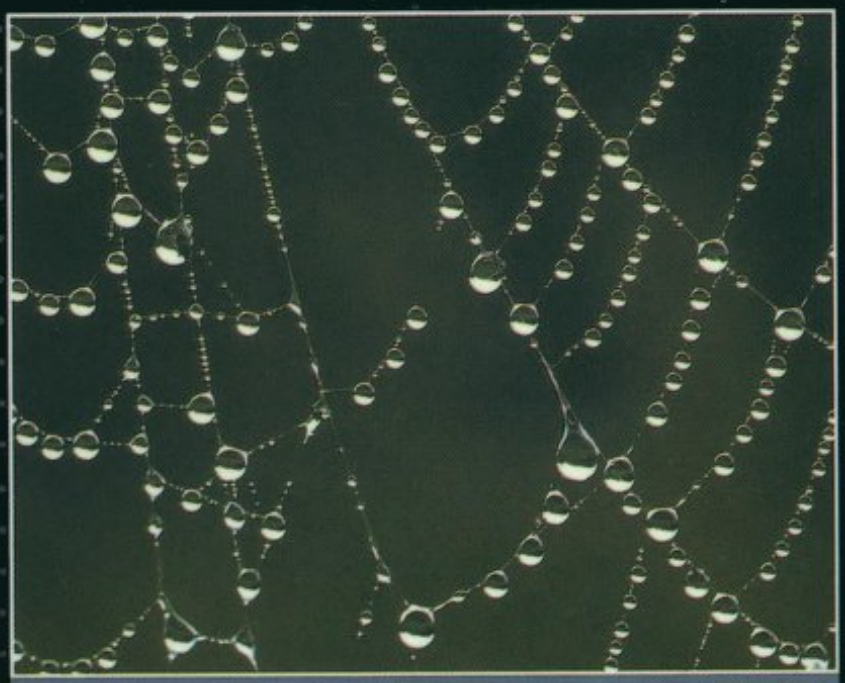
NETWORK FLOWS



THEORY, ALGORITHMS, AND APPLICATIONS

RAVINDRA K. AHUJA
THOMAS L. MAGNANTI
JAMES B. ORLIN

NETWORK FLOWS

Theory, Algorithms, and Applications

RAVINDRA K. AHUJA
Department of Industrial & Management Engineering
Indian Institute of Technology, Kanpur

THOMAS L. MAGNANTI
Sloan School of Management
Massachusetts Institute of Technology, Cambridge

JAMES B. ORLIN
Sloan School of Management
Massachusetts Institute of Technology, Cambridge

PRENTICE HALL, Upper Saddle River, New Jersey 07458

Library of Congress Cataloging-in-Publication Data

Ahuja, Ravindra K. (date)
Network flows : theory, algorithms, and applications / Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-617549-X
1. Network analysis (Planning) 2. Mathematical optimization. I. Magnanti, Thomas L. II. Orlin, James B., (date). III. Title.
T57.85.A37 1993
658.4'032—dc20 92-26702
CIP

Acquisitions editor: Pete Janzow
Production editor: Merrill Peterson
Cover designer: Design Source
Prepress buyer: Linda Behrens
Manufacturing buyer: David Dickey
Editorial assistant: Phyllis Morgan

Printed in the United States of America

16 17 18 19

ISBN 0-13-617549-X

PRENTICE-HALL INTERNATIONAL (UK) LIMITED, *London*
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, *Sydney*
PRENTICE-HALL CANADA INC., *Toronto*
PRENTICE-HALL HISPANOAMERICANA, S.A., *Mexico*
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*
EDITORA PRENTICE-HALL DO BRASIL, LTDA., *Rio de Janeiro*

Ravi dedicates this book to his spiritual master,
Revered Sri Ranaji Saheb.

Tom dedicates this book to his favorite network,
Beverly and Randy.

Jim dedicates this book to Donna,
who inspired him in so many ways.

Collectively, we offer this book as a tribute
to Lester Ford and Ray Fulkerson, whose pioneering research
and seminal text in network flows have been an enduring
inspiration to us and to a generation
of researchers and practitioners.

CONTENTS

PREFACE, xi

1 *INTRODUCTION, 1*

1.1 Introduction, *1*
1.2 Network Flow Problems, *4*
1.3 Applications, *9*
1.4 Summary, *18*
Reference Notes, *19*
Exercises, *20*

2 *PATHS, TREES, AND CYCLES, 23*

2.1 Introduction, *23*
2.2 Notation and Definitions, *24*
2.3 Network Representations, *31*
2.4 Network Transformations, *38*
2.5 Summary, *46*
Reference Notes, *47*
Exercises, *47*

3 *ALGORITHM DESIGN AND ANALYSIS, 53*

3.1 Introduction, *53*
3.2 Complexity Analysis, *56*
3.3 Developing Polynomial-Time Algorithms, *66*
3.4 Search Algorithms, *73*
3.5 Flow Decomposition Algorithms, *79*
3.6 Summary, *84*
Reference Notes, *85*
Exercises, *86*

4 *SHORTEST PATHS: LABEL-SETTING ALGORITHMS, 93*

4.1 Introduction, *93*
4.2 Applications, *97*
4.3 Tree of Shortest Paths, *106*
4.4 Shortest Path Problems in Acyclic Networks, *107*
4.5 Dijkstra's Algorithm, *108*
4.6 Dial's Implementation, *113*
4.7 Heap Implementations, *115*
4.8 Radix Heap Implementation, *116*

4.9 Summary, *121*
Reference Notes, *122*
Exercises, *124*

5 *SHORTEST PATHS: LABEL-CORRECTING ALGORITHMS, 133*

5.1 Introduction, *133*
5.2 Optimality Conditions, *135*
5.3 Generic Label-Correcting Algorithms, *136*
5.4 Special Implementations of the Modified Label-Correcting Algorithm, *141*
5.5 Detecting Negative Cycles, *143*
5.6 All-Pairs Shortest Path Problem, *144*
5.7 Minimum Cost-to-Time Ratio Cycle Problem, *150*
5.8 Summary, *154*
Reference Notes, *156*
Exercises, *157*

6 *MAXIMUM FLOWS: BASIC IDEAS, 166*

6.1 Introduction, *166*
6.2 Applications, *169*
6.3 Flows and Cuts, *177*
6.4 Generic Augmenting Path Algorithm, *180*
6.5 Labeling Algorithm and the Max-Flow Min-Cut Theorem, *184*
6.6 Combinatorial Implications of the Max-Flow Min-Cut Theorem, *188*
6.7 Flows with Lower Bounds, *191*
6.8 Summary, *196*
Reference Notes, *197*
Exercises, *198*

7 *MAXIMUM FLOWS: POLYNOMIAL ALGORITHMS, 207*

7.1 Introduction, *207*
7.2 Distance Labels, *209*
7.3 Capacity Scaling Algorithm, *210*
7.4 Shortest Augmenting Path Algorithm, *213*
7.5 Distance Labels and Layered Networks, *221*
7.6 Generic Preflow-Push Algorithm, *223*
7.7 FIFO Preflow-Push Algorithm, *231*
7.8 Highest-Label Preflow-Push Algorithm, *233*
7.9 Excess Scaling Algorithm, *237*
7.10 Summary, *241*
Reference Notes, *241*
Exercises, *243*

8 *MAXIMUM FLOWS: ADDITIONAL TOPICS, 250*

8.1 Introduction, *250*
8.2 Flows in Unit Capacity Networks, *252*
8.3 Flows in Bipartite Networks, *255*
8.4 Flows in Planar Undirected Networks, *260*
8.5 Dynamic Tree Implementations, *265*

8.6 Network Connectivity, *273*
8.7 All-Pairs Minimum Value Cut Problem, *277*
8.8 Summary, *285*
Reference Notes, *287*
Exercises, *288*

9 *MINIMUM COST FLOWS: BASIC ALGORITHMS, 294*

9.1 Introduction, *294*
9.2 Applications, *298*
9.3 Optimality Conditions, *306*
9.4 Minimum Cost Flow Duality, *310*
9.5 Relating Optimal Flows to Optimal Node Potentials, *315*
9.6 Cycle-Canceling Algorithm and the Integrality Property, *317*
9.7 Successive Shortest Path Algorithm, *320*
9.8 Primal–Dual Algorithm, *324*
9.9 Out-of-Kilter Algorithm, *326*
9.10 Relaxation Algorithm, *332*
9.11 Sensitivity Analysis, *337*
9.12 Summary, *339*
Reference Notes, *341*
Exercises, *344*

10 *MINIMUM COST FLOWS: POLYNOMIAL ALGORITHMS, 357*

10.1 Introduction, *357*
10.2 Capacity Scaling Algorithm, *360*
10.3 Cost Scaling Algorithm, *362*
10.4 Double Scaling Algorithm, *373*
10.5 Minimum Mean Cycle-Canceling Algorithm, *376*
10.6 Repeated Capacity Scaling Algorithm, *382*
10.7 Enhanced Capacity Scaling Algorithm, *387*
10.8 Summary, *395*
Reference Notes, *396*
Exercises, *397*

11 *MINIMUM COST FLOWS: NETWORK SIMPLEX ALGORITHMS, 402*

11.1 Introduction, *402*
11.2 Cycle Free and Spanning Tree Solutions, *405*
11.3 Maintaining a Spanning Tree Structure, *409*
11.4 Computing Node Potentials and Flows, *411*
11.5 Network Simplex Algorithm, *415*
11.6 Strongly Feasible Spanning Trees, *421*
11.7 Network Simplex Algorithm for the Shortest Path Problem, *425*
11.8 Network Simplex Algorithm for the Maximum Flow Problem, *430*
11.9 Related Network Simplex Algorithms, *433*
11.10 Sensitivity Analysis, *439*
11.11 Relationship to Simplex Method, *441*
11.12 Unimodularity Property, *447*
11.13 Summary, *450*
Reference Notes, *451*
Exercises, *453*

12 ***ASSIGNMENTS AND MATCHINGS, 461***

12.1 Introduction, *461*
12.2 Applications, *463*
12.3 Bipartite Cardinality Matching Problem, *469*
12.4 Bipartite Weighted Matching Problem, *470*
12.5 Stable Marriage Problem, *473*
12.6 Nonbipartite Cardinality Matching Problem, *475*
12.7 Matchings and Paths, *494*
12.8 Summary, *498*
Reference Notes, *499*
Exercises, *501*

13 ***MINIMUM SPANNING TREES, 510***

13.1 Introduction, *510*
13.2 Applications, *512*
13.3 Optimality Conditions, *516*
13.4 Kruskal's Algorithm, *520*
13.5 Prim's Algorithm, *523*
13.6 Sollin's Algorithm, *526*
13.7 Minimum Spanning Trees and Matroids, *528*
13.8 Minimum Spanning Trees and Linear Programming, *530*
13.9 Summary, *533*
Reference Notes, *535*
Exercises, *536*

14 ***CONVEX COST FLOWS, 543***

14.1 Introduction, *543*
14.2 Applications, *546*
14.3 Transformation to a Minimum Cost Flow Problem, *551*
14.4 Pseudopolynomial-Time Algorithms, *554*
14.5 Polynomial-Time Algorithm, *556*
14.6 Summary, *560*
Reference Notes, *561*
Exercises, *562*

15 ***GENERALIZED FLOWS, 566***

15.1 Introduction, *566*
15.2 Applications, *568*
15.3 Augmented Forest Structures, *572*
15.4 Determining Potentials and Flows for an Augmented Forest Structure, *577*
15.5 Good Augmented Forests and Linear Programming Bases, *582*
15.6 Generalized Network Simplex Algorithm, *583*
15.7 Summary, *591*
Reference Notes, *591*
Exercises, *593*

16 ***LAGRANGIAN RELAXATION AND NETWORK OPTIMIZATION, 598***

16.1 Introduction, *598*
16.2 Problem Relaxations and Branch and Bound, *602*
16.3 Lagrangian Relaxation Technique, *605*
16.4 Lagrangian Relaxation and Linear Programming, *615*
16.5 Applications of Lagrangian Relaxation, *620*
16.6 Summary, *635*
Reference Notes, *637*
Exercises, *638*

17 ***MULTICOMMODITY FLOWS, 649***

17.1 Introduction, *649*
17.2 Applications, *653*
17.3 Optimality Conditions, *657*
17.4 Lagrangian Relaxation, *660*
17.5 Column Generation Approach, *665*
17.6 Dantzig–Wolfe Decomposition, *671*
17.7 Resource-Directive Decomposition, *674*
17.8 Basis Partitioning, *678*
17.9 Summary, *682*
Reference Notes, *684*
Exercises, *686*

18 ***COMPUTATIONAL TESTING OF ALGORITHMS, 695***

18.1 Introduction, *695*
18.2 Representative Operation Counts, *698*
18.3 Application to Network Simplex Algorithm, *702*
18.4 Summary, *713*
Reference Notes, *713*
Exercises, *715*

19 ***ADDITIONAL APPLICATIONS, 717***

19.1 Introduction, *717*
19.2 Maximum Weight Closure of a Graph, *719*
19.3 Data Scaling, *725*
19.4 Science Applications, *728*
19.5 Project Management, *732*
19.6 Dynamic Flows, *737*
19.7 Arc Routing Problems, *740*
19.8 Facility Layout and Location, *744*
19.9 Production and Inventory Planning, *748*
19.10 Summary, *755*
Reference Notes, *759*
Exercises, *760*

APPENDIX A ***DATA STRUCTURES,*** ***765***

A.1 Introduction, *765*
A.2 Elementary Data Structures, *766*
A.3 *d*-Heaps, *773*
A.4 Fibonacci Heaps, *779*
Reference Notes, *787*

APPENDIX B ***$\mathcal{NP}$-COMPLETENESS,*** **788**

B.1 Introduction, *788*
B.2 Problem Reductions and Transformations, *790*
B.3 Problem Classes $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NP}$-Complete, and $\mathcal{NP}$-Hard, *792*
B.4 Proving $\mathcal{NP}$-Completeness Results, *796*
B.5 Concluding Remarks, *800*
Reference Notes, *801*

APPENDIX C ***LINEAR PROGRAMMING,*** ***802***

C.1 Introduction, *802*
C.2 Graphical Solution Procedure, *804*
C.3 Basic Feasible Solutions, *805*
C.4 Simplex Method, *810*
C.5 Bounded Variable Simplex Method, *814*
C.6 Linear Programming Duality, *816*
Reference Notes, *820*

REFERENCES, ***821***

INDEX, ***840***

PREFACE

If you would not be forgotten,
As soon as you are dead and rotten,
Either write things worthy reading,
Or do things worth the writing.
—Benjamin Franklin

Network flows is an exciting field that brings together what many students, practitioners, and researchers like best about the mathematical and computational sciences. It couples deep intellectual content with a remarkable range of applicability, covering literally thousands of applications in such wide-ranging fields as chemistry and physics, computer networking, most branches of engineering, manufacturing, public policy and social systems, scheduling and routing, telecommunications, and transportation. It is classical, dating from the work of Gustav Kirchhoff and other eminent physical scientists of the last century, and yet vibrant and current, bursting with new results and new approaches. Its heritage is rooted in the traditional fields of mechanics, engineering, and applied mathematics as well as the contemporary fields of computer science and operations research.

In writing this book we have attempted to capture these varied perspectives and in doing so to fill a need that we perceived for a comprehensive text on network flows that would bring together the old and the new, and provide an integrative view of theory, algorithms, and applications. We have attempted to design a book that could be used either as an introductory or advanced text for upper-level undergraduate or graduate students or as a reference for researchers and practitioners. We have also strived to make the coverage of this material as readable, accessible, and insightful as possible, particularly for readers with a limited background in computer science and optimization.

The book has the following features:

- In-depth and self-contained treatment of shortest path, maximum flow, and minimum cost flow problems, including descriptions of new and novel polynomial-time algorithms for these core models.
- Emphasis on powerful algorithmic strategies and analysis tools, such as data scaling, geometric improvement arguments, and potential function arguments.
- An easy-to-understand description of several important data structures, including *d*-heaps, Fibonacci heaps, and dynamic trees.
- Treatment of other important topics in network optimization and of practical solution techniques such as Lagrangian relaxation.

- Each new topic introduced by a set of applications and an entire chapter devoted to applications.
- A special chapter devoted to conducting empirical testing of algorithms.
- Over 150 applications of network flows to a variety of engineering, management, and scientific domains.
- Over 800 exercises that vary in difficulty, including many that develop extensions of material covered in the text.
- Approximately 400 figures that illustrate the material presented in the text.
- Extensive reference notes that provide readers with historical contexts and with guides to the literature.

As indicated by this list, we have not attempted to cover all topics at the same level of depth, but instead, have treated certain core topics more extensively than others. Moreover, although we have focused on the design and analysis of efficient algorithms, we have also placed considerable emphasis on applications.

In attempting to streamline the material in this book and present it in an integrated fashion, we have devoted considerable time, and in several cases conducted research, to improve our presentation. As a result, our coverage of some topics differs from the discussion in the current literature. We hope that this approach will not lead to any confusion, and that it will, in fact, promote a better understanding of the material and uncover new connections between topics that might not appear to be so related in the literature.

TO INSTRUCTORS AND STUDENTS

We have attempted to write this book so that it is accessible to students with many backgrounds. Although students require some mathematical maturity—for example, a basic understanding of proof techniques—and some familiarity with computer programming, they need not be specialists in mathematics, computer science, or optimization. Some basic knowledge of these topics would undoubtedly prove to be useful in using this book. In Chapter 3 and Appendices A, B, and C, we have provided some of this general background.

The book contains far more material than anyone could possibly cover in a one-semester course. Chapters 1 to 3, 19, selected material from Chapters 5 to 12, 16, and 17, and portions of Chapters 13 to 15 and 18 would serve as a broad-based course on network flows and network optimization. Because the chapters are generally modular, instructors might use the introductory material in the first few sections of each chapter as well as a selection of additional material for chapters that they would not want to cover in their entirety. An advanced course on algorithms might focus on Chapter 4 to 12, covering the material in these chapters in their entirety.

In teaching the algorithms in this book, we feel that it is important to understand the underlying methods of algorithm design and analysis as well as specific results. Therefore, we encourage both instructors and students to refer frequently back to Chapter 3 and its discussion of algorithm design and analysis.

Many of the topics that we have examined in this book are specially structured

linear programs. Therefore, we could have adopted a linear programming approach while presenting much of the material. Instead, with the exception of Chapter 17 and parts of Chapters 15 and 16, we have argued almost exclusively from first principles and adopted a network or graphical viewpoint. We believe that this approach, while occasionally imposing a less streamlined development than might be possible using linear programming, offers several advantages. First, the material is readily accessible to a wider audience. Second, this approach permits students who are not optimization specialists to learn many of the ideas of linear programming in a concrete setting with easy geometric and algebraic interpretations; it also permits students with prior knowledge of linear programming to refine their understanding by seeing material in a different light. In fact, when the audience for a course has a background in linear programming, we would encourage instructors and students to make explicit connections between our coverage and more general results in linear programming.

Although we have included some numerical exercises that test basic understanding of material presented in the text, many of the exercises address applications or theory. Instructors might like to use this material, with suitable amplification, as lecture material. Instructors wishing more numerical examples might modify the ones we have provided.

TO OUR GENERAL READERS

Professionals in applied mathematics, computer science, engineering, and management science/operations research as well as practitioners in a variety of application domains might wish to extract specific information from the text without covering the book in detail. Since the book is organized primarily by model type (e.g., shortest path or spanning tree problems), readers have ready access to the material along these dimensions. For a guide to applications, readers might consult Section 19.10, which contains a set of tables summarizing the various network flow applications that we have considered in the text. The end-of-chapter reference notes also contain references to a number of applications that we have either not discussed or considered only in the exercises. For the most part, with the exception of applications, we have been selective in our citations to the literature. Many of the general references that we have mentioned at the end of Chapter 1 contain more detailed references to the literature.

We have described many algorithms in this book using a pseudocode that should be understandable to readers with any passing familiarity with computer programming. This approach provides us with a certain degree of universality in describing algorithms, but also requires that those wishing to use the algorithms must translate them into specific programming languages and add material such as input/output and error handling procedures that will be implementation dependent.

FEEDBACK

Any book of this size and complexity will undoubtedly contain errors; moreover, in writing this book, we might have inadvertently not given proper credit to everyone deserving recognition for specific results. We would be pleased to learn about any

comments that you might have about this book, including errors that you might find. Please direct any feedback as follows:

Professor James B. Orlin
Sloan School of Management, MIT
Cambridge, MA 02139, USA
e-mail: jorlin@eagle.mit.edu
fax: 617-258-7579

ACKNOWLEDGMENTS

Many individuals have contributed to this text by enhancing our understanding of network flows, by advising us in general about the book's content, or by providing constructive feedback on early drafts.

We owe an intellectual debt to several individuals. From direct collaborations or through the writings of E. Dinic, Jack Edmonds, Hal Gabow, Fred Glover, Matsao Iri, Bruce Golden, Richard Karp, A. Karzanov, Darwin Klingman, Eugene Lawler, Robert Tarjan, Eva Tardos, Richard Wong, and so many others, we have learned much about the general topic of networks and network optimization; the pioneering efforts of George Dantzig, Lester Ford, and Delbert Fulkerson in the 1950s defined the field of network flows as we know it today. We hope that our treatment of the subject is true to the spirit of these individuals. Many of our colleagues, too numerous to mention by name, responded to a questionnaire that we distributed soliciting advice on the coverage for this book. Their comments were very helpful in refining our ideas about the book's overall design. Anant Balakrishnan and Janny Leung read and commented on portions of the manuscript; S. K. Gupta, Leslie Hall, Prakash Mirchandani, and Steffano Pallottino each commented in detail about large segments of the manuscript. We are especially indebted to Steffano Pallottino for several careful reviews of the manuscript and for identifying numerous corrections. Bill Cunningham offered detailed suggestions on an earlier book chapter that served as the starting point for this book. Each of these individual's advice has greatly improved our final product. Over the years, many of our doctoral students have helped us to test and refine our ideas. Several recent students—including Murali Kodialam, Yusin Lee, Tim Magee, S. Raghavan, Rina Schneur, and Jim Walton—have helped us in developing exercises. These students and many others in our classes have discovered errors and offered constructive criticisms on the manuscript. In this regard, we are particularly grateful to the Spring 1991 class at MIT in Network Optimization. Thanks also to Prentice Hall reviewer Leslie Hall, Princeton University. Charu Aggarwal and Ajay Mishra also provided us valuable assistance in debugging the book, proofreading it, and preparing the index; our special thanks go to them.

Ghanshyam Hoshing (I.I.T., Kanpur) and Karen Martel and Laura Terrell (both at M.I.T., Cambridge) each did a superb job in typing portions of the manuscript. Ghanshyam Hoshing deserves much credit for typing/drawing and editing most of the text and figures.

We are indebted to the Industrial and Management Engineering Department at I.I.T., Kanpur and to the Sloan School of Management and Operations Research Center at M.I.T. for providing us with an environment conducive to conducting

research in network flows and to writing this book. We are also grateful to the National Science Foundation, the Office of Naval Research, the Department of Transportation, and GTE Laboratories for supporting our research that underlies much of this book.

We'd like to acknowledge our parents, Kailash and Ganesh Das Ahuja, Florence and Lee Magnanti, and Roslyn and Albert Orlin, for their affection and encouragement and for instilling in us a love for learning. Finally, we offer our heartfelt thanks to our wives—Smita Ahuja, Beverly Magnanti, Donna Orlin—and our children—Saumya and Shaman Ahuja; Randy Magnanti; and Jenna, Ben, and Caroline Orlin—for their love and understanding as we wrote this book. This book started as a far less ambitious project and so none of our families could possibly have realized how much of our time and energies we would be wresting from them as we wrote it. This work is as much theirs as it is ours!

Kanpur and Cambridge

R. K. Ahuja
T. L. Magnanti
J. B. Orlin

1

INTRODUCTION

Begin at the beginning . . . and go on till you come to the end: then stop.
—Lewis Carroll

Chapter Outline

1.1 Introduction
1.2 Network Flow Problems
1.3 Applications
1.4 Summary

1.1 INTRODUCTION

Everywhere we look in our daily lives, networks are apparent. Electrical and power networks bring lighting and entertainment into our homes. Telephone networks permit us to communicate with each other almost effortlessly within our local communities and across regional and international borders. National highway systems, rail networks, and airline service networks provide us with the means to cross great geographical distances to accomplish our work, to see our loved ones, and to visit new places and enjoy new experiences. Manufacturing and distribution networks give us access to life's essential foodstock and to consumer products. And computer networks, such as airline reservation systems, have changed the way we share information and conduct our business and personal lives.

In all of these problem domains, and in many more, we wish to move some entity (electricity, a consumer product, a person or a vehicle, a message) from one point to another in an underlying network, and to do so as efficiently as possible, both to provide good service to the users of the network and to use the underlying (and typically expensive) transmission facilities effectively. In the most general sense, this objective is what this book is all about. We want to learn how to model application settings as mathematical objects known as network flow problems and to study various ways (algorithms) to solve the resulting models.

Network flows is a problem domain that lies at the cusp between several fields of inquiry, including applied mathematics, computer science, engineering, management, and operations research. The field has a rich and long tradition, tracing its roots back to the work of Gustav Kirchhof and other early pioneers of electrical engineering and mechanics who first systematically analyzed electrical circuits. This early work set the foundations of many of the key ideas of network flow theory and established networks (graphs) as useful mathematical objects for representing many

physical systems. Much of this early work was descriptive in nature, answering such questions as: If we apply a set of voltages to a given network, what will be the resulting current flow? The set of questions that we address in this book are a bit different: If we have alternative ways to use a network (i.e., send flow), which alternative will be most cost-effective? Our intellectual heritage for answering such questions is much more recent and can be traced to the late 1940s and early 1950s when the research and practitioner communities simultaneously developed optimization as an independent field of inquiry and launched the computer revolution, leading to the powerful instruments we know today for performing scientific and managerial computations.

For the most part, in this book we wish to address the following basic questions:

1. *Shortest path problem.* What is the best way to traverse a network to get from one point to another as cheaply as possible?
2. *Maximum flow problem.* If a network has capacities on arc flows, how can we send as much flow as possible between two points in the network while honoring the arc flow capacities?
3. *Minimum cost flow problem.* If we incur a cost per unit flow on a network with arc capacities and we need to send units of a good that reside at one or more points in the network to one or more other points, how can we send the material at minimum possible cost?

In the sense of traditional applied and pure mathematics, each of these problems is trivial to solve. It is not very difficult (but not at all obvious for the later two problems) to see that we need only consider a finite number of alternatives for each problem. So a traditional mathematician might say that the problems are well solved: Simply enumerate the set of possible solutions and choose the one that is best. Unfortunately, this approach is far from pragmatic, since the number of possible alternatives can be very large—more than the number of atoms in the universe for many practical problems! So instead, we would like to devise algorithms that are in a sense "good," that is, whose computation time is small, or at least reasonable, for problems met in practice. One way to ensure this objective is to devise algorithms whose running time is guaranteed not to grow very fast as the underlying network becomes larger (the computer science, operations research, and applied mathematics communities refer to the development of algorithms with such performance guarantees as *worst-case analysis*). Developing algorithms that are good in this sense is another major theme throughout this book, and our development builds heavily on the theory of computational complexity that began to develop within computer science, applied mathematics, and operations research circles in the 1970s, and has flourished ever since.

The field of computational complexity theory combines both craftsmanship and theory; it builds on a confluence of mathematical insight, creative algorithm design, and the careful, and often very clever use of data structures to devise solution methods that are provably good in the sense that we have just mentioned. In the field of network flows, researchers devised the first, seminal contributions of this nature in the 1950s before the field of computational complexity theory even existed as a separate discipline as we know it today. And throughout the last three decades,

researchers have made a steady stream of innovations that have resulted in new solution methods and in improvements to known methods. In the past few years, however, researchers have made contributions to the design and analysis of network flow algorithms with improved worst-case performance guarantees at an explosive, almost dizzying pace; moreover, these contributions were very surprising: Throughout the 1950s, 1960s, and 1970s, network flows had evolved into a rather mature field, so much so that most of the research and practitioner communities believed that the core models that we study in this book were so very well understood that further innovations would be hard to come by and would be few and far between. As it turns out, nothing could have been further from the truth.

Our presentation is intended to reflect these new developments; accordingly, we place a heavy emphasis on designing and analyzing good algorithms for solving the core optimization models that arise in the context of network flows. Our intention is to bring together and synthesize the many new contributions concerning efficient network flow algorithms with traditional material that has evolved over the past four decades. We have attempted to distill and highlight some of the essential core ideas (e.g., scaling and potential function arguments) that underlie many of the recent innovations and in doing so to give a unified account of the many algorithms that are now available. We hope that this treatment will provide our readers not only with an accessible entrée to these exciting new developments, but also with an understanding of the most recent and advanced contributions from the literature. Although we are bringing together ideas and methodologies from applied mathematics, computer science, and operations research, our approach has a decidedly computer science orientation as applied to certain types of models that have traditionally arisen in the context of managing a variety of operational systems (the foodstuff of operations research).

We feel that a full understanding of network flow algorithms and a full appreciation for their use requires more than an in-depth knowledge of good algorithms for core models. Consequently, even though this topic is our central thrust, we also devote considerable attention to describing applications of network flow problems. Indeed, we feel that our discussion of applications throughout the text, in the exercises, and in a concluding chapter is one of the major distinguishing features of our coverage.

We have not adopted a linear programming perspective throughout the book, however, because we feel there is much to be gained from a more direct approach, and because we would like the material we cover to be readily accessible to readers who are not optimization specialists. Moreover, we feel that an understanding of network flow problems from first principles provides a useful concrete setting from which to draw considerable insight about more general linear programs.

Similarly, since several important variations of the basic network flow problems are important in practice, or in placing network flows in the broader context of the field of combinatorial optimization, we have also included several chapters on additional topics: assignments and matchings, minimum spanning trees, models with convex (instead of linear) costs, networks with losses and gains, and multicommodity flows. In each of these chapters we have not attempted to be comprehensive, but rather, have tried to provide an introduction to the essential ideas of the topics.

The Lagrangian relaxation chapter permits us to show how the core network

models arise in broader problem contexts and how the algorithms that we have developed for the core models can be used in conjunction with other methods to solve more complex problems that arise frequently in practice. In particular, this discussion permits us to introduce and describe the basic ideas of decomposition methods for several important network optimization models—constrained shortest paths, the traveling salesman problem, vehicle routing problem, multicommodity flows, and network design.

Since the proof of the pudding is in the eating, we have also included a chapter on some aspects of computational testing of algorithms. We devote much of our discussion to devising the best possible algorithms for solving network flow problems, in the theoretical sense of computational complexity theory. Although the theoretical model of computation that we are using has proven to be a valuable guide for modeling and predicting the performance of algorithms in practice, it is not a perfect model, and therefore algorithms that are not theoretically superior often perform best in practice. Although empirical testing of algorithms has traditionally been a valuable means for investigating algorithmic ideas, the applied mathematics, computer science, and operations research communities have not yet reached a consensus on how to measure algorithmic performance empirically. So in this chapter we not only report on computational experience with an algorithm we have presented, but also offer some thoughts on how to measure computational performance and compare algorithms.

1.2 NETWORK FLOW PROBLEMS

In this section we introduce the network flow models we study in this book, and in the next section we present several applications that illustrate the practical importance of these models. In both the text and exercises throughout the remaining chapters, we introduce many other applications. In particular, Chapter 19 contains a more comprehensive summary of applications with illustrations drawn from several specialties in applied mathematics, engineering, logistics, manufacturing, and the physical sciences.

Minimum Cost Flow Problem

The minimum cost flow model is the most fundamental of all network flow problems. Indeed, we devote most of this book to the minimum cost flow problem, special cases of it, and several of its generalizations. The problem is easy to state: We wish to determine a least cost shipment of a commodity through a network in order to satisfy demands at certain nodes from available supplies at other nodes. This model has a number of familiar applications: the distribution of a product from manufacturing plants to warehouses, or from warehouses to retailers; the flow of raw material and intermediate goods through the various machining stations in a production line; the routing of automobiles through an urban street network; and the routing of calls through the telephone system. As we will see later in this chapter and in Chapters 9 and 19, the minimum cost flow model also has many less transparent applications.

In this section we present a mathematical programming formulation of the minimum cost flow problem and then describe several of its specializations and

variants as well as other basic models that we consider in later chapters. We assume our readers are familiar with the basic notation and definitions of graph theory; those readers without this background might consult Section 2.2 for a brief account of this material.

Let $G = (N, A)$ be a directed network defined by a set N of n *nodes* and a set A of m *directed arcs*. Each arc $(i, j) \in A$ has an associated *cost* c_{ij} that denotes the cost per unit flow on that arc. We assume that the flow cost varies linearly with the amount of flow. We also associate with each arc $(i, j) \in A$ a *capacity* u_{ij} that denotes the maximum amount that can flow on the arc and a *lower bound* l_{ij} that denotes the minimum amount that must flow on the arc. We associate with each node $i \in N$ an integer number $b(i)$ representing its supply/demand. If $b(i) > 0$, node i is a *supply node*; if $b(i) < 0$, node i is a *demand node* with a demand of $-b(i)$; and if $b(i) = 0$, node i is a *transshipment node*. The decision variables in the minimum cost flow problem are arc flows and we represent the flow on an arc $(i, j) \in A$ by x_{ij}. The minimum cost flow problem is an optimization model formulated as follows:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1.1a}$$

subject to

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i) \qquad \text{for all } i \in N, \tag{1.1b}$$

$$l_{ij} \le x_{ij} \le u_{ij} \qquad \text{for all } (i,j) \in A, \tag{1.1c}$$

where $\sum_{i=1}^{n} b(i) = 0$. In matrix form, we represent the minimum cost flow problem as follows:

$$\text{Minimize } cx \tag{1.2a}$$

subject to

$$\mathcal{N}x = b, \tag{1.2b}$$

$$l \le x \le u. \tag{1.2c}$$

In this formulation, $\mathcal{N}$ is an $n \times m$ matrix, called the *node–arc incidence matrix* of the minimum cost flow problem. Each column $\mathcal{N}_{ij}$ in the matrix corresponds to the variable x_{ij}. The column $\mathcal{N}_{ij}$ has a $+1$ in the ith row, a -1 in the jth row; the rest of its entries are zero.

We refer to the constraints in (1.1b) as *mass balance constraints*. The first term in this constraint for a node represents the total *outflow* of the node (i.e., the flow emanating from the node) and the second term represents the total *inflow* of the node (i.e., the flow entering the node). The mass balance constraint states that the outflow minus inflow must equal the supply/demand of the node. If the node is a supply node, its outflow exceeds its inflow; if the node is a demand node, its inflow exceeds its outflow; and if the node is a transshipment node, its outflow equals its inflow. The flow must also satisfy the lower bound and capacity constraints (1.1c), which we refer to as *flow bound constraints*. The flow bounds typically model physical capacities or restrictions imposed on the flows' operating ranges. In most applications, the lower bounds on arc flows are zero; therefore, if we do not state lower bounds for any problem, we assume that they have value zero.

In most parts of the book we assume that the data are integral (i.e., all arc capacities, arc costs, and supplies/demands of nodes are integral). We refer to this assumption as the *integrality assumption*. The integrality assumption is not restrictive for most applications because we can always transform rational data to integer data by multiplying them by a suitably large number. Moreover, we necessarily need to convert irrational numbers to rational numbers to represent them on a computer.

The following special versions of the minimum cost flow problem play a central role in the theory and applications of network flows.

Shortest path problem. The shortest path problem is perhaps the simplest of all network flow problems. For this problem we wish to find a path of minimum cost (or length) from a specified *source node* s to another specified *sink node* t, assuming that each arc $(i, j) \in A$ has an associated cost (or length) c_{ij}. Some of the simplest applications of the shortest path problem are to determine a path between two specified nodes of a network that has minimum length, or a path that takes least time to traverse, or a path that has the maximum reliability. As we will see in our later discussions, this basic model has applications in many different problem domains, such as equipment replacement, project scheduling, cash flow management, message routing in communication systems, and traffic flow through congested cities. If we set $b(s) = 1$, $b(t) = -1$, and $b(\text{i}) = 0$ for all other nodes in the minimum cost flow problem, the solution to the problem will send 1 unit of flow from node s to node t along the shortest path. The shortest path problem also models situations in which we wish to send flow from a single-source node to a single-sink node in an uncapacitated network. That is, if we wish to send v units of flow from node s to node t and the capacity of each arc of the network is at least v, we would send the flow along a shortest path from node s to node t. If we want to determine shortest paths from the source node s to every other node in the network, then in the minimum cost flow problem we set $b(s) = (n - 1)$ and $b(i) = -1$ for all other nodes. [We can set each arc capacity u_{ij} to any number larger than $(n - 1)$.] The minimum cost flow solution would then send unit flow from node s to every other node i along a shortest path.

Maximum flow problem. The maximum flow problem is in a sense a complementary model to the shortest path problem. The shortest path problem models situations in which flow incurs a cost but is not restricted by any capacities; in contrast, in the maximum flow problem flow incurs no costs but is restricted by flow bounds. The maximum flow problem seeks a feasible solution that sends the maximum amount of flow from a specified source node s to another specified sink node t. If we interpret u_{ij} as the maximum flow rate of arc (i, j), the maximum flow problem identifies the maximum steady-state flow that the network can send from node s to node t per unit time. Examples of the maximum flow problem include determining the maximum steady-state flow of (1) petroleum products in a pipeline network, (2) cars in a road network, (3) messages in a telecommunication network, and (4) electricity in an electrical network. We can formulate this problem as a minimum cost flow problem in the following manner. We set $b(i) = 0$ for all $i \in N$, $c_{ij} = 0$ for all $(i, j) \in A$, and introduce an additional arc (t, s) with cost $c_{ts} = -1$ and flow bound $u_{ts} = \infty$. Then the minimum cost flow solution maximizes the flow on arc (t, s); but

since any flow on arc (t, s) must travel from node s to node t through the arcs in A [since each $b(i) = 0$], the solution to the minimum cost flow problem will maximize the flow from node s to node t in the original network.

Assignment problem. The data of the assignment problem consist of two equally sized sets N_1 and N_2 (i.e., $| N_1 | = | N_2 |$), a collection of pairs $A \subseteq N_1 \times N_2$ representing possible assignments, and a cost c_{ij} associated with each element $(i, j) \in A$. In the assignment problem we wish to pair, at minimum possible cost, each object in N_1 with exactly one object in N_2. Examples of the assignment problem include assigning people to projects, jobs to machines, tenants to apartments, swimmers to events in a swimming meet, and medical school graduates to available internships. The assignment problem is a minimum cost flow problem in a network $G = (N_1 \cup N_2, A)$ with $b(i) = 1$ for all $i \in N_1$, $b(i) = -1$ for all $i \in N_2$, and $u_{ij} = 1$ for all $(i, j) \in A$.

Transportation problem. The transportation problem is a special case of the minimum cost flow problem with the property that the node set N is partitioned into two subsets N_1 and N_2 (of possibly unequal cardinality) so that (1) each node in N_1 is a supply node, (2) each node N_2 is a demand node, and (3) for each arc (i,j) in A, $i \in N_1$ and $j \in N_2$. The classical example of this problem is the distribution of goods from warehouses to customers. In this context the nodes in N_1 represent the warehouses, the nodes in N_2 represent customers (or, more typically, customer zones), and an arc (i, j) in A represents a distribution channel from warehouse i to customer j.

Circulation problem. The circulation problem is a minimum cost flow problem with only transshipment nodes; that is, $b(i) = 0$ for all $i \in N$. In this instance we wish to find a feasible flow that honors the lower and upper bounds l_{ij} and u_{ij} imposed on the arc flows x_{ij}. Since we never introduce any exogenous flow into the network or extract any flow from it, all the flow circulates around the network. We wish to find the circulation that has the minimum cost. The design of a routing schedule of a commercial airline provides one example of a circulation problem. In this setting, any airplane circulates among the airports of various cities; the lower bound l_{ij} imposed on an arc (i, j) is 1 if the airline needs to provide service between cities i and j, and so must dispatch an airplane on this arc (actually, the nodes will represent a combination of both a physical location and a time of day so that an arc connects, for example, New York City at 8 A.M. with Boston at 9 A.M.).

In this book, we also study the following generalizations of the minimum cost flow problem.

Convex cost flow problems. In the minimum cost flow problem, we assume that the cost of the flow on any arc varies linearly with the amount of flow. Convex cost flow problems have a more general cost structure: The cost is a convex function of the amount of flow. Flow costs vary in a convex manner in numerous problem settings, including (1) power losses in an electrical network due to resistance, (2) congestion costs in a city transportation network, and (3) expansion costs of a communication network.