicine, Dentistry and Biomedical Sciences, Queen's University Belfast, 97 Lisburn Road, Belfast BT9 7BL, UK
e-mail: v@bio-complexity.com

a *graph* refers first of all to a mathematical object regardless of its realization in, e.g., nature, whereas a *network* represents a “real-world” object rather than a pure mathematical one. Because we want to focus on applied aspects of graphs most of the time we prefer the expression *network*.

In this chapter we provide a brief introduction to complex networks and their analysis. In Sect. 1.2 we review some important network classes. In Sect. 1.3 we present some methods for the structural analysis of networks that help, e.g., to characterize them as a whole or allow us to identify specific nodes in the network with certain properties. In Sect. 1.3.5 we present important methods to analyze networks comparatively. This means that means these measures always compare two graphs with each other and provide, hence, similarity or dissimilarity measures for this comparison. Such methods are especially useful for data analysis or machine learning because they allow a combination with, e.g., clustering methods to extract regularities from the obtained, e.g., similarity values for corpora of networks. Section 1.3.6 presents a method for the identification of community or module structure of networks as important, e.g., for the analysis of communication networks. In Sect. 1.4 we discuss information-theoretic measures and this chapter finishes in Sect. 1.5 with a short summary and conclusions.

1.2 Important Network Classes

We begin this chapter by reviewing well-known network classes. In the following we mainly restrict ourselves to undirected unweighted networks, however, most of the presented networks can be generalized easily.

1.2.1 Simple Networks

A simple network consists of *regular* connections among the nodes. One of the most prominent examples therefore is the two-dimensional lattice as shown in Fig. 1.1. Here each node is connected to its nearest neighbors. Despite its simplicity, such networks have been used extensively, e.g., in physics to study phenomena like ferromagnetism with the Ising model [32]. Other examples of this class are linear chains or nonrectangular lattices as used, e.g., in the context of protein structure prediction to model protein folding [74].

1.2.2 Random Networks

Random networks have been extensively studied by Erdös and Rényi [54, 55]. A random graph with N nodes is obtained by connecting every pair of nodes with

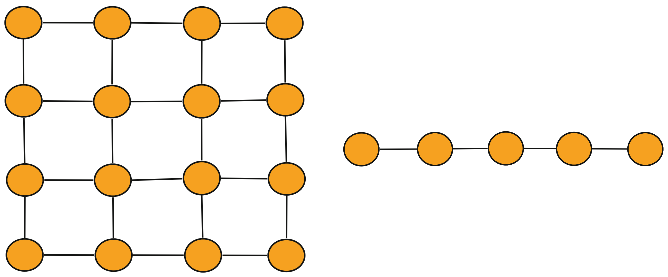

Fig. 1.1 *Left*: Regular two-dimensional lattice. *Right*: Linear, regular chain

probability p. The expected number of edges for a (undirected) network constructed this way is

$$E(N) = p\frac{N(N-1)}{2}. \tag{1.1}$$

1.2.3 Degree Distribution

The degree distribution of a node i in a random network is binomial

$$P(k_i = k) = \binom{N-1}{k} p^k (1-p)^{N-1-k}, \tag{1.2}$$

because the maximal degree of node i is $N-1$, the probability that the vertex has k links is $p^k(1-p)^{N-1-k}$ and there are $\binom{N-1}{k}$ possibilities to choose k links from $N-1$ nodes. In the limit $N \to \infty$ (1.2) becomes

$$P(k_i = k) = \frac{z^k \exp(-z)}{k!}. \tag{1.3}$$

Here $z = p(N-1)$ is the expected number of links for a node. This means that the degree distribution of a node in a random network can be approximated by the Poisson distribution for large N. For this reason random networks are also called Poisson random networks [96].

Furthermore one can show that the degree distribution of a random network (instead of just a node) also approximately follows a Poisson distribution

$$P(X_k = r) = \frac{z^r \exp(-z)}{r!}, \tag{1.4}$$

meaning that there are $X_k = r$ nodes in the network having degree k [2].

1.2.4 Clustering Coefficient

In general the clustering coefficient of a node i is defined as the fraction E_i of existing connections among its k_i nearest neighbors divided by the total number of possible connections,

$$C_i = \frac{2E_i}{k_i(k_i - 1)}. \tag{1.5}$$

This corresponds to the probability that two nearest neighbors of i are connected with each other. However, the probability in a random graph that two nodes are connected with each other is $C_i = p$. This can be approximated by

$$C_i \sim \frac{z}{N}, \tag{1.6}$$

because the mean degree of a node is $z = p(N - 1) \sim pN$.

1.2.5 Small-World Networks

Small-world networks were introduced by Watts and Strogatz [115]. They can be obtained via the following algorithm. First, arrange all nodes on a ring and connect each node with its $k/2$ nearest neighbors. Second, start with an arbitrary node i and rewire its connection to its nearest neighbor on, e.g., the left side with probability p_{rw} to any other node j in the network. If node i and j are already connected reject this selection and change nothing. Then choose the next node in the ring in a, e.g., clockwise direction and repeat this procedure. Third, after all next neighbor connections have been checked repeat this procedure for the second and all higher next neighbors successively. This algorithm guarantees that each connection occurring in the network is chosen exactly once to test for a rewiring with probability p_{rw}. The rewiring probability p_{rw} controls the disorder of the resulting topology. For $p_{rw} = 0$ the regular topology is conserved, whereas $p_{rw} = 1.0$ leads to a random network. Intermediate values $0 < p_{rw} < 1$ give a topological structure that is between regular and random.

1.2.6 Scale-Free Networks

Neither random nor small-world networks have a property frequently observed in real-world networks, namely a scale-free behavior of the degrees

$$P(k) \sim k^{-\gamma}. \tag{1.7}$$

```
which means that there is no “top” or “bottom”
1: $t = 0$
2: Start with $N_0$ unconnected nodes
3: **repeat**
4:     Add one new node to the existing network consisting thus far of $N_t$
       nodes.
5:     Connect the new node to $e$ ($\leq N_0$) nodes from the existing network.
       A node is chosen based on the degree distribution of the node,
           $$p_i = \frac{k_i}{\sum_j k_j},$$
6:     $t = t + 1$
7:     $N_t = N_{t-1} + 1$
8: **until** $N_t = N$
```

Algorithm 1.1: Generation of a scale-free network (*preferential attachment*)

To explain this feature Barabasi and Albert introduced a model [4] now known as the Barabasi–Albert (BA) or *preferential attachment* model [96] that results in so-called scale-free networks which have a degree distribution following a power law [4]. The major difference between the *preferential attachment* model and the other algorithms described above to generate random or small-world networks is that the *preferential attachment* model does not assume a fixed number of nodes N and then start to rewire them with fixed probability with other nodes but N grows and is connected with a certain probability (is not constant) to other nodes depending on their degree. In Algorithm 1.1 we provide a principle algorithm to generate a scale-free network.

1.2.7 Trees

A graph G is a *tree* if it has no loops (cycles) in G. This means that a tree is an acyclic graph. Alternatively, upon removal of an edge a connected tree becomes unconnected. Trees were first studied by Cayley [30, 31] and are, in addition to their importance for graph theory, an important data structure in computer science which appears in many different algorithms. Figure 1.2 shows two trees. We want to emphasize that a tree does not represent a hierarchy, in the graph of a tree. This is in contrast to rooted trees. A rooted tree is obtained, e.g., from a tree by the identification of a so-called root node which forms the start of a hierarchy. In Fig. 1.3 we show two rooted trees obtained from the tree on the right-hand side of Fig. 1.2. The important difference is that rooted trees are ordered; they have a “top” corresponding to the root node and a “bottom” corresponding to the leaf nodes having no children. It is interesting to note that there is no restriction to the degree a node can have. Apparently, the minimal number is one because otherwise

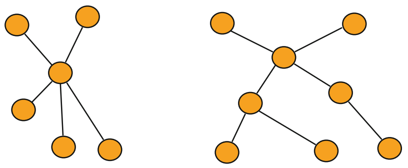
Fig. 1.2 Trees

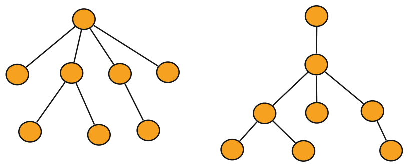
Fig. 1.3 Rooted trees obtained from the tree on the right-hand side in Fig. 1.2. *Left*: Root node is the third node from the top. *Right*: Root node is the leftmost node at the *top*

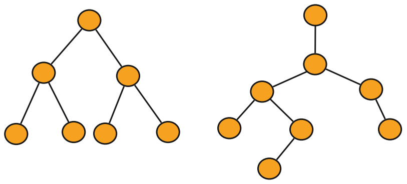
Fig. 1.4 Rooted binary trees

the tree would be unconnected, however, other than that it is arbitrary. This brings us to the next subclass of trees, rooted binary trees.

A special case of a rooted tree is a rooted binary tree. A node in a binary tree has at most two children. Figure 1.4 shows two examples of rooted binary trees, frequently just called binary trees. In Fig. 1.4 one can see that a node has at most two children (maximal degree of a node is three). Intermediate nodes can have only one child. This holds also for the root nodes as the right figure shows. Finally, we want to mention that a disjoint union of trees is called a forest.

1.2.8 Generalized Trees

The graph class of directed *generalized trees* was introduced in [39,90]. Generalized trees are an important extension to trees maintaining their characteristic of having a hierarchy but in addition allowing a richer connectivity among nodes. Before we provide a formal definition we give a motivation for their introduction visualized by Fig. 1.5. On the left side in Fig. 1.5, a (normal) tree is shown. The dotted horizontal lines should remind the reader that a tree is a hierarchical graph and the dotted lines explicitly represent the hierarchy levels. These lines are included for didactic reasons only and are not actually part of the tree. On the right-hand side of Fig. 1.5 a generalized tree is shown obtained from the tree on the left side by including two additional edges, labeled E_2 and E_3. In general, edges that connect nodes on the

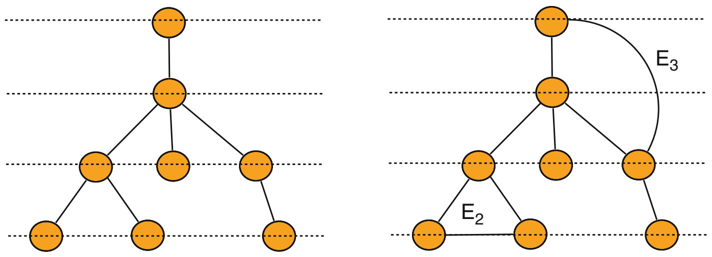

Fig. 1.5 *Left*: Tree. *Right*: Generalized tree

same hierarchical level are of type E_2 and edges that connect nodes on different hierarchical levels that are farther apart than one level are of type E_3 (formally defined below). From this, we note that every generalized tree will result in a tree after deleting *all* edges of type E_2 and E_3 from the generalized tree. Vice versa, starting from a tree and including edges of type E_2 and/or type E_3 results in a generalized tree. A formal definition is given as follows [50].

Definition 1 (Generalized Tree). A generalized tree GT_i is defined by a vertex set V, an edge set E, a level set L, and a multilevel function $\mathcal{L}_i$. The vertex and edge set define the connectivity and the level set and the multilevel function induce a hierarchy between the nodes of GT_i. The index $i \in V$ indicates the root.

The multilevel function is defined as follows.

Definition 2 (Multilevel Function). The function $\mathcal{L}_i : V \setminus \{i\} \to L$ is called a multilevel function.

The multilevel function $\mathcal{L}_i$ assigns to all nodes except i an element $l \in L$ that corresponds to the level it will be assigned. From these definitions it is immediately clear that a generalized tree is similar to a graph but additionally equipped with a level set L and a multilevel function $\mathcal{L}_i$ introducing a node grouping corresponding to the introduction of a hierarchy between nodes and sets thereof.

Definition 3 (Edge Types). A generalized tree GT_i has three edge types:

- Edges with $|\mathcal{L}_i(m) - \mathcal{L}_i(n)| = 1$ are called kernel edges (E_1).
- Edges with $|\mathcal{L}_i(m) - \mathcal{L}_i(n)| = 0$ are called cross edges (E_2).
- Edges with $|\mathcal{L}_i(m) - \mathcal{L}_i(n)| > 1$ are called up edges (E_3).

We want to remark that for directed generalized trees edge type E_3 will be split into two edge types: one for up and another for down links. Using Definition 1 a tree is characterized by $|\mathcal{L}_i(m) - \mathcal{L}_i(n)| = 1$ for all node pairs (m, n).

From the given definitions and the visualization in Fig. 1.5 it is apparent that a generalized tree is between a tree and a graph. It is hierarchical like a tree, but can contain cycles like a graph which is not hierarchical.

1.3 Structural Network Analysis

In this section, we summarize measures to characterize structural properties of networks.

1.3.1 Degree Distribution

Degree distributions [16, 88] can be calculated by

$$P(k) := \frac{|\delta_k(v)|}{N}, \tag{1.8}$$

where $|\delta_k(v)|$ denotes the number of vertices in the network G of degree k and N denotes the size of G (number of nodes). Equation (1.8) is just the proportion of vertices in G having degree k. The degree k_i of node i is the number of links connected with node i. From this, it follows that (1.8) also has the meaning that a randomly chosen node in the network has, with probability $P(k)$, k links to other nodes.

It was an interesting and important finding that many real world networks like the World Wide Web (WWW), the Internet, social networks, citation networks, or food webs [1, 16, 19, 22, 29, 42] are not Poisson distributed but follow a power law

$$P(k) \sim k^{-\gamma}, \quad \gamma > 1. \tag{1.9}$$

1.3.2 Clustering Coefficient

The clustering coefficient C_i is a local measure defined for every node i. It is defined as the fraction of connections (E_i) between nearest neighbors of i among each other divided by the maximum number of such connections

$$C_i = \frac{2E_i}{k_i(k_i - 1)}. \tag{1.10}$$

The clustering coefficient can be interpreted as the probability that two nearest neighbors of i are connected with each other.

1.3.3 Path-Based Measures

Graph-theoretical quantities or properties are often used to characterize special types or classes of complex networks. For example, it turned out that average path lengths and diameters of certain biological networks are rather small compared to the size of a network [86, 88, 98]. Related to this is the so-called "small-world" property [115] that has been observed in a number of network types, e.g., social, metabolic, and protein interaction networks in molecular biology [86, 88, 98]. We give a brief overview of path-based network measures [17, 21, 23, 67, 71, 107].

Distance Matrix

$(d(v_i, v_j))_{v_i, v_j \in V}$. $d(v_i, v_j)$ denotes the shortest distance (path) between v_i and v_j measured in the number of edges or nodes that are between start node v_i and end node v_j.

Mean or Characteristic Distance

$$\bar{d}(G) := \frac{1}{\binom{N}{2}} \sum_{v_i \neq v_j \in V} d(v_i, v_j). \tag{1.11}$$

E is the total number of edges in the network.

j-Sphere

The set

$$S_j(v_i, G) := \{v \in V \mid d(v_i, v) = j,\ j \geq 1\}, \tag{1.12}$$

is called the j-sphere of v_i regarding G. Starting from v_i, the cardinality $|S_j(v_i, G)|$ denotes the number of vertices that have a shortest distance equal to j.

Eccentricity, Diameter, and Radius

Let $G = (V, E)$ be a graph. Then,

$$\sigma(v) = \max_{u \in V} d(u, v), \tag{1.13}$$

is called the eccentricity of $v \in V$.

$$\rho(G) = \max_{v \in V} \sigma(v), \tag{1.14}$$

and

$$r(G) = \min_{v \in V} \sigma(v), \tag{1.15}$$

are called the diameter and radius of G, respectively.

Degree, Degree Statistics, and Edge Density

For undirected graphs $G = (V, E)$, $k_v = \sum_i A_{v,v_i}$ equals the number of edges which are adjacent to $v \in V$. k_v is called the degree of node v. From this, one obtains straightforwardly the following degree measures for the whole network:

$$k = k(G) := \sum_{v \in V} \frac{k_v}{N}, \tag{1.16}$$

$$\sigma_k(G) := \frac{1}{N-1} \sum_{v \in V} (k_v - k)^2, \tag{1.17}$$

and

$$\tau_k(G) := \frac{1}{N} \sum_{v \in V} |k_v - k|. \tag{1.18}$$

Equation (1.16) is the mean degree of the network, (1.17) is the variance of the degree, and (1.18) is the mean of absolute distances between k_v and k. Finally, the edge density of G is defined as

$$\beta(G) := \frac{E}{\binom{N}{2}}. \tag{1.19}$$

Further network statistics and advanced aspects can be found in, e.g., [21, 67, 71, 107].

1.3.4 Centrality Measures

Identifying *important* vertices in networks is an interesting problem that has gained much attention especially in the context of communication networks. For example, the communication among a group of humans forms a communication network. Social scientists in the late 1940s developed graph-theoretical measures to detect *important* vertices in networks. An important class of such measures is based on the centrality concept [66, 69, 114] which intuitively tries to identify nodes that are *central* to the communication within the network among all nodes. There are two fundamentally different types of centrality measures [58, 59]. The first type of measures evaluates the centrality of each node in a network and is called *point centrality* measures where the word "point" refers to a node or vertex. The second type is called *graph centrality* measures because it assigns a centrality value to the *whole* network.

Point Centrality

In the following we provide some examples of point centrality measures.

Degree Centrality

For an undirected graph $G = (V, E)$, the degree centrality of a vertex $v \in V$ is simply defined as its degree; i.e.,

$$C_D(v) = k_v. \tag{1.20}$$

For a directed network, the degree centrality can be analogously defined by using the definition of in-degree and out-degree.

Betweenness Centrality

The centrality index *betweenness* is based on shortest paths found in the network [5, 6, 18, 58, 66, 83, 88, 103, 104, 114] and is defined by

$$C_B(v_k) = \sum_{v_i, v_j \in V, v_i \neq v_j} \frac{\sigma_{v_i v_j}(v_k)}{\sigma_{v_i v_j}}. \tag{1.21}$$

Here $\sigma_{v_i v_j}$ denotes the number of shortest paths from v_i to v_j and $\sigma_{v_i v_j}(v_k)$ the number of shortest paths from v_i to v_j that include node v_k. That means

$$\frac{\sigma_{v_i v_j}(v_k)}{\sigma_{v_i v_j}}, \tag{1.22}$$

is the probability that node v_k lies on a shortest path connecting v_i with v_j. Hence, $C_B(v_k)$ evaluates the appearance of node v_k on all shortest paths in a network.

Closeness Centrality

The centrality index *closeness* tries to measure how close a node is to other nodes in the network. This is done in terms of communication distance as measured by the number of edges between two nodes if connected via the shortest path.

$$C_C(v_k) = \frac{1}{\sum_{i=1}^{N} d(v_k, v_i)}. \tag{1.23}$$

Here $d(v_k, v_i)$ is the number of edges on a shortest path between node v_k and v_i. In the case where there are multiple shortest paths connecting v_k with v_i, $d(v_k, v_i)$ is unchanged.

Graph Centrality

To evaluate the centrality of whole networks instead of single nodes in the network *graph centrality* measures have been suggested which form extensions to the three measures discussed above [59]. The basic idea is to use these individual measures to obtain an average characteristic for the whole network. It has been suggested to calculate

$$C_x = \frac{\sum_{i=1}^{N} C_x(v^m) - C_x(v_i)}{C_x^{max}}. \tag{1.24}$$

Here x stands for any of the three point centrality measures,

$$C_x(v^m) = \max_i\{C_x(v_i)\}, \tag{1.25}$$

for the maximal value of $C_x(v_i)$ that can be found in the network and C_x^{max} for the maximal value possible for a network with N nodes,

$$C_x^{max} = \max_{G \in \mathcal{G}(N)} \sum_{i=1}^{N} C_x(v^m) - C_x(v_i). \tag{1.26}$$

Degree Centrality

For the degree centrality of a network one obtains

$$C_d = \frac{\sum_{i=1}^{N} C_d(v^m) - C_d(v_i)}{N^2 - 3N + 2}. \tag{1.27}$$

Intuitively, the denominator is obtained by remembering that the maximal degree of a node is $N - 1$, hence, $C_x(v^m) - C_x(v_i) = N - 2$ because the minimal degree is one. This number multiplied by $N - 1$ (one node needs to have degree one) gives the denominator in (1.27).

Betweenness Centrality

For the betweenness centrality of a network [58] one obtains

$$C_b = \frac{\sum_{i=1}^{N} C_d(v^m) - C_d(v_i)}{N^3 - 4N^2 + 5N - 2}. \tag{1.28}$$

Closeness Centrality

For the closeness centrality of a network one obtains

$$C_c = \frac{2N-3}{N^3 - 4N^2 + 5N - 2} \sum_{i=1}^{N} C_d(v^m) - C_d(v_i). \tag{1.29}$$

Extended Centrality Measures

In addition to the classical centrality measures described above there are many extensions. Here we present some of these.

Eigenvector Centrality

The eigenvector centrality, a point centrality measure, was introduced by Bonacich [15]. The key idea of eigenvector centrality is to express that an important vertex is connected to important neighbors. To define the eigenvector centrality measures one needs to find the eigenvector of the adjacency matrix A of graph G with the largest eigenvalue. Then eigenvector centrality is given by

$$C_e = x^m = \frac{1}{\lambda^m} A x^m. \tag{1.30}$$

Here λ^m is the largest eigenvalue and x^m the corresponding eigenvector solving the equation

$$\lambda^m x^m = A x^m. \tag{1.31}$$

Hence C_e is the principle eigenvector of A.

We want to emphasize that eigenvector centrality is a point centrality measure because each vertex in the network obtains a value corresponding to the component of C_e. In [83], advanced properties and further possibilities to compute eigenvector centrality measures are presented.

Joint Betweenness Centrality

The centrality measures discussed above, including *betweenness*, form a family of measures [58] that have been introduced with the purpose of analyzing communication networks. It is interesting to note that all point centrality measures focus solely on one vertex in the network. In the context of gene networks, which also form communication networks, the identification of a function of genes is an outstanding problem. It has been suggested to identify the unknown function of a gene by associating it with the known function of another gene. Because all measures from the centrality family are point measures they cannot be used for such studies. For this reason it has been suggested that an extension involving more than one node be called *joint betweenness* (JB) [47]. JB is a natural extension of the betweenness centrality evaluating the joint occurrence of two nodes on shortest communication paths in the network. Formally, it is defined as

$$C_{jb}(v_m, v_n) = \sum_{v_i, v_j \in V, v_i \neq v_j} \frac{\sigma_{v_i,v_j}(v_m, v_n)}{\sigma_{v_i,v_j}}. \tag{1.32}$$

Here σ_{v_i,v_j} is the number of shortest paths connecting node v_i with node v_j and $\sigma_{v_i,v_j}(v_m, v_n)$ is the number of shortest paths connecting v_i with v_j that contain the nodes v_m and v_n. Similar to other measures of the centrality family, it is sometimes more useful to use a different normalization. In fact, for the analysis conducted in [47] the following modification has been used,

$$C_{jb}(v_m, v_n) = \sum_{v_i, v_j \in V, v_i \neq v_j} \frac{\sigma_{v_i,v_j}(v_m, v_n)}{\sigma_{max}}. \tag{1.33}$$

Here σ_{max} is defined as

$$\sigma_{max} = \max_{v_i, v_j}\{\sigma_{v_i,v_j}\}. \tag{1.34}$$

1.3.5 Comparative Network Analysis

In this section measures structurally comparing whole networks are reviewed.

Measures Based on Isomorphic Relations

Classical graph similarity or distance methods deal with finding appropriate measures which are based on isomorphic and subgraph relations [75–77, 108, 109, 117]. A prominent example of such a measure is the Zelinka-distance [117], where this graph distance is based on the principle that two graphs are more similar, the bigger the common induced isomorphic subgraph is. First, Zelinka introduced this measure for unlabeled graphs with the same number of vertices. Later, Sobik [108, 109] and Kaden [75–77] generalized this measure for arbitrary graphs allowing them to have even different order. It is known that the subgraph isomorphism problem is NP-complete [113]. This implies for large graphs that these methods can be computationally demanding. A key result for exact graph matching was found by Zelinka [117].

Theorem 1. *Let G, $\tilde{G}$ be unlabeled graphs without loops and multiple edges. Further, let $|V| = |\tilde{V}| = n$. $\overline{SUB}_m(G)$ denotes the set of induced subgraphs of order m. $G^\star$ denotes the isomorphism classes of such graphs in which G lies and let*

$$SUB_m(G) := \{G^\star \,|\, G \in \overline{SUB}_m(G)\}. \tag{1.35}$$

$SUB_m(G)$ is just the set of isomorphism classes in which the induced subgraphs of G with order m lie. Then,

$$d_Z(G, \tilde{G}) := n - SIM(G, \tilde{G}), \tag{1.36}$$

is a graph metric, where

$$SIM(G, \tilde{G}) := \max\{m | SUB_m(G) \cap SUB_m(\tilde{G}) \neq \emptyset\}, \tag{1.37}$$

holds.

Sobik [108,109] and Kaden [75–77] generalized this theorem by considering labeled graphs with a different number of vertices.

Theorem 2. *Let $G := (V, E, f_V, f_E, A_V, A_E)$ be a finite, labeled, and directed graph. A_V, A_E denote finite, nonempty vertex and edge alphabets and $f_V : V \rightarrow A_V$, $f_E : E \rightarrow A_E$ the associated vertex and edge labeling functions. Now, let G and $\tilde{G}$ be finite labeled graphs of arbitrary orders. Then,*

$$d_S(G, \tilde{G}) := \max\{|G|, |\tilde{G}|\} - SIM(G, \tilde{G}), \tag{1.38}$$

is a graph metric.

Another classical graph distance measure based on the maximum common subgraph of two graphs has been found by Bunke et al. [25, 26, 28].

Theorem 3. *Let G and $\tilde{G}$ be graphs and let G_{MCS} be their maximum common subgraph. Then, the distance measure*

$$d_{MCS}(G, \tilde{G}) := 1 - \frac{|V|_{MCS}}{\max(|V|, |\tilde{V}|)}, \tag{1.39}$$

is a graph metric.

Measures Based on Graph Transformations

In contrast to graph similarity measures from the exact graph matching paradigm, i.e., those based on isomorphic relations, a well-known class of graph similarity measures from inexact graph matching is based on graph transformations. The main idea behind this concept is not to match graphs exactly because one often wants to take structural errors of the underlying graphs into account. Therefore, this concept is often referred to as error-tolerant graph matching [25, 26, 28, 91]. For example, the so-called graph edit distance (GED) [24, 27, 28, 91] is a prominent example of such a graph similarity measure. The definition of GED is based on weighted transformation steps, e.g., deletions, substitutions, and insertions of vertices and edges, and, hence, the distance of two graphs is defined as the minimum cost of graph transformations that transform (map) one graph into another graph. The key result of error-tolerant graph matching, originally stated by Bunke [24], can be expressed as follows.

Theorem 4. *Let $d(G, \tilde{G})$ be the costs for determining the optimal inexact match between G and $\tilde{G}$ where an optimal inexact match is a sequence of graph transformations that transforms a graph G to $\tilde{G}$ by producing minimal edit costs. Then, it holds that $d(G, \tilde{G})$ is a graph metric.*

Regarding the computational complexity of GED, we want to remark that for unlabeled graphs there is no algorithm to compute GED efficiently [28, 91, 118]. For uniquely labeled graphs, it has been proven [43] that the computational complexity to compute GED is $O(|V|^2)$.

Measures Based on Graph Grammars

Methods to determine the similarity or distance between graphs based on graph grammars also belong to the paradigm of inexact graph matching. A classical contribution in this field has been made by Gernert [63, 64]. We want to note that the application of grammar-based measures is complex because the underlying graph grammar is quite often difficult to obtain.

Methods Based on Machine Learning Techniques

Machine learning techniques can be divided into two major categories: supervised and unsupervised learning methods [38,72]. A newly developed supervised learning method, based on support vector machines [38], to measure the structural similarity of graphs is based on using so-called graph kernels [62, 73]. A graph kernel is a function $K : \mathcal{G} \times \mathcal{G} \longrightarrow \mathbb{R}$, for $G \in \mathcal{G}$, that maps the data implicitly into a high-dimensional feature space. For example, some graph kernels are based on the principle to determine the frequency of certain subgraph patterns of the given graph set and then to apply a proper kernel function to the obtained subgraphs. Following this principle Horváth et al. [73] proposed a graph kernel that is based on mapping graphs into cyclic graph patterns. Besides cycle-based graph kernels, so-called random-walk-based kernels [62,80] are also often used to define graph kernels [36].

Another method to detect the structural similarity of graphs is based on dynamic programming [7]. In the following we just give an outline of the main construction