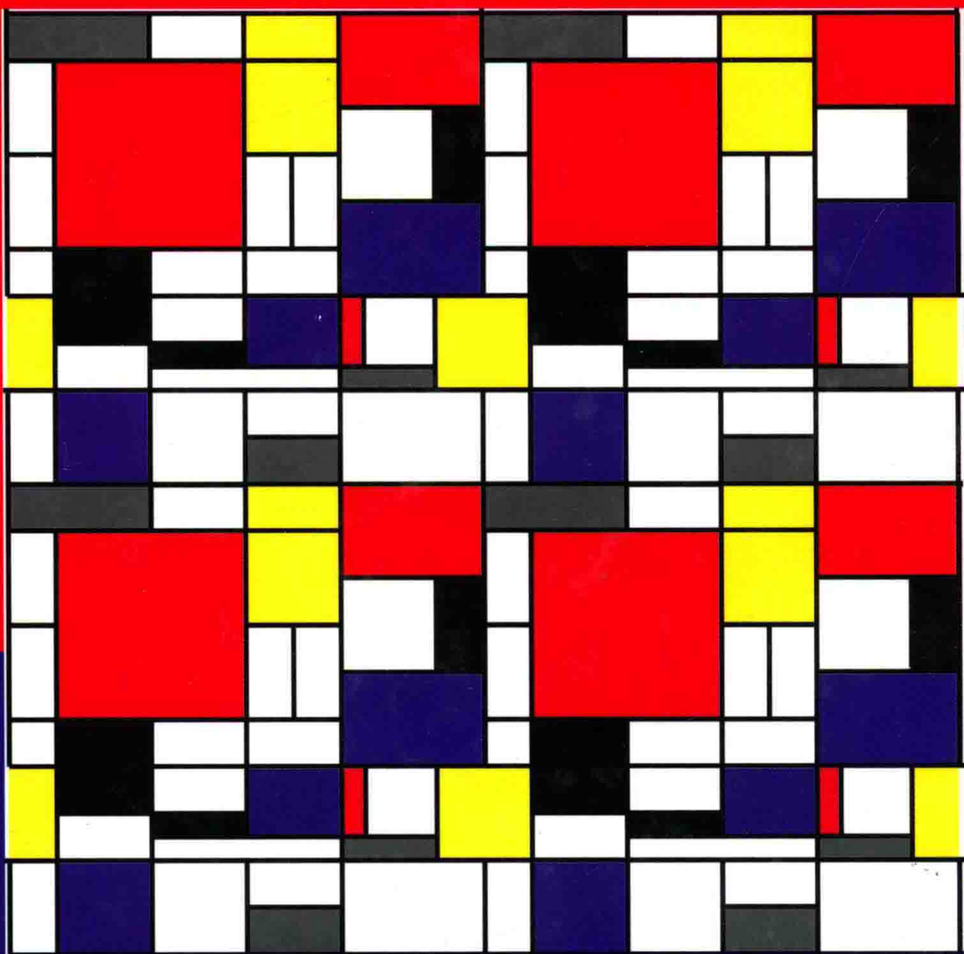
DISCRETE MATHEMATICS AND ITS APPLICATIONS
Series Editor KENNETH H. ROSEN

Handbook of Graph Drawing and Visualization



Edited by

Roberto Tamassia

CRC Press
Taylor & Francis Group

DISCRETE MATHEMATICS AND ITS APPLICATIONS
Series Editor KENNETH H. ROSEN

Handbook of Graph Drawing and Visualization

Edited by
Roberto Tamassia
Brown University
Providence, Rhode Island, USA

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
A CHAPMAN & HALL BOOK

MIX
Paper from
responsible sources
FSC® C014174

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed on acid-free paper
Version Date: 20130719

International Standard Book Number-13: 978-1-58488-412-5 (Hardback)

Library of Congress Cataloging-in-Publication Data

Handbook of graph drawing and visualization / editor, Roberto Tamassia.
pages cm -- (Discrete mathematics and its applications)
Includes bibliographical references and index.
ISBN 978-1-58488-412-5 (hardcover : alk. paper)
1. Graph theory--Handbooks, manuals, etc. 2. Graphic methods--Handbooks, manuals, etc. 3. Information visualization--Handbooks, manuals, etc. I. Tamassia, Roberto, 1960-

QA166.H358 2014
511'.5--dc23 2013025835

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Handbook of
Graph Drawing and Visualization

DISCRETE MATHEMATICS AND ITS APPLICATIONS

Series Editor
Kenneth H. Rosen, Ph.D.

R. B. J. T. Allenby and Alan Slomson, How to Count: An Introduction to Combinatorics, Third Edition

Craig P. Bauer, Secret History: The Story of Cryptology

Juergen Bierbrauer, Introduction to Coding Theory

Katalin Bimbó, Combinatory Logic: Pure, Applied and Typed

Donald Bindner and Martin Erickson, A Student's Guide to the Study, Practice, and Tools of Modern Mathematics

Francine Blanchet-Sadri, Algorithmic Combinatorics on Partial Words

Miklós Bóna, Combinatorics of Permutations, Second Edition

Jason I. Brown, Discrete Structures and Their Interactions

Richard A. Brualdi and Dragoš Cvetković, A Combinatorial Approach to Matrix Theory and Its Applications

Kun-Mao Chao and Bang Ye Wu, Spanning Trees and Optimization Problems

Charalambos A. Charalambides, Enumerative Combinatorics

Gary Chartrand and Ping Zhang, Chromatic Graph Theory

Henri Cohen, Gerhard Frey, et al., Handbook of Elliptic and Hyperelliptic Curve Cryptography

Charles J. Colbourn and Jeffrey H. Dinitz, Handbook of Combinatorial Designs, Second Edition

Abhijit Das, Computational Number Theory

Martin Erickson, Pearls of Discrete Mathematics

Martin Erickson and Anthony Vazzana, Introduction to Number Theory

Steven Furino, Ying Miao, and Jianxing Yin, Frames and Resolvable Designs: Uses, Constructions, and Existence

Mark S. Gockenbach, Finite-Dimensional Linear Algebra

Randy Goldberg and Lance Riek, A Practical Handbook of Speech Coders

Titles (continued)

Jacob E. Goodman and Joseph O'Rourke, Handbook of Discrete and Computational Geometry, Second Edition

Jonathan L. Gross, Combinatorial Methods with Computer Applications

Jonathan L. Gross and Jay Yellen, Graph Theory and Its Applications, Second Edition

Jonathan L. Gross and Jay Yellen, Handbook of Graph Theory

David S. Gunderson, Handbook of Mathematical Induction: Theory and Applications

Richard Hammack, Wilfried Imrich, and Sandi Klavžar, Handbook of Product Graphs, Second Edition

Darrel R. Hankerson, Greg A. Harris, and Peter D. Johnson, Introduction to Information Theory and Data Compression, Second Edition

Darel W. Hardy, Fred Richman, and Carol L. Walker, Applied Algebra: Codes, Ciphers, and Discrete Algorithms, Second Edition

Daryl D. Harms, Miroslav Kraetzl, Charles J. Colbourn, and John S. Devitt, Network Reliability: Experiments with a Symbolic Algebra Environment

Silvia Heubach and Toufik Mansour, Combinatorics of Compositions and Words

Leslie Hogben, Handbook of Linear Algebra

Derek F. Holt with Bettina Eick and Eamonn A. O'Brien, Handbook of Computational Group Theory

David M. Jackson and Terry I. Visentin, An Atlas of Smaller Maps in Orientable and Nonorientable Surfaces

Richard E. Klima, Neil P. Sigmon, and Ernest L. Stitzinger, Applications of Abstract Algebra with Maple™ and MATLAB®, Second Edition

Richard E. Klima and Neil P. Sigmon, Cryptology: Classical and Modern with Maplets

Patrick Knupp and Kambiz Salari, Verification of Computer Codes in Computational Science and Engineering

William Kocay and Donald L. Kreher, Graphs, Algorithms, and Optimization

Donald L. Kreher and Douglas R. Stinson, Combinatorial Algorithms: Generation Enumeration and Search

Hang T. Lau, A Java Library of Graph Algorithms and Optimization

C. C. Lindner and C. A. Rodger, Design Theory, Second Edition

San Ling, Huaxiong Wang, and Chaoping Xing, Algebraic Curves in Cryptography

Nicholas A. Loehr, Bijective Combinatorics

Toufik Mansour, Combinatorics of Set Partitions

Alasdair McAndrew, Introduction to Cryptography with Open-Source Software

Elliott Mendelson, Introduction to Mathematical Logic, Fifth Edition

Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, Handbook of Applied Cryptography

Titles (continued)

Stig F. Mjølsnes, A Multidisciplinary Introduction to Information Security

Jason J. Molitierno, Applications of Combinatorial Matrix Theory to Laplacian Matrices of Graphs

Richard A. Mollin, Advanced Number Theory with Applications

Richard A. Mollin, Algebraic Number Theory, Second Edition

Richard A. Mollin, Codes: The Guide to Secrecy from Ancient to Modern Times

Richard A. Mollin, Fundamental Number Theory with Applications, Second Edition

Richard A. Mollin, An Introduction to Cryptography, Second Edition

Richard A. Mollin, Quadratics

Richard A. Mollin, RSA and Public-Key Cryptography

Carlos J. Moreno and Samuel S. Wagstaff, Jr., Sums of Squares of Integers

Gary L. Mullen and Daniel Panario, Handbook of Finite Fields

Goutam Paul and Subhamoy Maitra, RC4 Stream Cipher and Its Variants

Dingyi Pei, Authentication Codes and Combinatorial Designs

Kenneth H. Rosen, Handbook of Discrete and Combinatorial Mathematics

Douglas R. Shier and K.T. Wallenius, Applied Mathematical Modeling: A Multidisciplinary Approach

Alexander Stanoyevitch, Introduction to Cryptography with Mathematical Foundations and Computer Implementations

Jörn Steuding, Diophantine Analysis

Douglas R. Stinson, Cryptography: Theory and Practice, Third Edition

Roberto Tamassia, Handbook of Graph Drawing and Visualization

Roberto Togneri and Christopher J. deSilva, Fundamentals of Information Theory and Coding Design

W. D. Wallis, Introduction to Combinatorial Designs, Second Edition

W. D. Wallis and J. C. George, Introduction to Combinatorics

Jiacun Wang, Handbook of Finite State Based Models and Applications

Lawrence C. Washington, Elliptic Curves: Number Theory and Cryptography, Second Edition

Preface

Objective

This handbook aims at providing a broad survey of the field of graph drawing. It covers topological and geometric foundations, algorithms, software systems, and visualization applications in business, education, science, and engineering.

The intended readership of this handbook includes:

- Practitioners and researchers in traditional and emerging disciplines of the physical, life, and social sciences interested in understanding and using graph drawing methods and graph visualization systems in their field.
- Information technology practitioners and software developers aiming to incorporate graph drawing solutions into their products.
- Researchers and students in graph drawing and information visualization seeking an up-to-date survey of the field.
- Researchers and students in related fields of mathematics and computer science (including graph theory, computational geometry, information visualization, software engineering, user interfaces, social networks, and data management) interested in using graph-drawing techniques in support of their research.

Organization

The chapters of this handbook are organized into four parts, as follows.

Topological and Geometric Foundations of Graph Drawing The first part (Chapters 1–4) deals with fundamental topological and geometric concepts and techniques used in graph drawing: planarity testing and embedding, crossings and planarization, symmetric drawings, and proximity drawings.

Graph Drawing Algorithms The second part (Chapters 5–14) presents an extensive collection of algorithms for constructing drawings of graphs. Some methods are designed to draw special classes of graphs (e.g., trees, planar graphs, or directed acyclic graphs) while other methods work for general graphs. Topics covered in this part include tree drawing algorithms, planar straight-line drawing algorithms, planar orthogonal and polyline drawing algorithms, spine and radial drawings, circular drawing algorithms, rectangular drawing algorithms, simultaneous embeddings, force-directed methods, hierarchical drawing algorithms, three-dimensional drawing algoritms, and labeling algorithms.

Graph Drawing Systems The third part begins by introducing the GraphML language for representing graphs and their drawings (Chapter 16). Next, it overviews three software systems for constructing drawings of graphs: OGDF, GDToolkit, and PIGALE (Chapters 17–19).

Applications of Graph Drawing The fourth part (Chapters 20–26) gives examples of the use of graph drawing methods for the visualization of networks in various important application domains: biological networks, computer security, data analytics, education, computer networks, and social networks.

Each chapter is intended to be self-contained and has its own bibliography.

Acknowledgments

I would like to thank all the authors of the chapters in this handbook and all the reviewers who have provided expert feedback on the initial drafts and revised versions of the chapters.

This handbook is a collective effort of the graph drawing research community, which has developed around the annual Symposium on Graph Drawing. I am grateful to the people who have founded with me this conference and have continued providing leadership for it: Franz Brandenburg, Giuseppe Di Battista, Peter Eades, Hubert de Fraysseix, Takao Nishizeki, Pierre Rosenstiehl, and Ioannis Tollis.

A huge thanks goes to Sunil Nair for proposing this handbook project and supporting its development. I truly appreciate his constant encouragement and patience. Help received from the entire CRC Press production team, and especially from Andre Barnett, Kari Budyk, Rachel Holt, Jim McGovern, and Shashi Kumar, is gratefully acknowledged. I would like to thank also my executive assistant Angel Murakami for her expert proofreading of the chapters.

I am indebted to Carlo Batini for introducing me to the problem of drawing graphs and inspiring me to pursue an academic path. A special thanks also goes to Franco Preparata, whose guidance and support, first as PhD advisor and then as colleague, have shaped by career.

Finally, warm thanks go to Isabel Cruz, Giuseppe Di Battista, Michael Goodrich, and Ioannis Tollis for their encouragement and support throughout this project.

Roberto Tamassia

About the Editor

Roberto Tamassia is the Plastech Professor of Computer Science and the Chair of the Department of Computer Science at Brown University. He is also the Director of Brown's Center for Geometric Computing. His research interests include analysis, design, and implementation of algorithms, applied cryptography, cloud computing, computational geometry data security, and graph drawing. He has published six textbooks and more than 250 research articles and books in the above areas and has given more than 70 invited lectures worldwide. He is a Fellow of the American Association for the Advancement of Science (AAAS), the Association for Computing Machinery (ACM), and the Institute of Electrical and Electronics Engineers (IEEE). He is the recipient of a Technical Achievement Award from the IEEE Computer Society for pioneering the field of graph drawing. He is listed among the 360 most cited computer science authors by Thomson Scientific, Institute for Scientific Information (ISI). He serves regularly on program committees of international conferences. His research has been funded by ARO, DARPA, NATO, NSF, and several industrial sponsors. He co-founded the Journal of Graph Algorithms and Applications (JGAA) and the Symposium on Graph Drawing. He serves as Co-Editor-in-Chief of JGAA. He received the PhD degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign and the "Laurea" in Electrical Engineering from the "Sapienza" University of Rome.

Contents

1

Planarity Testing and Embedding

Maurizio Patrignani
Roma Tre University

1.1 Introduction

Testing the planarity of a graph and possibly drawing it without intersections is one of the most fascinating and intriguing algorithmic problems of the graph drawing and graph theory areas. Although the problem *per se* can be easily stated, and a complete characterization of planar graphs has been known since 1930, the first linear-time solution to this problem was found only in the 1970s.

Planar graphs play an important role both in the graph theory and in the graph drawing areas. In fact, planar graphs have several interesting properties: for example, they are sparse and 4-colorable, they allow a number of operations to be performed more efficiently than for general graphs, and their inner structure can be described more succinctly and elegantly (see Section 1.2.2). From the information visualization perspective, instead, as edge crossings turn out to be the main reason for reducing readability, planar drawings of graphs are considered clear and comprehensible.

In this chapter, we review a number of different algorithms from the literature for efficiently testing planarity and computing planar embeddings. Our main thesis is that all known linear-time planarity algorithms fall into two categories: cycle based algorithms and vertex addition algorithms. The first family of algorithms is based on the simple observation that in a planar drawing of a graph any cycle necessarily partitions the graph into the inside and outside portion, and this partition can be suitably used to split the embedding problem. Vertex addition algorithms are based on the incremental construction of the final planar drawing starting from planar drawings of smaller graphs. The fact that some algorithms were based on the same paradigm was already envisaged by several researchers [Tho99, HT08]. However, the evidence that all known algorithms boil down to two simple approaches is a relatively new concept.

The chapter is organized as follows: Section 1.2 introduces basic definitions, properties, and characterizations for planar graphs; Section 1.3 formally defines the planarity testing and embedding problems; Section 1.4 follows a historic perspective to introduce the main algorithms and a conventional classification for them. Some algorithmic techniques are common to more than one algorithm and sometimes to all of them. These are collected in Section 1.5. Finally, Sections 1.6 and 1.7 are devoted to the two approaches to the planarity testing problem, namely, the "cycle based" and the "vertex addition" approaches, respectively.

Algorithms for constructing planar drawings of graphs are discussed in Chapters 6 (straight-line drawings), 7 (orthogonal and polyline drawings), and 10 (rectangular drawings). Methods for reducing crossings in nonplanar drawings of graphs are discussed in Chapter 2.

1.2 Properties and Characterizations of Planar Graphs

1.2.1 Basic Definitions

A *graph* $G(V, E)$ is an ordered pair consisting of a finite set V of *vertices* and a finite set E of *edges*, that is, pairs (u, v) of vertices. If each edge is an unordered (ordered) pair of vertices, then the graph is *undirected* (*directed*). An edge (u, v) is a *self-loop* if $u = v$. A graph $G(V, E)$ is *simple* if E is not a multiple set and it does not contain self-loops. For the purposes of this chapter, we can restrict us to simple graphs.

The sets of edges and vertices of G can be also denoted $E(G)$ and $V(G)$, respectively. If edge $(u, v) \in E$, vertices u and v are said to be *adjacent* and (u, v) is said to be *incident* to u and v. Two edges are *adjacent* if they have a vertex in common.

A (rooted) *tree* T is a connected acyclic graph with one distinguished vertex, called the *root* r. A *spanning tree* of a graph G is a tree T such that $V(T) = V(G)$ and $E(T) \subseteq E(G)$.

Given two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, their *union* $G_1 \cup G_2$ is the graph $G(V_1 \cup V_2, E_1 \cup E_2)$. Analogously, their *intersection* $G_1 \cap G_2$ is the graph $G(V_1 \cap V_2, E_1 \cap E_2)$. A graph G_2 is a *subgraph* of G_1 if $G_1 \cup G_2 = G_1$.

Given a graph $G(V, E)$ and a subset V' of V, the subgraph *induced by* V' is the graph $G'(V', E')$, where E' is the set of edges of E that have both endvertices in V'. Given a graph $G(V, E)$ and a subset E' of E, the subgraph *induced by* E' is the graph $G'(V', E')$, where V' is the set of vertices incident to E'. A *subdivision* of an edge (u, v) consists of the insertion of a new node w and the replacement of (u, v) with edges (u, w) and (w, v). A graph G_2 is a *subdivision* of G_1 if it can be obtained from G_1 through a sequence of edge subdivisions.

A *drawing* Γ of a graph G maps each vertex v to a distinct point $\Gamma(v)$ of the plane and each edge (u, v) to a simple open Jordan curve $\Gamma(u, v)$ with endpoints $\Gamma(u)$ and $\Gamma(v)$. A drawing is *planar* if no two distinct edges intersect except, possibly, at common endpoints. A graph is *planar* if it admits a planar drawing. A planar drawing partitions the plane into connected regions called *faces*. The unbounded face is usually called *external face* or *outer face*. If all the vertices are incident to the outer face, the planar drawing is called *outerplanar* and the graph admitting it is an *outerplanar graph*. Given a planar drawing, the (clockwise) circular order of the edges incident to each vertex is fixed. Two planar drawings are *equivalent* if they determine the same circular orderings of the edges incident to each vertex (sometimes called *rotation scheme*). A *(planar) embedding* is an equivalence class of planar drawings and is described by the clockwise circular order of the edges incident to each vertex. A graph together with one of its planar embedding is sometimes referred to as a *plane* graph.

A *path* is a sequence of distinct vertices $v_1, v_2, \ldots, v_k$, with $k \geq 2$, together with the edges $(v_1, v_2), \ldots, (v_{k-1}, v_k)$. The *length* of the path is the number of its edges.

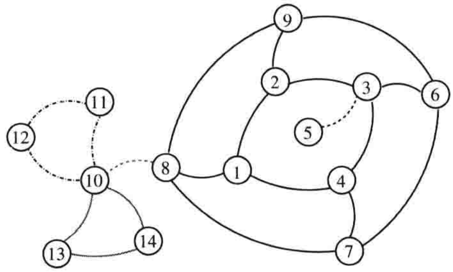
A *cycle* is a sequence of distinct vertices $v_1, v_2, \ldots, v_k$, with $k \geq 2$, together with the edges $(v_1, v_2), \ldots, (v_{k-1}, v_k), (v_k, v_1)$. The *length* of a cycle is the number of its vertices or the number of its edges.

An undirected graph G is *connected* if, for each pair of nodes u and v, G contains a path from u to v. A graph G with at least $k+1$ vertices is *k-connected* if removing any $k-1$ vertices leaves G connected. Equivalently, by Menger's theorem, a graph is k-connected if there are k independent paths between each pair of vertices [Men27]. 3-connected, 2-connected, and 1-connected graphs are also called *triconnected*, *biconnected*, and *simply connected* graphs, respectively. It is usual in the planarity literature to relax the definition of biconnected graph so to include *bridges*, i.e., graphs composed by a single edge between two vertices. A *separating k-set* is a set of k vertices whose removal disconnects the graph. Separating 1- and 2-sets are called *cutvertices* and *separation pairs*, respectively. Hence, a connected graph is biconnected if it has no cutvertices and it is triconnected if it has no separation pairs.

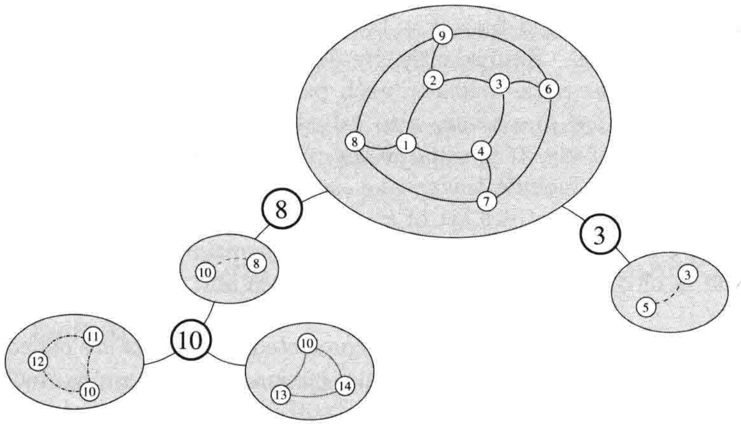
If a graph G is not connected, its maximal connected subgraphs are called the *connected components* of G. If G is connected, its maximal biconnected subgraphs (including bridges) are called the *biconnected components*, or *blocks* of G. Note that a cutvertex belongs to several blocks and that a biconnected graph has only one block. The graph whose vertices are the blocks and the cutvertices of G and whose edges link cutvertices to the blocks they belong to is a tree and is called the *block-cutvertex tree* (or *BC-tree*) of G (see Figure 1.1 for an example).

Given a biconnected graph G, its *triconnected components* are obtained by a complex splitting and merging process. The first linear-time algorithm to compute them was introduced in [HT73], while an implementation of it is described in [GM01]. The computation has two phases: first, G is recursively split into its *split components*; second, some split components are merged together to obtain triconnected components. The *split* operation is performed with respect to a pair of vertices $\{v_1, v_2\}$ of the biconnected (multi)graph G. Suppose the edges of G are divided into the equivalence classes $E_1, E_2, \ldots, E_k$ such that two edges are in the same class if both lie in a common path not containing a vertex in $\{v_1, v_2\}$ except, possibly, as an end point. If there are at least two such classes, then $\{v_1, v_2\}$ is a *split pair*. Let G_1 be the graph induced by E_1 and G_2 be the graph induced by E/E_1. A *split* operation consists of replacing G with G_1' and G_2', where G_1' and G_2' are obtained from G_1 and G_2 by adding the same *virtual edge* (v_1, v_2). The two copies of the virtual edge added to G_1 and G_2 are called *twin virtual edges*. Figure 1.2(b) shows the result of a split operation performed on the graph of Figure 1.2(a) with respect to split pair $\{2, 4\}$. The *split components* of a graph G are obtained by recursively splitting G until no split pair can be found in the obtained graphs. Figure 1.2(c) shows the split components of the graph of Figure 1.2(a). Split components are not unique and, hence, are not suitable for describing the structure of G.

Two split components sharing the same twin virtual edges (v_1, v_2) can be merged by identifying the two copies of v_1 and v_2 and by removing the twin virtual edges. Split components consisting of cycles are called *series split components*, while split components



(a)



(b)

Figure 1.1 A connected graph (a) and its BC-tree (b). Different line styles are used for edges of different blocks.

that have only two vertices are called *parallel split components*. By recursively merging together series split components that share twin virtual edges we obtain *series triconnected components*, while by recursively merging together parallel split components that share twin virtual edges we obtain *parallel triconnected components*. Split components that are not affected by the merging operations described above are called *rigid triconnected components*. Figure 1.3(a) shows the triconnected components of the graph of Figure 1.2(a).

Triconnected components are unique and are used to describe the inner structure of a graph. In fact, a graph G can be succinctly described by its SPQR-tree $\mathcal{T}$, which provides a high-level view of the unique decomposition of the graph into its triconnected components [DT96a, DT96b, GM01]. Namely, each triconnected component corresponds to a node of $\mathcal{T}$. The triconnected component corresponding to a node μ of $\mathcal{T}$ is called the *skeleton of* μ. As there are parallel, series, and rigid triconnected components, their corresponding tree nodes are called P-, S-, and R-nodes, respectively. Triconnected components sharing a virtual edge are adjacent in $\mathcal{T}$. Usually, a fourth type of node, called Q-node, is used to represent an edge (u, v) of G. Q-nodes are the leaves of $\mathcal{T}$ and they don't have skeletons. Tree $\mathcal{T}$ is unrooted, but for some applications, it could be thought as rooted at an arbitrary Q-node. See Figure 1.3 for an example of SPQR-tree.

The connectivity properties of a graph have a strict relationship with its embedding properties. Triconnected planar graphs (and triconnected planar components) have a single