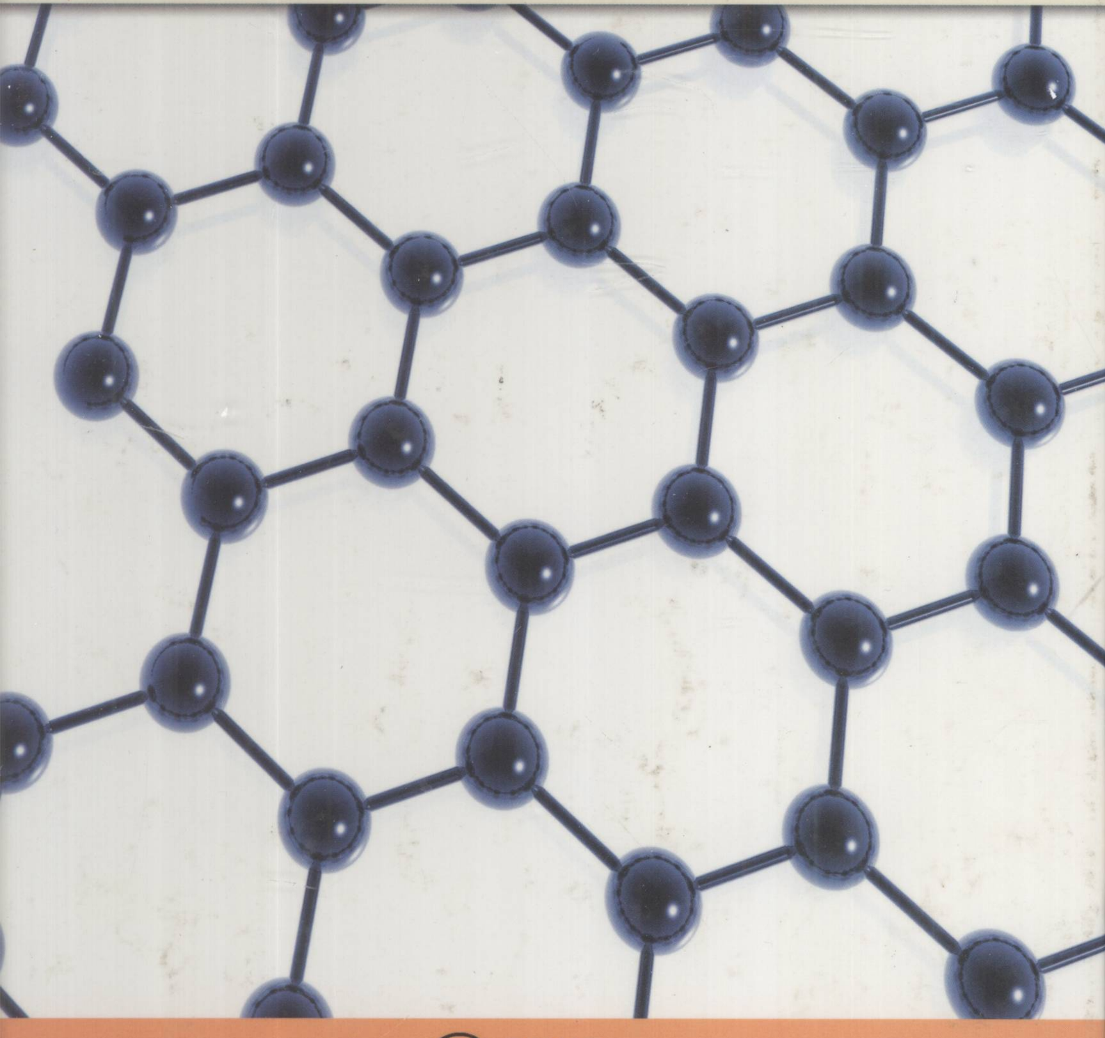
Graph Theory and Interconnection Networks

Lih-Hsing Hsu and Cheng-Kuan Lin



CRC Press
Taylor & Francis Group

Graph Theory and Interconnection Networks

Lih-Hsing Hsu and Cheng-Kuan Lin

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

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

Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number-13: 978-1-4200-4481-2 (Hardcover)

Library of Congress Cataloging-in-Publication Data

Hsu, Lih-Hsing.
Graph theory and interconnection networks / Lih-Hsing Hsu, Cheng-Kuan Lin.
p. cm.
Includes bibliographical references and index.
ISBN 978-1-4200-4481-2 (hardback : acid-free paper)
1. Graph theory. I. Lin, Cheng-Kuan. II. Title.

QA166.H78 2008
511'.5--dc22 2008019917

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

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

Graph Theory and Interconnection Networks

OTHER AUERBACH PUBLICATIONS

Chaos Applications in Telecommunications
Peter Stavroulakis
ISBN: 0-8493-3832-8

Contemporary Coding Techniques and Applications for Mobile Communications
Onur Osman and Osman Nuri Ucan
ISBN: 1-4200-5461-9

Design Science Research Methods and Patterns: Innovating Information and Communication Technology
Vijay K. Vaishnavi and William Kuechler Jr.
ISBN: 1-4200-5932-7

Embedded Linux System Design and Development
P. Raghavan, Amol Lad and Sriram Neelakandan
ISBN: 0-8493-4058-6

Enhancing Computer Security with Smart Technology
V. Rao Vemuri
ISBN: 0-8493-3045-9

Graph Theory and Interconnection Networks
Lih-Hsing Hsu; Cheng-Kuan Lin
ISBN: 1-4200-4481-8

Modeling Software with Finite State Machines: A Practical Approach
Ferdinand Wagner, Ruedi Schmuki, Thomas Wagner and Peter Wolstenholme
ISBN: 0-8493-8086-3

MEMS and Nanotechnology-Based Sensors and Devices for Communications, Medical and Aerospace Applications
A. R. Jha
ISBN: 0-8493-8069-3

Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition
Sandhya Samarasinghe
ISBN: 0-8493-3375-X

Patterns for Performance and Operability: Building and Testing Enterprise Software
Chris Ford, Ido Gileadi, Sanjiv Purba and Mike Moerman
ISBN: 1-4200-5334-5

Software Engineering Foundations: A Software Science Perspective
Yingxu Wang
ISBN: 0-8493-1931-5

Systemic Yoyos: Some Impacts of the Second Dimension
Yi Lin
ISBN: 1-4200-8820-3

AUERBACH PUBLICATIONS

www.auerbach-publications.com
To Order Call: 1-800-272-7737 • Fax: 1-800-374-3401
E-mail: orders@crcpress.com

Preface

Since the origin of graph theory with Euler, people have liked graph theory, because it is a delightful playground for the exploration of proof techniques without much previous background needed. With the application of graph theory to many areas of computing, social, and natural sciences, the importance of graph theory has been recognized. For this reason, many good graph theoretic results have been developed in recent years. It is almost impossible to write a book that covers all these good graph theories.

The architecture of an interconnection network is always represented by a graph, where vertices represent processors and edges represent links between processors. It is almost impossible to design a network that is optimum from all aspects. One has to design a suitable network depending on its properties and requirements. Thus, many graphs are proposed as possible interconnection network topologies. Thus, these graphs can be referred to as good graphs. For this reason, the theory of interconnection networks is referred to as *good-graph theory*.

Thus, we need some basic background in graph theory to study interconnection networks. The first 10 chapters cover those materials presented in most graph theory texts and add some concepts of interconnection networks. Later, we discuss some interesting properties of interconnection networks. Sometimes, these properties are too good to be true—so the beginner finds them hard to believe. Such properties can be observed from interconnection networks, diameter, connectivity, Hamiltonian properties, and diagnosis properties. We can also study these properties in general graphs.

Ends of proofs are marked with the symbol □. This symbol can be found directly following a formal assertion. It means that the proof should be clear after what has been said. There are also some theorems that are stated, without proof, via background information. In this case, such theorems are stated without proofs and the symbol □.

The main purpose of this book is to share our research experience. Our experience tells us that much background is not needed for doing research. Looking for a proper theme is a difficult problem for research. We are just lucky in finding problems. For this reason, we put some of our results at the end of each chapter but we do not put any exercises in this book. We observed that all the material, contents, and exercises in every book are exercises of previous books. For this reason, we hope that the readers can find their own exercises.

It is a great pleasure to acknowledge the work of Professor Jimmy J.M. Tan. We have worked together for a long time. Yet, at least one-third of this book is his contribution. We also thank all the members of the Computer Theory Laboratory of National Chiao-Tung University: T.Y. Sung, T. Liang, H.M. Huang, Chiuyuan Chen, Y.C. Chuang, S. S. Kao, Y.C. Chang, T.Y. Ho, R.S. Chou, C.N. Hung, J.J. Wang, S.Y. Wang, C.Y. Lin, C.H. Tsai, J.Y. Hwang, C.P. Chang, J.N. Wang, J.J. Sheu, C.H. Chang, T.K. Li, W.T. Huang, Y.C. Chen, H.C. Hsu, K.Y. Liang, Y.L. Hsieh, M.C. Yang, P.L. Lai, L.C. Chiang, Y.H. Teng, C.M. Sun, and K.M. Hsu.

We also thank CRC Press for offering us the opportunity of publishing this book. Lih-Hsing Hsu also thanks his advisor, Professor Joel Spencer, for entering the area of graph theory. We also thank Professor Frank Hsu for strongly recommending to us the area of interconnection networks. Finally, we thank all of our teachers; without their encouragement, this book could never have been written.

Authors

Lih-Hsing Hsu received his BS in mathematics from Chung Yuan Christian University, Taiwan, Republic of China, in 1975, and his PhD in mathematics from the State University of New York at Stony Brook in 1981. From 1981 to 1985, he was an associate professor at the Department of Applied Mathematics at National Chiao Tung University in Taiwan. From 1985 to 2006, he was a professor at National Chiao Tung University. From 1985 to 1988, he was the chairman of the Department of Applied Mathematics at National Chiao Tung University. After 1988, he joined the Department of Computer and Information Science of National Chiao Tung University. In 2004, he retired from National Chiao Tung University, holding a title as an honorary scholar of that university. He is currently the chairman of the Department of Computer Science and Information Engineering, Providence University, Taichung, Taiwan. His research interests include interconnection networks, algorithms, graph theory, and VLSI layout.

Cheng-Kuan Lin received his BS in applied mathematics from Chinese Culture University, Taiwan, Republic of China, in 2000, and his MS in mathematics from the National Central University in Taiwan in 2002. His research interests include interconnection networks, algorithms, and graph theory.

Contents

1 Fundamental Concepts

1.1 GRAPHS AND SIMPLE GRAPHS

A *graph* G with n vertices and m edges consists of the *vertex set* $V(G) = \{v_1, v_2, \ldots, v_n\}$ and *edge set* $E(G) = \{e_1, e_2, \ldots, e_m\}$, where each edge consists of two (possibly equal) vertices called, *endpoints*. An element in $V(G)$ is called a *vertex* of G. An element in $E(G)$ is called an *edge* of G. Because graph theory has a variety of applications, we may also use a *node* for a vertex and a *link* for an edge to fit the common terminology in that area. We use the *unordered pair* (u, v) for an edge $e = \{u, v\}$. If $(u, v) \in E(G)$, then u and v are *adjacent*. We write $u \leftrightarrow v$ to mean *u is adjacent to v*. A *loop* is an edge whose endpoints are equal. *Parallel edges* or *multiple edges* are edges that have the same pair of endpoints. A *simple graph* is a graph having no loops or multiple edges. For a graph $G = (V, E)$, the *underlying simple graph* UG is the simple graph with vertex V and $(x, y) \in E(UG)$ if and only if $x \neq y$ and $(x, y) \in E$. A graph is *finite* if its vertex set and edge set are finite. *We adopt the convention that every graph mentioned is finite.*

We illustrate a graph on paper by assigning a point to each vertex and drawing a curve for each edge between the points representing its endpoints, sometimes omitting the name of the vertices or edges. In Figure 1.1a, e is a loop and f and g are parallel edges. Figures 1.1b and 1.1c illustrate two ways of drawing one simple graph.

The *order* of a graph G, written $n(G)$, is the number of vertices in G. An *n-vertex graph* is a graph of order n. The *size* of a graph G, written $e(G)$, is the number of edges in G, even though we also use e by itself to denote an edge. The *degree* of a vertex v in a graph, written $\deg_G(v)$, or $\deg(v)$, is the number of nonloop edges containing v plus twice the number of loops containing v. The *maximum degree* of G, denoted by $\Delta(G)$, is $\max\{\deg(v) \mid v \in V(G)\}$ and the *minimum degree* of G, denoted by $\delta(G)$, is $\min\{\deg(v) \mid v \in V(G)\}$. A graph G is *regular* if $\Delta(G) = \delta(G)$, and *k-regular* if $\Delta(G) = \delta(G) = k$. A vertex of degree k is *k-valent*. The *neighborhood* of v, written $N_G(v)$ or $N(v)$, is $\{x \in V(G) \mid x \leftrightarrow v\}$; x is a *neighbor* of v if $x \in N(v)$. An *isolated vertex* has a degree of 0.

A *directed graph* or *digraph* G consists of a *vertex set* $V(G)$ and an *edge set* $E(G)$, where each edge is an ordered pair of vertices. We use the *ordered pair* (u, v) for the edge uv, where u is the *tail* and v is the *head*. (Note that the notation (u, v) is used both as an unordered pair in an unordered graph and as an ordered pair in a directed graph. However, it is easy to distinguish each from the other from the context.) Sometimes, we may use the term *arc* for an edge of a digraph. We write $u \rightarrow v$ when $(u, v) \in E(G)$, meaning *there is an edge from u to v*. A *simple digraph* is a digraph in which each ordered pair of vertices occur at most once as an edge. For a digraph $G = (V, E)$, the

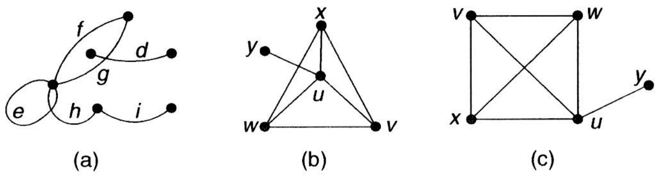


FIGURE 1.1 Examples of graphs.

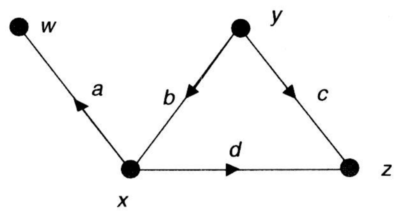


FIGURE 1.2 A digraph.

underlying graph UG is the simple graph with the vertex set V and $(x, y) \in E(UG)$ if and only if $x \neq y$ and either $(x, y) \in E(G)$ or $(y, x) \in E(G)$.

The choice of head and tail assigns a *direction* to an edge, which we illustrate by assigning edges as arrows. See Figure 1.2 for an example of a digraph. Sometimes, weights are assigned to the edges of graphs or digraphs. Thus, we have *weighted graphs*, *weighted digraphs*, *(unweighted) graphs*, and *(unweighted) digraphs*.

Graphs arise in many settings, which suggests useful concepts and terminologies about the structure of graphs.

Example 1.1

A famous brain-teaser asks: Does every set of six people have three mutual acquaintances or three mutual strangers?

Because an *acquaintance* is symmetric, we can model it by a simple graph having a vertex for each person and an edge for each acquainted pair. The *nonacquaintance* relation on the same set yields another graph. The *complement* $\overline{G}$ of a simple graph G with the same vertex set $V(G)$ is defined by $(u, v) \in E(\overline{G})$ if and only if $(u, v) \notin E(G)$. In Figure 1.3, we draw a graph and its complement.

Let two graphs equal $G = (V(G), E(G))$ and $H = (V(H), E(H))$. We say that H is a *subgraph* of G or G is a *supergraph* of H if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$; we write $H \subseteq G$ and say that G *contains* H. In particular, H is called a *spanning subgraph* of G or G is a *spanning supergraph* of H if H is a subgraph of G and $V(H) = V(G)$.

Let $G = (V, E)$ be a graph and S be a subset of V. The subgraph of G *induced* by S, denoted by $G[S]$, is the subgraph with vertex set S and those edges of G with both ends in S. In particular, the graph $G[V - S]$ is denoted by $G - S$. Let v be a vertex of G. We use $G - v$ to denote $G - \{v\}$. Let F be a subset of E. The subgraph generated

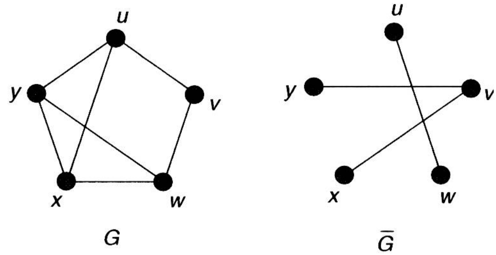


FIGURE 1.3 A graph and its complement.

by F, denoted by G_F, is the subgraph of G with F as its edge set and $\{v \mid v$ is incident with some edge of $F\}$ as its vertex set. The subgraph $G - F$ denotes the subgraph of G with V as its vertex set and $E - F$ as its edge set. Let e be an edge of E. The graph $G - \{e\}$ is denoted by $G - e$.

A *complete graph* or *clique* is a simple graph in which every pair of vertices is an edge. A complete graph has many subgraphs that are not cliques, but every induced subgraph of a complete graph is a clique. The complement of a complete graph has no edges.

An *independent set* in a graph G is a vertex set $S \subseteq V(G)$ that contains no edge of G (that is, $G[S]$ has no edges). In the graph G shown in Figure 1.3, the largest clique and the largest independent set have cardinalities three and two, respectively. These values reverse in the complement $\overline{G}$, because cliques become independent sets (and vice versa) under complementation. Our six-person brain-teaser asks whether every six-vertex graph has a clique or an independent set of size 3. It is worthwhile to verify this statement. The generalization of the six-person brain-teaser leads the Ramsey Theory [128].

EXAMPLE 1.2

Suppose that we have m jobs and n people, and each person can do some of the jobs. Can we make assignments to fill the jobs? We model the available assignments by a graph H having a vertex for each job and each person, putting job j adjacent to person p if p can do j.

A graph G is *bipartite* if $V(G)$ is the union of two disjoint sets such that each edge consists of one vertex from each set. The graph H of available assignments is bipartite, with the two sets being the people and the jobs. Because a person can do only one job and we assign a job to only one person, we seek m pairwise disjoint edges in H.

A *complete bipartite graph*, illustrated in Figure 1.4, is a bipartite graph whose edge set consists of all pairs having a vertex set from each of two disjoint sets covering the vertices. When the assignment graph H is a complete bipartite graph, pairing up vertices is easy, so we seek the *best* way to do it. We can assign numerical weights on the edges to measure desirability. The best way to match up the vertices is often the one where the selected edges have maximum total weight.