Network Optimization: Continuous and Discrete Models

Dimitri P. Bertsekas

Massachusetts Institute of Technology

Athena Scientific, Belmont, Massachusetts

Athena Scientific
Post Office Box 391
Belmont, Mass. 02178-9998
U.S.A.

Email: info@athenasc.com
WWW: http://www.athenasc.com

Cover Design: *Ann Gallager*

Publisher's Cataloging-in-Publication Data

Bertsekas, Dimitri P.
Network Optimization: Continuous and Discrete Models
Includes bibliographical references and index
1. Network analysis (Planning). 2. Mathematical Optimization. I. Title.
T57.85.B44 1998 658.4'032-dc20 98-70298

ISBN 1-886529-02-7

ABOUT THE AUTHOR

Dimitri Bertsekas studied Mechanical and Electrical Engineering at the National Technical University of Athens, Greece, and obtained his Ph.D. in system science from the Massachusetts Institute of Technology.

He has held faculty positions at Stanford University and the University of Illinois. Since 1979 he has been teaching at the Massachusetts Institute of Technology (M.I.T.), where he is currently McAfee Professor of Engineering. He consults regularly with private industry and has held editorial positions in several journals. His research spans several fields, including optimization, control, large-scale computation, and data communication networks. He has written many research papers and he is the author or coauthor of thirteen textbooks and research monographs.

Professor Bertsekas was awarded the INFORMS 1997 Prize for Research Excellence in the Interface Between Operations Research and Computer Science for his book ”Neuro-Dynamic Programming” (co-authored with John Tsitsiklis), the 2000 Greek National Award for Operations Research, and the 2001 ACC John R. Ragazzini Education Award. In 2001, he was elected to the United States National Academy of Engineering.

Contents

Preface

Network optimization lies in the middle of the great divide that separates the two major types of optimization problems, continuous and discrete. The ties between linear programming and combinatorial optimization can be traced to the representation of the constraint polyhedron as the convex hull of its extreme points. When a network is involved, however, these ties become much stronger because the extreme points of the polyhedron are integer and represent solutions of combinatorial problems that are seemingly unrelated to linear programming. Because of this structure and also because of their intuitive character, network models provide ideal vehicles for explaining many of the fundamental ideas in both continuous and discrete optimization.

Aside from their interesting methodological characteristics, network models are also used extensively in practice, in an ever expanding spectrum of applications. Indeed collectively, network problems such as shortest path, assignment, max-flow, transportation, transhipment, spanning tree, matching, traveling salesman, generalized assignment, vehicle routing, and multicommodity flow constitute the most common class of practical optimization problems. There has been steady progress in the solution methodology of network problems, and in fact the progress has accelerated in the last fifteen years thanks to algorithmic and technological advances.

The purpose of this book is to provide a fairly comprehensive and up-to-date development of linear, nonlinear, and discrete network optimization problems. The interplay between continuous and discrete structures has been highlighted, the associated analytical and algorithmic issues have been treated quite extensively, and a guide to important network models and applications has been provided.

Regarding continuous network optimization, we focus on two ideas, which are also fundamental in general mathematical programming: *duality* and *iterative cost improvement*. We provide an extensive treatment of iterative algorithms for the most common linear cost problem, the minimum cost flow or transhipment problem, and for its convex cost extensions. The discussion of duality is comprehensive: it starts with linear network programming duality, and culminates with Rockafellar's development of monotropic programming duality.

Regarding discrete network optimization, we illustrate problem formulation through major paradigms such as traveling salesman, generalized assignment, spanning tree, matching, and routing. This is essential because the structure of discrete optimization problems is far less streamlined than the structure of their continuous counterparts, and familiarity with important types of problems is important for modeling, analysis, and algorithmic solution. We also develop the main algorithmic approaches, including branch-and-bound, Lagrangian relaxation, Dantzig-Wolfe decomposition, heuristics, and local search methods.

This is meant to be an introductory book that covers a very broad variety of topics. It is thus inevitable that some topics have been treated in less detail than others. The choices made reflect in part personal taste and expertise, and in part a preference for simple models that can help most effectively the reader develop insight. At the same time, our analysis and presentation aims to enhance the reader's mathematical modeling ability in two ways: by delineating the range of problems for which various algorithms are applicable and efficient, and by providing many examples of problem formulation.

The chapter-by-chapter description of the book follows:

Chapter 1: This is an introductory chapter that establishes terminology and basic notions about graphs, discusses some examples of network models, and provides some orientation regarding linear network optimization algorithms.

Chapter 2: This chapter provides an extensive treatment of shortest path problems. It covers the major methods, and discusses their theoretical and practical performance.

Chapter 3: This chapter focuses on the max-flow problem and develops the class of augmenting path algorithms for its solution. In addition to the classical variants of the Ford-Fulkerson method, a recent algorithm based on auction ideas is discussed.

Chapter 4: The minimum cost flow problem (linear cost, single commodity, no side constraints) and its equivalent variants are introduced here. Subsequently, the basic duality theory for the problem is developed and interpreted.

Chapter 5: This chapter focuses on simplex methods for the minimum cost flow problem. The basic results regarding the integrality of solutions are developed here constructively, using the simplex method. Furthermore, the duality theory of Chapter 4 is significantly strengthened.

Chapter 6: This chapter develops dual ascent methods, including primal-dual, sequential shortest path, and relaxation methods.

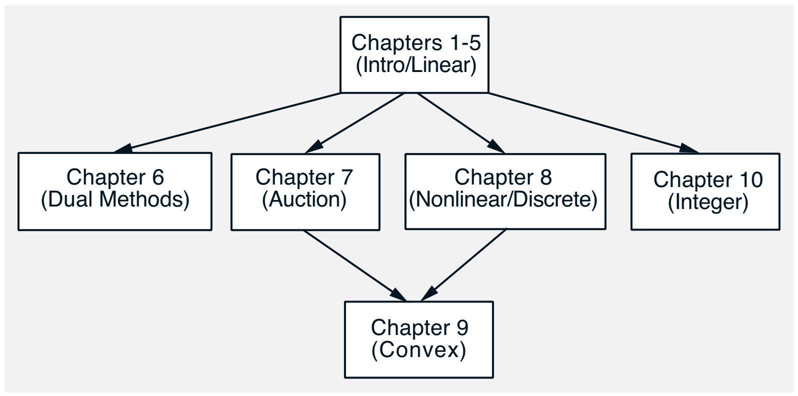
Chapter 7: This chapter starts with the auction algorithm for the assignment problem, and proceeds to show how this algorithm can be extended to more complex problems. In this way, preflow-push methods for the max-flow problem and the ϵ-relaxation method for the minimum cost flow problem are obtained. Several additional variants of auction algorithms are developed.

Chapter 8: This is an important chapter that marks the transition from linear to nonlinear network optimization. The primary focus is on continuous (convex) problems, and their associated broad variety of structures and methodology. In particular, there is an overview of the types of algorithms from nonlinear programming that are useful in connection with various convex network problems. There is also some discussion of discrete (integer) problems with an emphasis on their ties with continuous problems.

Chapter 9: This is a fairly sophisticated chapter that is directed primarily towards the advanced and/or research-oriented reader. It deals with separable convex problems, discusses their connection with classical network equilibrium problems, and develops their rich theoretical structure. The salient features of this structure are a particularly sharp duality theory, and a combinatorial connection of descent directions with the finite set of elementary vectors of the subspace defined by the conservation of flow constraints. Besides treating convex separable network problems, this chapter provides an introduction to monotropic programming, which is the largest class of nonlinear programming problems that possess the strong duality and combinatorial properties of linear programs. This chapter also develops auction algorithms for convex separable problems and provides an analysis of their running time.

Chapter 10: This chapter deals with the basic methodological approaches for integer-constrained problems. There is a treatment of exact methods such as branch-and-bound, and the associated methods of Lagrangian relaxation, subgradient optimization, and cutting plane. There is also a description of approximate methods based on local search, such as genetic algorithms, tabu search, and simulated annealing. Finally, there is a discussion of rollout algorithms, a relatively new and broadly applicable class of approximate methods, which can be used in place of, or in conjunction with local search.

The book can be used for a course on network optimization or for part of a course on introductory optimization at the first-year graduate level. With the exception of some of the material in Chapter 9, the prerequisites are fairly elementary. The main one is a certain degree of mathematical maturity, as provided for example by a rigorous mathematics course beyond the calculus level. One may cover most of the book in a course on linear and nonlinear network optimization. A shorter version of this course may consist of Chapters 1-5, and 8. Alternatively, one may teach a course that focuses on linear and discrete network optimization, using Chapters 1-5, a small part of Chapter 8, and Chapter 10. Actually, in these chapter sequences, it is not essential to cover Chapter 5, if one is content with weaker versions of duality results (given in Chapter 4) and one establishes the integrality properties of optimal solutions with a line of argument such as the one given in Exercise 1.34. The following figure illustrates the chapter dependencies.



The book contains a large number of examples and exercises, which should enhance its suitability for classroom instruction. Some of the exercises are theoretical in nature and supplement substantially the main text. Solutions to a subset of these (as well as errata and additional material) will be posted and periodically updated on the book's web page:

http://www.athenasc.com/netsbook.html

Also, the author's web page

http://web.mit.edu/dimitrib/www/home.html

contains listings of FORTRAN codes implementing many of the algorithms discussed in the book.

There is a very extensive literature on continuous and discrete network optimization, and to give a complete bibliography and a historical account of the research that led to the present form of the subject would have been impossible. Thus I have not attempted to compile a comprehensive list of original contributions to the field. I have cited sources that I have used extensively, that provide important extensions to the material of the book, that survey important topics, or that are particularly well suited for further reading. I have also cited selectively a few sources that are historically significant, but the reference list is far from exhaustive in this respect. Generally, to aid researchers in the field, I have preferred to cite surveys and textbooks for subjects that are relatively mature, and to give a larger number of references for relatively recent developments.

A substantial portion of this book is based on the author's research on network optimization over the last twenty years. I was fortunate to have several outstanding collaborators in this research, and I would like to mention those with whom I have worked extensively. Eli Gafni assisted with the computational experimentation using the auction algorithm and the relaxation method for assignment problems in 1979. The idea of ϵ-scaling arose during my interactions with Eli at that time. Furthermore, Eli collaborated extensively with me on various routing methods for data networks, including projection methods for convex multicommodity flow problems. Paul Tseng worked with me on network optimization starting in 1982. Together we developed the RELAX codes, we developed several extensions to the basic relaxation method and we collaborated closely on a broad variety of other subjects, including the recent auction algorithms for convex network problems and network problems with gains. David Castanon has worked extensively with me on a broad variety of algorithms for assignment, transportation, and minimum cost flow problems, for both serial and parallel computers, since 1987. John Tsitsiklis has been my coauthor and close collaborator for many years on a variety of optimization and large scale computation topics, including some that deal with networks. In addition to Eli, Paul, David, and John, I have had substantial research collaborations with several colleagues, the results of which have been reflected in this book. In this regard, I would like to mention Jon Eckstein, Bob Gallager, Francesca Guerriero, Roberto Musmanno, Stefano Pallottino, and Maria-Grazia Scutellà. Several colleagues proofread portions of the book, and contributed greatly with their suggestions. David Castanon, Stefano Pallottino, Steve Patek, Serap Savari, Paul Tseng, and John Tsitsiklis were particularly helpful in this regard. The research support of NSF under grants from the DDM and the CCI divisions are very much appreciated. My family has been a source of stability and loving support, without which the book would not have been written.

Dimitri P. Bertsekas
Cambridge, Mass.
Spring 1998

1

Introduction

Contents

Network flow problems are one of the most important and most frequently encountered class of optimization problems. They arise naturally in the analysis and design of large systems, such as communication, transportation, and manufacturing networks. They can also be used to model important classes of combinatorial problems, such as assignment, shortest path, and traveling salesman problems.

Loosely speaking, network flow problems consist of supply and demand points, together with several routes that connect these points and are used to transfer the supply to the demand. These routes may contain intermediate transhipment points. Often, the supply, demand, and transhipment points can be modeled by the nodes of a graph, and the routes can be modeled by the paths of the graph. Furthermore, there may be multiple "types" of supply/demand (or "commodities") sharing the routes. There may also be some constraints on the characteristics of the routes, such as their carrying capacities, and some costs associated with using particular routes. Such situations are naturally modeled as network optimization problems whereby, roughly speaking, we try to select routes that minimize the cost of transfer of the supply to the demand.

This book deals with a broad spectrum of network optimization problems, involving linear and nonlinear cost functions. We pay special attention to four major classes of problems:

(a) The *transhipment* or *minimum cost flow problem*, which involves a single commodity and a linear cost function. This problem has several important special cases, such as the shortest path, the max-flow, the assignment, and the transportation problems.

(b) The *single commodity network flow problem with convex cost*. This problem is identical to the preceding transhipment problem, except that the cost function is convex rather than linear.

(c) The *multicommodity network flow problem with linear or convex cost*. This problem generalizes the preceding two classes of problems to the case of multiple commodities.

(d) *Discrete network optimization problems*. These are problems where the quantities transferred along the routes of the network are restricted to take one of a finite number of values. Many combinatorial optimization problems can be modeled in this way, including some problems where the network structure is not immediately apparent. Some discrete optimization problems are computationally very difficult, and in practice can only be solved approximately. Their algorithmic solution often involves the solution of "continuous" subproblems that belong to the preceding three classes.

All of the network flow problems above can be mathematically modeled in terms of graph-related notions. In Section 1.1, we introduce the associated notation and terminology. In Section 1.2, we provide mathematical formulations and practical examples of network optimization models. Finally, in Section 1.3, we give an overview of some of the types of computational algorithms that we develop in subsequent chapters.

1.1 GRAPHS AND FLOWS

In this section, we introduce some of the basic definitions relating to graphs, paths, flows, and other related notions. Graph concepts are fairly intuitive, and can be understood in terms of suggestive figures, but often involve hidden subtleties. Thus the reader may wish to revisit the present section and pay close attention to some of the fine points of the definitions.

A *directed graph*, $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, consists of a set $\mathcal{N}$ of *nodes* and a set $\mathcal{A}$ of pairs of distinct nodes from $\mathcal{N}$ called *arcs*. The numbers of nodes and arcs are denoted by N and A, respectively, and it is assumed throughout that $1 \leq N < \infty$ and $0 \leq A < \infty$. An arc (i, j) is viewed as an ordered pair, and is to be distinguished from the pair (j, i). If (i, j) is an arc, we say that (i, j) is *outgoing* from node i and *incoming* to node j; we also say that j is an *outward neighbor* of i and that i is an *inward neighbor* of j. We say that arc (i, j) is *incident* to i and to j, and that i is the *start* node and j is the *end* node of the arc. We also say that i and j are the *end nodes* of arc (i, j). The *degree* of a node i is the number of arcs that are incident to i. A graph is said to be *complete* if it contains all possible arcs; that is, if there exists an arc for each ordered pair of nodes.

We do not exclude the possibility that there is a separate arc connecting a pair of nodes in each of the two directions. However, we do not allow more than one arc between a pair of nodes in the same direction, so that we can refer unambiguously to the arc with start i and end j as arc (i, j). This is done for notational convenience.† Our analysis can be simply extended to handle multiple arcs with start i and end j; the extension is based on modifying the graph by introducing for each such arc, an additional node, call it n, together with the two arcs (i, n) and (n, j). On occasion, we will pause to provide examples of this type of extension.

We note that much of the literature of graph theory distinguishes between *directed* graphs where an arc (i, j) is an ordered pair to be distinguished from arc (j, i), and *undirected* graphs where an arc is associated with a pair of nodes regardless of order. One may use directed graphs, even in contexts where the use of undirected graphs would be appropriate and conceptually simpler. For this, one may need to replace an undirected arc (i, j) with two directed arcs (i, j) and (j, i) having identical characteristics.

† Some authors use a single symbol, such as a, to denote an arc, and use something like $s(a)$ and $e(a)$ to denote the start and end nodes of a, respectively. This notational method allows the existence of multiple arcs with the same start and end nodes, but is also more cumbersome and less suggestive.

We have chosen to deal exclusively with directed graphs because in our development there are only a few occasions where undirected graphs are convenient. Thus, *all our references to a graph implicitly assume that the graph is directed*. In fact we often omit the qualifier "directed" and refer to a directed graph simply as a *graph*.

1.1.1 Paths and Cycles

A *path* P in a directed graph is a sequence of nodes $(n_1, n_2, \dots, n_k)$ with $k \geq 2$ and a corresponding sequence of $k-1$ arcs such that the ith arc in the sequence is either (n_i, n_{i+1}) (in which case it is called a *forward* arc of the path) or (n_{i+1}, n_i) (in which case it is called a *backward* arc of the path). Nodes n_1 and n_k are called the *start node* (or *origin*) and the *end node* (or *destination*) of P, respectively. A path is said to be *forward* (or *backward*) if all of its arcs are forward (respectively, backward) arcs. We denote by P^+ and P^- the sets of forward and backward arcs of P, respectively.

A *cycle* is a path for which the start and end nodes are the same. A path is said to be *simple* if it contains no repeated arcs and no repeated nodes, except that the start and end nodes could be the same (in which case the path is called a *simple cycle*). A *Hamiltonian cycle* is a simple forward cycle that contains all the nodes of the graph. These definitions are illustrated in Fig. 1.1. We mention that some authors use a slightly different terminology: they use the term "walk" to refer to a path and they use the term "path" to refer to a simple path.

Note that the sequence of nodes $(n_1, n_2, \dots, n_k)$ is not sufficient to specify a path; the sequence of arcs may also be important, as Fig. 1.1(c) shows. The difficulty arises when for two successive nodes n_i and n_{i+1} of the path, both (n_i, n_{i+1}) and (n_{i+1}, n_i) are arcs, so there is ambiguity as to which of the two is the corresponding arc of the path. If a path is known to be forward or is known to be backward, it is uniquely specified by the sequence of its nodes. Otherwise, however, the intended sequence of arcs must be explicitly defined.

A graph that contains no simple cycles is said to be *acyclic*. A graph is said to be *connected* if for each pair of nodes i and j, there is a path starting at i and ending at j; it is said to be *strongly connected* if for each pair of nodes i and j, there is a forward path starting at i and ending at j. Thus, for example, the graph of Fig. 1.1(b) is connected but not strongly connected. It can be shown that if a graph is connected and each of its nodes has even degree, there is a cycle (not necessarily forward) that contains all the arcs of the graph exactly once (see Exercise 1.5). Such a cycle is called an *Euler cycle*, honoring the historically important work of Euler; see the discussion in Section 10.1 about the Königsberg bridge problem. Figure 1.2 gives an example of an Euler cycle.

We say that a graph $\mathcal{G}' = (\mathcal{N}', \mathcal{A}')$ is a *subgraph* of a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ if $\mathcal{N}' \subset \mathcal{N}$ and $\mathcal{A}' \subset \mathcal{A}$. A *tree* is a connected acyclic graph. A *spanning*

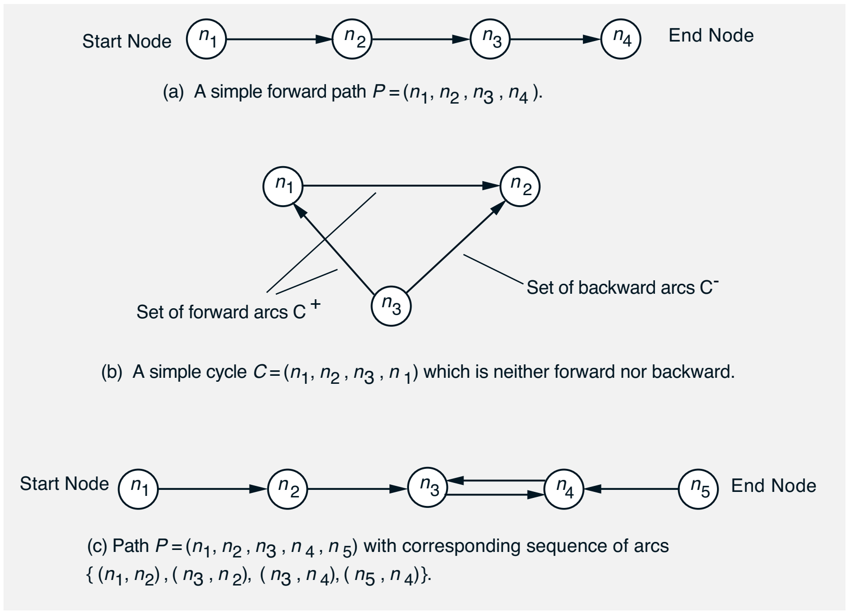

Start Node
n_1
n_2
n_3
n_4
End Node
(a) A simple forward path $P = (n_1, n_2, n_3, n_4)$.
n_1
n_2
Set of forward arcs C^+
n_3
Set of backward arcs C^-
(b) A simple cycle $C = (n_1, n_2, n_3, n_1)$ which is neither forward nor backward.
Start Node
n_1
n_2
n_3
n_4
n_5
End Node
(c) Path $P = (n_1, n_2, n_3, n_4, n_5)$ with corresponding sequence of arcs
$\{(n_1, n_2), (n_3, n_2), (n_3, n_4), (n_5, n_4)\}$.

Figure 1.1: Illustration of various types of paths and cycles. The cycle in (b) is not a Hamiltonian cycle; it is simple and contains all the nodes of the graph, but it is not forward. Note that for the path (c), in order to resolve ambiguities, it is necessary to specify the sequence of arcs of the path (rather than just the sequence of nodes) because both (n_3, n_4) and (n_4, n_3) are arcs.

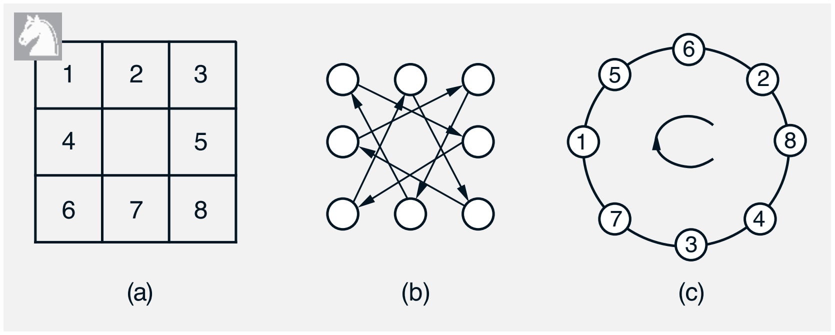


Figure 1.2: Example of an Euler cycle. Consider a 3×3 chessboard, where the middle square has been deleted. A knight starting at one of the squares of the board can visit every other square exactly once and return to the starting square as shown in the graph (b), or equivalently in (c). In the process, the knight will make all the possible moves (in one direction only), or equivalently, it will cross every arc of the graph in (b) exactly once. The knight's tour is an Euler cycle for the graph of (b).

tree of a graph $\mathcal{G}$ is a subgraph of $\mathcal{G}$, which is a tree and includes all the nodes of $\mathcal{G}$. It can be shown [Exercise 1.14(c)] that a subgraph is a spanning tree if and only if it is connected and it contains $N-1$ arcs.

1.1.2 Flow and Divergence

In many applications involving graphs, it is useful to introduce a variable that measures the quantity flowing through each arc, like for example, electric current in an electric circuit, or water flow in a hydraulic network. We refer to such a variable as the *flow of an arc*. Mathematically, the flow of an arc (i,j) is simply a scalar (real number), which we usually denote by x_{ij}. It is convenient to allow negative as well as positive values for flow. In applications, a negative arc flow indicates that whatever is represented by the flow (material, electric current, etc.), moves in a direction opposite to the direction of the arc. We can always change the sign of a negative arc flow to positive as long as we change the arc direction, so in many situations we can assume without loss of generality that all arc flows are nonnegative. For the development of a general methodology, however, this device is often cumbersome, which is why we prefer to simply accept the possibility of negative arc flows.

Given a graph $(\mathcal{N},\mathcal{A})$, a set of flows $\big\{x_{ij} \mid (i,j) \in \mathcal{A}\big\}$ is referred to as a *flow vector*. The *divergence vector* y associated with a flow vector x is the N-dimensional vector with coordinates

$$y_i = \sum_{\{j \mid (i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j \mid (j,i)\in\mathcal{A}\}} x_{ji}, \qquad \forall\ i \in \mathcal{N}. \tag{1.1}$$

Thus, y_i is the total flow departing from node i less the total flow arriving at i; it is referred to as the *divergence of* i.

We say that node i is a *source* (respectively, *sink*) for the flow vector x if $y_i > 0$ (respectively, $y_i < 0$). If $y_i = 0$ for all $i \in \mathcal{N}$, then x is called a *circulation*. These definitions are illustrated in Fig. 1.3. Note that by adding Eq. (1.1) over all $i \in \mathcal{N}$, we obtain

$$\sum_{i\in\mathcal{N}} y_i = 0.$$

Every divergence vector y must satisfy this equation.

The flow vectors x that we will consider will often be constrained to lie between given lower and upper bounds of the form

$$b_{ij} \le x_{ij} \le c_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}.$$

Given a flow vector x that satisfies these bounds, we say that a path P is *unblocked with respect to* x if, roughly speaking, we can send some positive flow along P without violating the bound constraints; that is, if flow can

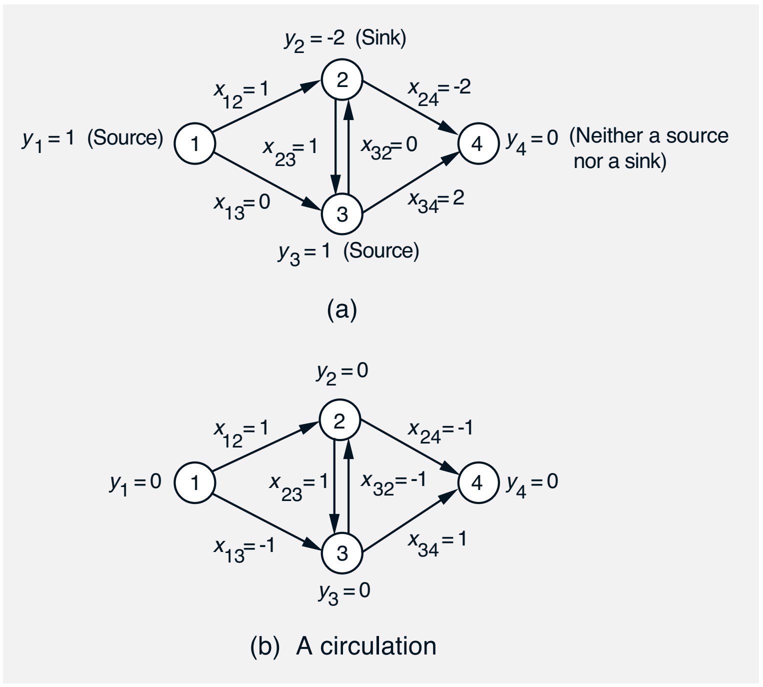


Figure 1.3: Illustration of flows x_{ij} and the corresponding divergences y_i. The flow in (b) is a circulation because $y_i = 0$ for all i.

be increased on the set P^+ of the forward arcs of P, and can be decreased on the set P^- of the backward arcs of P:

$$x_{ij} < c_{ij}, \quad \forall\ (i,j) \in P^+, \qquad b_{ij} < x_{ij}, \quad \forall\ (i,j) \in P^-.$$

For example, in Fig. 1.3(a), suppose that all arcs (i,j) have flow bounds $b_{ij} = -2$ and $c_{ij} = 2$. Then the path consisting of the sequence of nodes $(1,2,4)$ is unblocked, while the reverse path $(4,2,1)$ is not unblocked.

1.1.3 Path Flows and Conformal Decomposition

A *simple path flow* is a flow vector that corresponds to sending a positive amount of flow along a simple path; more precisely, it is a flow vector x with components of the form

$$x_{ij} = \begin{cases} a & \text{if } (i,j) \in P^+, \\ -a & \text{if } (i,j) \in P^-, \\ 0 & \text{otherwise,} \end{cases} \tag{1.2}$$

where a is a positive scalar, and P^+ and P^- are the sets of forward and backward arcs, respectively, of some simple path P. Note that the path P may be a cycle, in which case x is also called a *simple cycle flow*.

It is often convenient to break down a flow vector into the sum of simple path flows. This leads to the notion of a conformal realization, which we proceed to discuss.

We say that a path P *conforms* to a flow vector x if $x_{ij} > 0$ for all forward arcs (i, j) of P and $x_{ij} < 0$ for all backward arcs (i, j) of P, and furthermore either P is a cycle or else the start and end nodes of P are a source and a sink of x, respectively. Roughly, a path conforms to a flow vector if it "carries flow in the forward direction," i.e., in the direction from the start node to the end node. In particular, for a forward cycle to conform to a flow vector, all its arcs must have positive flow. For a forward path which is not a cycle to conform to a flow vector, its arcs must have positive flow, and in addition the start and end nodes must be a source and a sink, respectively; for example, in Fig. 1.3(a), the path consisting of the sequence of arcs (1,2), (2,3), (3,4) does not conform to the flow vector shown, because node 4, the end node of the path, is not a sink.

We say that a simple path flow x^s *conforms* to a flow vector x if the path P corresponding to x^s via Eq. (1.2) conforms to x. This is equivalent to requiring that

$$\begin{aligned} 0 < x_{ij} \qquad & \text{for all arcs } (i,j) \text{ with } 0 < x^s_{ij}, \\ x_{ij} < 0 \qquad & \text{for all arcs } (i,j) \text{ with } x^s_{ij} < 0, \end{aligned}$$

and that either P is a cycle or else the start and end nodes of P are a source and a sink of x, respectively.

An important fact is that any flow vector can be decomposed into a set of conforming simple path flows, as illustrated in Fig. 1.4. We state this as a proposition. The proof is based on an algorithm that can be used to construct the conforming components one by one (see Exercise 1.2).

Proposition 1.1: (Conformal Realization Theorem) A nonzero flow vector x can be decomposed into the sum of t simple path flow vectors $x^1, x^2, \ldots, x^t$ that conform to x, with t being at most equal to the sum of the numbers of arcs and nodes $A + N$. If x is integer, then $x^1, x^2, \ldots, x^t$ can also be chosen to be integer. If x is a circulation, then $x^1, x^2, \ldots, x^t$ can be chosen to be simple cycle flows, and $t \leq A$.

1.2 NETWORK FLOW MODELS – EXAMPLES

In this section we introduce some of the major classes of problems that will be discussed in this book. We begin with the *minimum cost flow problem*, which, together with its special cases, will be the subject of the following six chapters.

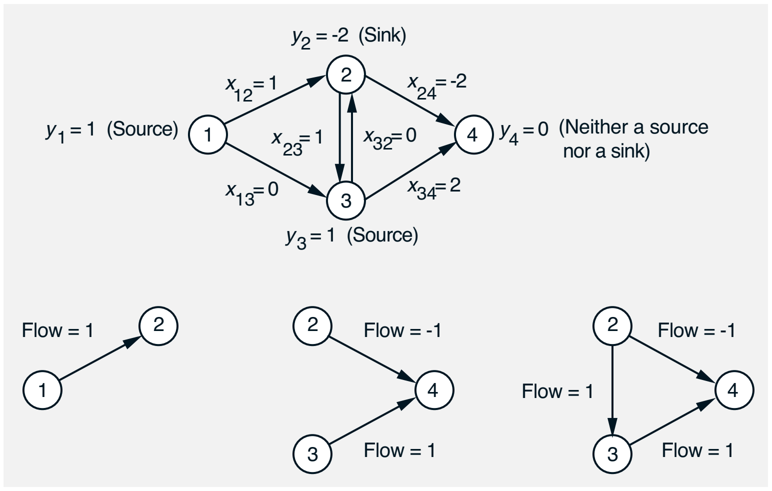


Figure 1.4: Decomposition of a flow vector x into three simple path flows conforming to x. Consistent with the definition of conformance of a path flow, each arc (i, j) of the three component paths carries positive (or negative) flow only if $x_{ij} > 0$ (or $x_{ij} < 0$, respectively). The first two paths $[(1, 2)$ and $(3, 4, 2)]$ are not cycles, but they start at a source and end at a sink, as required. Arcs $(1, 3)$ and $(3, 2)$ do not belong to any of these paths because they carry zero flow. In this example, the decomposition is unique, but in general this need not be the case.

1.2.1 The Minimum Cost Flow Problem

This problem is to find a set of arc flows that minimize a linear cost function, subject to the constraints that they produce a given divergence vector and they lie within some given bounds; that is,

$$\text{minimize} \quad \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} \tag{1.3}$$

subject to the constraints

$$\sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j|(j,i)\in\mathcal{A}\}} x_{ji} = s_i, \qquad \forall\ i \in \mathcal{N}, \tag{1.4}$$

$$b_{ij} \le x_{ij} \le c_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}, \tag{1.5}$$

where a_{ij}, b_{ij}, c_{ij}, and s_i are given scalars. We use the following terminology:

a_{ij}: the *cost coefficient* (or simply *cost*) of (i, j),

b_{ij} and c_{ij}: the *flow bounds* of (i, j),

$[b_{ij}, c_{ij}]$: the *feasible flow range* of (i, j),

s_i: the *supply* of node i (when s_i is negative, the scalar $-s_i$ is called the *demand* of i).

We also refer to the constraints (1.4) and (1.5) as the *conservation of flow constraints*, and the *capacity constraints*, respectively. A flow vector satisfying both of these constraints is called *feasible*, and if it satisfies just the capacity constraints, it is called *capacity-feasible*. If there exists at least one feasible flow vector, the minimum cost flow problem is called *feasible*; otherwise it is called *infeasible*. On occasion, we will consider the variation of the minimum cost flow problem where the lower or the upper flow bound of some of the arcs is either $-\infty$ or ∞, respectively. In these cases, we will explicitly state so.

For a typical application of the minimum cost flow problem, think of the nodes as locations (cities, warehouses, or factories) where a certain product is produced or consumed. Think of the arcs as transportation links between the locations, each with transportation cost a_{ij} per unit transported. The problem then is to move the product from the production points to the consumption points at minimum cost while observing the capacity constraints of the transportation links.

However, the minimum cost flow problem has many applications that are well beyond the transportation context just described, as will be seen from the following examples. These examples illustrate how some important discrete/combinatorial problems can be modeled as minimum cost flow problems, and highlight the important connection between continuous and discrete network optimization.

Example 1.1. The Shortest Path Problem

Suppose that each arc (i, j) of a graph is assigned a scalar cost a_{ij}, and suppose that we define the cost of a forward path to be the sum of the costs of its arcs. Given a pair of nodes, the shortest path problem is to find a forward path that connects these nodes and has minimum cost. An analogy here is made between arcs and their costs, and roads in a transportation network and their lengths, respectively. Within this transportation context, the problem becomes one of finding the shortest route between two geographical points. Based on this analogy, the problem is referred to as the *shortest path problem*, and the arc costs and path costs are commonly referred to as the *arc lengths* and *path lengths*, respectively.

The shortest path problem arises in a surprisingly large number of contexts. For example in a data communication network, a_{ij} may denote the average delay of a packet to cross the communication link (i, j), in which case a shortest path is a minimum average delay path that can be used for routing the packet from its origin to its destination. As another example, if p_{ij} is the probability that a given arc (i, j) in a communication network is usable, and each arc is usable independently of all other arcs, then the product of the probabilities of the arcs of a path provides a measure of reliability of the path. With this in mind, it is seen that finding the most reliable path connecting two nodes is equivalent to finding the shortest path between the two nodes with arc lengths $(-\ln p_{ij})$.

The shortest path problem also arises often as a subroutine in algorithms that solve other more complicated problems. Examples are the primal-dual algorithm for solving the minimum cost flow problem (see Chapter 6), and the conditional gradient and projection algorithms for solving multicommodity flow problems (see Chapter 8).

It is possible to cast the problem of finding a shortest path from node s to node t as the following minimum cost flow problem:

$$\begin{aligned}
&\text{minimize} \quad \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} \\
&\text{subject to} \quad \sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j|(j,i)\in\mathcal{A}\}} x_{ji} = \begin{cases} 1 & \text{if } i = s, \\ -1 & \text{if } i = t, \\ 0 & \text{otherwise,} \end{cases} \\
&\qquad 0 \le x_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}.
\end{aligned} \tag{1.6}$$

To see this, let us associate with any forward path P from s to t the flow vector x with components given by

$$x_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ belongs to } P, \\ 0 & \text{otherwise.} \end{cases} \tag{1.7}$$

Then x is feasible for problem (1.6) and the cost of x is equal to the length of P. Thus, if a vector x of the form (1.7) is an optimal solution of problem (1.6), the corresponding path P is shortest.

Conversely, it can be shown that if problem (1.6) has at least one optimal solution, then it has an optimal solution of the form (1.7), with a corresponding path P that is shortest. This is not immediately apparent, but its proof can be traced to a remarkable fact that we will show in Chapter 5 about minimum cost flow problems with node supplies and arc flow bounds that are integer: such problems, if they have an optimal solution, they have an *integer* optimal solution, that is, a set of optimal arc flows that are integer (an alternative proof of this fact is sketched in Exercise 1.34). From this it follows that if problem (1.6) has an optimal solution, it has one with arc flows that are 0 or 1, and which is of the form (1.7) for some path P. This path is shortest because its length is equal to the optimal cost of problem (1.6), so it must be less or equal to the cost of any other flow vector of the form (1.7), and therefore also less or equal to the length of any other path from s to t. Thus the shortest path problem is essentially equivalent with the minimum cost flow problem (1.6).

Example 1.2. The Assignment Problem

Suppose that there are n persons and n objects that we have to match on a one-to-one basis. There is a benefit or value a_{ij} for matching person i with object j, and we want to assign persons to objects so as to maximize the total

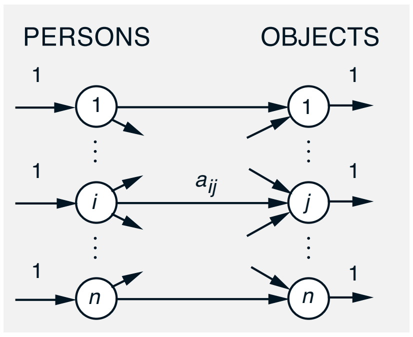


Figure 1.5: The graph representation of an assignment problem.

benefit. There is also a restriction that person i can be assigned to object j only if (i,j) belongs to a given set of pairs $\mathcal{A}$. Mathematically, we want to find a set of person-object pairs $(1,j_1),\ldots,(n,j_n)$ from $\mathcal{A}$ such that the objects $j_1,\ldots,j_n$ are all distinct, and the total benefit $\sum_{i=1}^n a_{ij_i}$ is maximized.

The assignment problem is important in many practical contexts. The most obvious ones are resource allocation problems, such as assigning employees to jobs, machines to tasks, etc. There are also situations where the assignment problem appears as a subproblem in methods for solving various complex combinatorial problems (see Chapter 10).

We may associate any assignment with the set of variables $\{x_{ij} \mid (i,j)\in \mathcal{A}\}$, where $x_{ij}=1$ if person i is assigned to object j and $x_{ij}=0$ otherwise. The value of this assignment is $\sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij}$. The restriction of one object per person can be stated as $\sum_j x_{ij}=1$ for all i and $\sum_i x_{ij}=1$ for all j. We may then formulate the assignment problem as the linear program

$$\begin{aligned} &\text{maximize} && \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} \\ &\text{subject to} && \sum_{\{j\mid(i,j)\in\mathcal{A}\}} x_{ij}=1, \qquad \forall\ i=1,\ldots,n, \\ &&& \sum_{\{i\mid(i,j)\in\mathcal{A}\}} x_{ij}=1, \qquad \forall\ j=1,\ldots,n, \\ &&& 0\le x_{ij}\le 1, \qquad \forall\ (i,j)\in\mathcal{A}. \end{aligned} \tag{1.8}$$

Actually we should further restrict x_{ij} to be either 0 or 1. However, as we will show in Chapter 5, the above linear program has the property that if it has a feasible solution at all, then it has an optimal solution where all x_{ij} are either 0 or 1 (compare also with the discussion in the preceding example and Exercise 1.34). In fact, the set of its optimal solutions includes all the optimal assignments.

We now argue that the assignment/linear program (1.8) is a minimum cost flow problem involving the graph shown in Fig. 1.5. Here, there are $2n$ nodes divided into two groups: n corresponding to persons and n corresponding to objects. Also, for every possible pair $(i,j)\in\mathcal{A}$, there is an arc connecting person i with object j. The variable x_{ij} is the flow of arc (i,j).

The constraint

$$\sum_{\{j|(i,j)\in\mathcal{A}\}} x_{ij} = 1$$

indicates that the divergence of person/node i should be equal to 1, while the constraint

$$\sum_{\{i|(i,j)\in\mathcal{A}\}} x_{ij} = 1$$

indicates that the divergence of object/node j should be equal to -1. Finally, we may view $(-a_{ij})$ as the cost coefficient of the arc (i,j) (by reversing the sign of a_{ij}, we convert the problem from a maximization to a minimization problem).

Example 1.3. The Max-Flow Problem

In the max-flow problem, we have a graph with two special nodes: the *source*, denoted by s, and the *sink*, denoted by t. Roughly, the objective is to move as much flow as possible from s into t while observing the capacity constraints. More precisely, we want to find a flow vector that makes the divergence of all nodes other than s and t equal to 0 while maximizing the divergence of s.

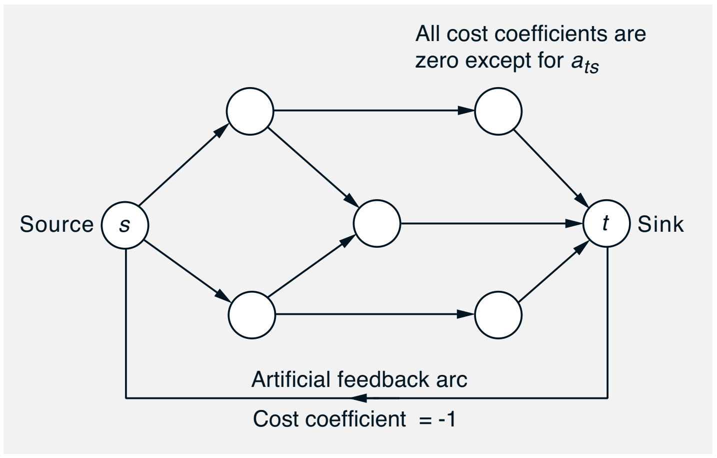


Figure 1.6: The minimum cost flow representation of a max-flow problem. At the optimum, the flow x_{ts} equals the maximum flow that can be sent from s to t through the subgraph obtained by deleting the artificial arc (t,s).

The max-flow problem arises in many practical contexts, such as calculating the throughput of a highway system or a communication network. It also arises often as a subproblem in more complicated problems or algorithms; in particular, it bears a fundamental connection to the question of existence of a feasible solution of a general minimum cost flow problem (see our discussion in Chapter 3). Finally, several discrete/combinatorial optimization problems can be formulated as max-flow problems (see the Exercises in Chapter 3).

We formulate the problem as a special case of the minimum cost flow problem by assigning cost 0 to all arcs and by introducing an artificial arc (t, s) with cost -1, as shown in Fig. 1.6. Mathematically, the problem is:

$$\begin{aligned}
&\text{maximize} \quad x_{ts} \\
&\text{subject to} \\
&\sum_{\{j \mid (i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j \mid (j,i)\in\mathcal{A}\}} x_{ji} = 0, \qquad \forall\, i \in \mathcal{N} \text{ with } i \neq s \text{ and } i \neq t, \\
&\sum_{\{j \mid (s,j)\in\mathcal{A}\}} x_{sj} = \sum_{\{i \mid (i,t)\in\mathcal{A}\}} x_{it} = x_{ts}, \\
&b_{ij} \le x_{ij} \le c_{ij}, \qquad \forall\, (i,j) \in \mathcal{A} \text{ with } (i,j) \neq (t,s).
\end{aligned}$$

Viewing the problem as a maximization is consistent with its intuitive interpretation. Alternatively, we could write the problem as a minimization of $-x_{ts}$ subject to the same constraints. Also, we could introduce upper and lower bounds on x_{ts},

$$\sum_{\{i \mid (i,t)\in\mathcal{A}\}} b_{it} \le x_{ts} \le \sum_{\{i \mid (i,t)\in\mathcal{A}\}} c_{it},$$

but these bounds are actually redundant since they are implied by the other upper and lower arc flow bounds.

Example 1.4. The Transportation Problem

This problem is the same as the assignment problem except that the node supplies need not be 1 or -1, and the numbers of sources and sinks need not be equal. It has the form

$$\begin{aligned}
&\text{minimize} \quad \sum_{(i,j)\in\mathcal{A}} a_{ij} x_{ij} \\
&\text{subject to} \quad \sum_{\{j \mid (i,j)\in\mathcal{A}\}} x_{ij} = \alpha_i, \qquad \forall\, i = 1,\ldots,m, \\
&\qquad\qquad \sum_{\{i \mid (i,j)\in\mathcal{A}\}} x_{ij} = \beta_j, \qquad \forall\, j = 1,\ldots,n, \\
&\qquad\qquad 0 \le x_{ij} \le \min\{\alpha_i, \beta_j\}, \qquad \forall\, (i,j) \in \mathcal{A}.
\end{aligned} \tag{1.9}$$

Here α_i and β_j are positive scalars, which for feasibility must satisfy

$$\sum_{i=1}^{m} \alpha_i = \sum_{j=1}^{n} \beta_j,$$

(add the conservation of flow constraints). In an alternative formulation, the upper bound constraint $x_{ij} \leq \min\{\alpha_i, \beta_j\}$ could be discarded, since it is implied by the conservation of flow and the nonnegativity constraints.

As a practical example of a transportation problem that has a combinatorial flavor, suppose that we have m communication terminals, each to be connected to one of n traffic concentrators. We introduce variables x_{ij}, which take the value 1 if terminal i is connected to concentrator j. Assuming that concentrator j can be connected to no more than b_j terminals, we obtain the constraints

$$\sum_{i=1}^{m} x_{ij} \leq b_j, \qquad \forall\ j = 1, \ldots, n.$$

Also, since each terminal must be connected to exactly one concentrator, we have the constraints

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad \forall\ i = 1, \ldots, m.$$

Assuming that there is a cost a_{ij} for connecting terminal i to concentrator j, the problem is to find the connection of minimum cost, that is, to minimize

$$\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} x_{ij}$$

subject to the preceding constraints. This problem is not yet a transportation problem of the form (1.9) for two reasons:

(a) The arc flows x_{ij} are constrained to be 0 or 1.

(b) The constraints $\sum_{i=1}^{m} x_{ij} \leq b_j$ are not equality constraints, as required in problem (1.9).

It turns out, however, that we can ignore the 0-1 constraint on x_{ij}. As discussed in connection with the shortest path and assignment problems, even if we relax this constraint and replace it with the capacity constraint $0 \leq x_{ij} \leq 1$, there is an optimal solution such that each x_{ij} is either 0 or 1. Furthermore, to convert the inequality constraints to equalities, we can introduce a total of $\sum_{j=1}^{n} b_j - m$ "dummy" terminals that can be connected at zero cost to all of the concentrators. In particular, we introduce a special supply node 0 together with the constraint

$$\sum_{j=1}^{n} x_{0j} = \sum_{j=1}^{n} b_j - m,$$

and we change the inequality constraints $\sum_{j=1}^{n} x_{ij} \leq b_j$ to

$$x_{0j} + \sum_{i=1}^{m} x_{ij} = b_j.$$

The resulting problem has the transportation structure of problem (1.9), and is equivalent to the original problem.

1.2.2 Network Flow Problems with Convex Cost

A more general version of the minimum cost flow problem arises when the cost function is convex rather than linear. An important special case is the problem

$$\begin{aligned} &\text{minimize} \quad \sum_{(i,j)\in\mathcal{A}} f_{ij}(x_{ij}) \\ &\text{subject to} \quad \sum_{\{j\mid(i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j\mid(j,i)\in\mathcal{A}\}} x_{ji} = s_i, \qquad \forall\, i\in\mathcal{N}, \\ &\qquad\qquad\quad x_{ij}\in X_{ij}, \qquad \forall\,(i,j)\in\mathcal{A}, \end{aligned}$$

where f_{ij} is a convex function of the flow x_{ij} of arc (i,j), s_i are given scalars, and X_{ij} are convex intervals of real numbers, such as for example

$$X_{ij} = [b_{ij}, c_{ij}],$$

where b_{ij} and c_{ij} are given scalars. We refer to this as the *separable convex cost network flow problem*, because the cost function separates into the sum of cost functions, one per arc. This problem will be discussed in detail in Chapters 8 and 9.

Example 1.5. The Matrix Balancing Problem

Here the problem is to find an $m \times n$ matrix X that has given row sums and column sums, and approximates a given $m \times n$ matrix M in some optimal manner. We can formulate such a problem in terms of a graph consisting of m sources and n sinks. In this graph, the set of arcs consists of the pairs (i,j) for which the corresponding entry x_{ij} of the matrix X is allowed to be nonzero. The given row sums r_i and the given column sums c_j are expressed as the constraints

$$\sum_{\{j\mid(i,j)\in A\}} x_{ij} = r_i, \qquad i = 1,\ldots,m,$$

$$\sum_{\{i\mid(i,j)\in A\}} x_{ij} = c_j, \qquad j = 1,\ldots,n.$$

There may be also bounds for the entries x_{ij} of X. Thus, the structure of this problem is similar to the structure of a transportation problem. The cost function to be optimized has the form

$$\sum_{(i,j)\in A} f_{ij}(x_{ij}),$$

and expresses the objective of making the entries of X close to the corresponding entries of the given matrix M. A commonly used example is the quadratic function

$$f_{ij}(x_{ij}) = w_{ij}(x_{ij} - m_{ij})^2,$$

where w_{ij} are given positive scalars.

Another interesting cost function is the logarithmic

$$f_{ij}(x_{ij}) = x_{ij}\left[\ln\left(\frac{x_{ij}}{m_{ij}}\right) - 1\right],$$

where we assume that $m_{ij} > 0$ for all $(i,j) \in A$. Note that this function is not defined for $x_{ij} \leq 0$, so to obtain a problem that fits our framework, we must use a constraint interval of the form $X_{ij} = (0,\infty)$ or $X_{ij} = (0, c_{ij}]$, where c_{ij} is a positive scalar.

An example of a practical problem that can be addressed using the preceding optimization model is to predict the distribution matrix X of telephone traffic between m origins and n destinations. Here we are given the total supplies r_i of the origins and the total demands c_j of the destinations, and we are also given some matrix M that defines a nominal traffic pattern obtained from historical data.

There are other types of network flow problems with convex cost that often arise in practice. We generically represent such problems in the form

$$\begin{aligned} &\text{minimize} \;\; f(x) \\ &\text{subject to} \;\; x \in F \end{aligned}$$

where F is a convex subset of flow vectors in a graph and f is a convex function over the set F. We will discuss in some detail various classes of problems of this type in Chapter 8, and we will see that they arise in several different ways; for example, the cost function may be *nonseparable* because of coupling of the costs of several arc flows, and/or there may be *side constraints*, whereby the flows of several arcs are jointly restricted by the availability of resource. An important example is multicommodity flow problems, which we discuss next.

1.2.1 Multicommodity Flow Problems

Multicommodity network flow problems involve several "types" of flow or *commodities*, which simultaneously use the network and are coupled through either constraints, such as arc flow bounds, or through the cost function. Some important examples of such problems arise in communication, transportation, and manufacturing networks. In the context of communication networks the commodities are the streams of different classes of traffic (telephone calls, data, video, etc.) that involve different origin-destination pairs. Thus there is a separate commodity per class of traffic and origin-destination pair. The following example introduces this context. In Chapter 8, we will discuss similar and/or more general multicommodity network flow problems that arise in other practical contexts.

Example 1.6. Routing in Data Networks

We are given a directed graph, which is viewed as a model of a data communication network. We are also given a set of ordered node pairs (i_m, j_m), $m = 1, \ldots, M$, referred to as *origin-destination (OD) pairs*. The nodes i_m and j_m are referred to as the *origin* and the *destination* of the OD pair. For each OD pair (i_m, j_m), we are given a scalar r_m that represents its input traffic. In the context of routing of data in a communication network, r_m (measured for example in bits/second) is the arrival rate of traffic entering the network at node i_m and exiting at node j_m. The routing objective is to divide each r_m among the many paths from the origin i_m to the destination j_m in a way that the resulting total arc flow pattern minimizes a suitable cost function (see Fig. 1.7).

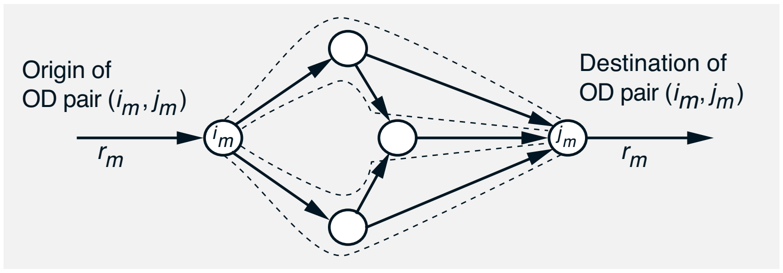


Figure 1.7: Illustration of how the input r_m of the OD pair (i_m, j_m) is divided into nonnegative path flows that start at i_m and end at j_m. The flows of the different OD pairs interact by sharing the arcs of the network.

If we denote by $x_{ij}(m)$ the flow on arc (i, j) of OD pair (i_m, j_m), we have the conservation of flow constraints

$$\sum_{\{j \mid (i,j) \in \mathcal{A}\}} x_{ij}(m) - \sum_{\{j \mid (j,i) \in \mathcal{A}\}} x_{ji}(m) = \begin{cases} r_m & \text{if } i = i_m, \\ -r_m & \text{if } i = j_m, \\ 0 & \text{otherwise}, \end{cases} \qquad \forall\, i \in \mathcal{N},$$

for each $m = 1, \ldots, M$. Furthermore, the flows $x_{ij}(m)$ are required to be nonnegative, and possibly to satisfy additional constraints, such as upper bounds. The cost function often has the form

$$f(x) = \sum_{(i,j) \in \mathcal{A}} f_{ij}(y_{ij}),$$

where f_{ij} is a function of the *total flow* of arc (i,j)

$$y_{ij} = \sum_{m=1}^{M} x_{ij}(m).$$

Such a cost function is often based on a queueing model of average delay (see for example the data network textbook by Bertsekas and Gallager [1992]).

1.2.4 Discrete Network Optimization Problems

Many linear or convex network flow problems, in addition to the conservation of flow constraints and arc flow bounds, involve some additional constraints. In particular, there may be constraints that couple the flows of different arcs, and there may also be *integer constraints* on the arc flows, such as for example that each arc flow be either 0 or 1. Several famous combinatorial optimization problems, such as the following one, are of this type.

Example 1.7. The Traveling Salesman Problem

This problem refers to a salesman who wants to find a minimum mileage/cost tour that visits each of N given cities exactly once and returns to the city he started from. To convert this to a network flow problem, we associate a node with each city $i = 1, \ldots, N$, and we introduce an arc (i,j) with traversal cost a_{ij} for each ordered pair of nodes i and j. A *tour* is synonymous to a Hamiltonian cycle, which was earlier defined to be a simple forward cycle that contains all the nodes of the graph. Equivalently, a tour is a connected subgraph that consists of N arcs, such that there is exactly one incoming and one outgoing arc for each node $i = 1, \ldots, N$. The problem is to find a tour with minimum sum of arc costs.

To formulate this problem as a network flow problem, we denote by x_{ij} the flow of arc (i,j) and we require that this flow is either 1 or 0, indicating that the arc is or is not part of the tour, respectively. The cost of a tour T is then

$$\sum_{(i,j)\in T} a_{ij} x_{ij}.$$

The constraint that each node has a single incoming and a single outgoing arc on the tour is expressed by the following two conservation of flow equations:

$$\sum_{\substack{j=1,\ldots,N \\ j\neq i}} x_{ij} = 1, \qquad i = 1, \ldots, N,$$

$$\sum_{\substack{i=1,\ldots,N \\ i\neq j}} x_{ij} = 1, \qquad j = 1, \ldots, N.$$

There is one additional connectivity constraint:

the subgraph with node set $\mathcal{N}$ and arc set $\{(i,j) \mid x_{ij} = 1\}$ is connected.

If this constraint was not present, the problem would be an ordinary assignment problem. Unfortunately, this constraint is essential, since without it, there would be feasible solutions involving multiple disconnected cycles.

Despite the similarity, the traveling salesman problem is far more difficult than the assignment problem. Solving problems having a mere few hundreds of nodes can be very challenging. By contrast, assignment problems with hundreds of thousands of nodes can be solved in reasonable time with the presently available methodology.

Actually, we have already described some discrete/combinatorial problems that fall within the framework of the minimum cost flow problem, such as shortest path and assignment (cf. Examples 1.1 and 1.2). These problems require that the arc flows be 0 or 1, but, as mentioned earlier, we can neglect these 0-1 constraints because it turns out that even if we relax them and replace them with flow bound intervals $[0, 1]$, we can obtain optimal flows that are 0 or 1 (for a proof, see Section 5.2 or Exercise 1.34).

On the other hand, once we deviate from the minimum cost flow structure and we impose additional constraints or use a nonlinear cost function, the integer character of optimal solutions is lost, and all integer constraints must be explicitly imposed. This often complicates dramatically the solution process, and in fact it may be practically impossible to obtain an exactly optimal solution. As we will discuss in Chapter 10, there are several approximate solution approaches that are based on simplified versions of the problem, such as relaxing the integer constraints. These simplified problems can often be addressed with the efficient minimum cost flow algorithms that we will develop in Chapters 2-7.

1.3 NETWORK FLOW ALGORITHMS – AN OVERVIEW

This section, which may be skipped without loss of continuity, provides a broad classification of the various classes of algorithms for linear and convex network optimization problems. It turns out that these algorithms rely on just a few basic ideas, so they can be easily grouped in a few major categories. By contrast, there is a much larger variety of algorithmic ideas for discrete optimization problems. For this reason, we postpone the corresponding discussion for Chapter 10.

Network optimization problems typically cannot be solved analytically. Usually they must be addressed computationally with one of several available algorithms. One possibility, for linear and convex problems, is to use a general purpose linear or nonlinear programming algorithm. However, the network structure can be exploited to speed up the solution by using either an adaptation of a general purpose algorithm such as the simplex method, or by using a specialized network optimization algorithm. In practice, network optimization problems can often be solved hundreds and even thousands of times faster than general linear or convex programs of comparable dimension.

The algorithms for linear and convex network problems that we will discuss in this book can be grouped in three main categories:

(a) *Primal cost improvement.* Here we try to iteratively improve the cost to its optimal value by constructing a corresponding sequence of feasible flows.

(b) *Dual cost improvement.* Here we define a problem related to the original network flow problem, called the *dual problem*, whose variables are called *prices*. We then try to iteratively improve the dual cost to its optimal value by constructing a corresponding sequence of prices. Dual cost improvement algorithms also iterate on flows, which are related to the prices through a property called *complementary slackness*.

(c) *Auction.* Here we generate a sequence of prices in a way that is reminiscent of real-life auctions. Strictly speaking, there is no primal or dual cost improvement here, although we will show that auction can be viewed as an approximate dual cost improvement process. In addition to prices, auction algorithms also iterate on flows, which are related to prices through a property called *ϵ-complementary slackness*; this is an approximate form of the complementary slackness property mentioned above.

All of the preceding types of algorithms can be used to solve both linear and convex network problems (although the structure of the given problem may favor significantly the use of some types of methods over others). For simplicity, in this chapter we will explain these ideas primarily through the assignment problem, deferring a more detailed development to subsequent chapters. Our illustrations, however, are relevant to the general minimum cost flow problem and to its convex cost extensions. Some of our explanations are informal. Precise statements of algorithms and results will be given in subsequent chapters.

1.3.1 Primal Cost Improvement

Primal cost improvement algorithms for the minimum cost flow problem start from an initial feasible flow vector and then generate a sequence of feasible flow vectors, each having a better cost than the preceding one. Let us derive an important characterization of the differences between successive vectors, which is the basis for algorithms as well as for optimality conditions.

Let x and $\overline{x}$ be two feasible flow vectors, and consider their difference $z = \overline{x} - x$. This difference must be a circulation with components

$$z_{ij} = \overline{x}_{ij} - x_{ij},$$

since both x and $\overline{x}$ are feasible. Furthermore, if the cost of $\overline{x}$ is smaller than the cost of x, the circulation z must have negative cost, i.e.,

$$\sum_{(i,j)\in\mathcal{A}} a_{ij} z_{ij} < 0.$$

We can decompose z into the sum of simple cycle flows by using the conformal realization theorem (Prop. 1.1). In particular, for some positive integer K, we have

$$z = \sum_{k=1}^{K} w^k \xi^k,$$

where w^k are positive scalars, and ξ^k are simple cycle flows whose nonzero components ξ_{ij}^k are 1 or -1, depending on whether $z_{ij} > 0$ or $z_{ij} < 0$, respectively. It is seen that the cost of z is

$$\sum_{(i,j)\in\mathcal{A}} a_{ij} z_{ij} = \sum_{k=1}^{K} w^k c^k,$$

where c^k is the cost of the simple cycle flow ξ^k. Thus, since the scalars w^k are positive, if the cost of z is negative, at least one c^k must be negative. Note that if C_k is the cycle corresponding to ξ^k, we have

$$c^k = \sum_{(i,j)\in\mathcal{A}} a_{ij}\xi_{ij}^k = \sum_{(i,j)\in C_k^+} a_{ij} - \sum_{(i,j)\in C_k^-} a_{ij},$$

where C_k^+ and C_k^- are the sets of forward and backward arcs of the cycle C_k, respectively. We refer to the expression in the right-hand side above as the *cost of the cycle* C_k.

The preceding argument has shown that *if x is feasible but not optimal, and $\overline{x}$ is feasible and has smaller cost than x, then at least one of the cycles corresponding to a conformal decomposition of the circulation $\overline{x} - x$ as above has negative cost*. This is used to prove the following important optimality condition.

Proposition 1.2: Consider the minimum cost flow problem. A flow vector x^* is optimal if and only if x^* is feasible and every simple cycle C that is unblocked with respect to x^* has nonnegative cost; that is,

$$\sum_{(i,j)\in C^+} a_{ij} - \sum_{(i,j)\in C^-} a_{ij} \geq 0.$$

Proof: Let x^* be an optimal flow vector and let C be a simple cycle that is unblocked with respect to x^*. Then there exists an $\epsilon > 0$ such that increasing (decreasing) the flow of arcs of C^+ (of C^-, respectively) by ϵ results in a feasible flow that has cost equal to the cost of x^* plus ϵ times the cost of C. Thus, since x^* is optimal, the cost of C must be nonnegative.

Conversely, suppose, to arrive at a contradiction, that x^* is feasible and has the nonnegative cycle property stated in the proposition, but is not optimal. Let $\overline{x}$ be a feasible flow vector with cost smaller that the one of x^*, and consider a conformal decomposition of the circulation $z = \overline{x} - x^*$. From the discussion preceding the proposition, we see that there is a simple cycle C with negative cost, such that $x^*_{ij} < \overline{x}_{ij}$ for all $(i,j) \in C^+$, and such that $x^*_{ij} > \overline{x}_{ij}$ for all $(i,j) \in C^-$. Since $\overline{x}$ is feasible, we have $b_{ij} \leq \overline{x}_{ij} \leq c_{ij}$ for all (i,j). It follows that $x^*_{ij} < c_{ij}$ for all $(i,j) \in C^+$, and $x^*_{ij} > b_{ij}$ for all $(i,j) \in C^-$, so that C is unblocked with respect to x^*. This contradicts the hypothesis that every simple cycle that is unblocked with respect to x^* has nonnegative cost. **Q.E.D.**

Most primal cost improvement algorithms (including for example the simplex method, to be discussed in Chapter 5) are based on the preceding proposition. They employ various mechanisms to construct negative cost cycles along which flow is pushed without violating the bound constraints. The idea of improving the cost by pushing flow along a suitable cycle often has an intuitive meaning as we illustrate in the context of the assignment problem.

Example 1.7. Multi-Person Exchanges in Assignment

Consider the $n \times n$ assignment problem (cf. Example 1.2) and suppose that we have a feasible assignment, that is, a set of n pairs (i,j) involving each person i exactly once and each object j exactly once. In order to improve this assignment, we could consider a *two-person exchange*, that is, replacing two pairs (i_1, j_1) and (i_2, j_2) from the assignment with the pairs (i_1, j_2) and (i_2, j_1). The resulting assignment will still be feasible, and it will have a higher value if and only if

$$a_{i_1 j_2} + a_{i_2 j_1} > a_{i_1 j_1} + a_{i_2 j_2}.$$

We note here that, in the context of the minimum cost flow representation of the assignment problem, a two-person exchange can be identified with a cycle involving the four arcs (i_1, j_1), (i_2, j_2), (i_1, j_2), and (i_2, j_1). Furthermore, this cycle is the difference between the assignment before and the assignment after the exchange, while the preceding inequality is equivalent to the cycle having a positive value.

Unfortunately, it may be impossible to improve the current assignment by a two-person exchange, even if the assignment is not optimal; see Fig. 1.8. An improvement, however, is possible by means of a *k-person exchange*, for some $k \geq 2$, where a set of pairs $(i_1, j_1), \ldots, (i_k, j_k)$ from the current assignment is replaced by the pairs $(i_1, j_2), \ldots, (i_{k-1}, j_k), (i_k, j_1)$. To see this,

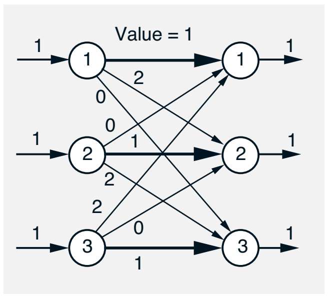


Figure 1.8: An example of a nonoptimal feasible assignment that cannot be improved by a two-person exchange. The value of each pair is shown next to the corresponding arc. Here, the value of the assignment $\{(1,1),(2,2),(3,3)\}$ is left unchanged at 3 by any two-person exchange. Through a three-person exchange, however, we obtain the optimal assignment, $\{(1,2),(2,3),(3,1)\}$, which has value 6.

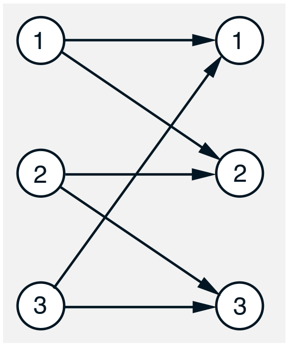


Figure 1.9: Illustration of the correspondence of a k-person exchange to a simple cycle. This is the same example as in the preceding figure. The backward arcs of the cycle are $(1,1)$, $(2,2)$, and $(3,3)$, and correspond to the current assignment pairs. The forward arcs of the cycle are $(1,2)$, $(2,3)$, and $(3,1)$, and correspond to the new assignment pairs. This three-person exchange is value-improving because the sum of the values of the forward arcs $(2+2+2)$ is greater than the sum of the values of the backward arcs $(1+1+1)$.

note that in the context of the minimum cost flow representation of the assignment problem, a k-person exchange corresponds to a simple cycle with k forward arcs (corresponding to the new assignment pairs) and k backward arcs (corresponding to the current assignment pairs that are being replaced); see Fig. 1.9. Thus, performing a k-person exchange is equivalent to pushing one unit of flow along the corresponding simple cycle. The k-person exchange improves the assignment if and only if

$$a_{i_k j_1} + \sum_{m=1}^{k-1} a_{i_m j_{m+1}} - \sum_{m=1}^{k} a_{i_m j_m},$$

which is equivalent to the corresponding cycle having positive value. Furthermore, by Prop. 1.2, a cost improving cycle exists if the flow corresponding to the current assignment is not optimal.

1.3.2 Dual Cost Improvement

Duality theory deals with the relation between the original network optimization problem and another optimization problem called the *dual*. To develop an intuitive understanding of duality, we will focus on an $n \times n$ assignment problem (cf. Example 1.2) and consider a closely related economic equilibrium problem. In particular, let us consider matching the n objects with the n persons through a market mechanism, viewing each person as an economic agent acting in his/her own best interest. Suppose that object j has a price p_j and that the person who receives the object must pay the price p_j. Then the net value of object j for person i is $a_{ij} - p_j$, and each person i will logically want to be assigned to an object j_i with maximal value, that is, with

$$a_{ij_i} - p_{j_i} = \max_{j \in A(i)} \{a_{ij} - p_j\}, \tag{1.10}$$

where

$$A(i) = \big\{j \mid (i,j) \in \mathcal{A}\big\}$$

is the set of objects that can be assigned to person i. When this condition holds for all persons i, we say that the assignment and the price vector $p = (p_1, \ldots, p_n)$ satisfy *complementary slackness* (CS for short); this name is standard in linear programming. The economic system is then at equilibrium, in the sense that no person would have an incentive to unilaterally seek another object. Such equilibrium conditions are naturally of great interest to economists, but there is also a fundamental relation with the assignment problem. We have the following proposition.

Proposition 1.3: If a feasible assignment and a set of prices satisfy the complementary slackness condition (1.10) for all persons i, then the assignment is optimal and the prices are an optimal solution of a dual problem, which is to minimize over $p = (p_1, \ldots, p_n)$ the cost function

$$\sum_{i=1}^{n} q_i(p) + \sum_{j=1}^{n} p_j,$$

where the functions q_i are given by

$$q_i(p) = \max_{j \in A(i)} \big\{a_{ij} - p_j\big\}, \qquad i = 1, \ldots, n.$$

Furthermore, the value of the optimal assignment and the optimal cost of the dual problem are equal.

Proof: The total value of any feasible assignment $\{(i, k_i) \mid i = 1, \ldots, n\}$ satisfies

$$\sum_{i=1}^{n} a_{ik_i} \le \sum_{i=1}^{n} \max_{j \in A(i)} \big\{a_{ij} - p_j\big\} + \sum_{j=1}^{n} p_j, \tag{1.11}$$

for any set of prices $\{p_j \mid j = 1, \ldots, n\}$, since the first term of the right-hand side is no less than

$$\sum_{i=1}^{n} (a_{ik_i} - p_{k_i}),$$

while the second term is equal to $\sum_{i=1}^{n} p_{k_i}$. On the other hand, the given assignment and set of prices, denoted by $\{(i, j_i) \mid i = 1, \ldots, n\}$ and $\{\overline{p}_j \mid j = 1, \ldots, n\}$, respectively, satisfy the CS conditions, so we have

$$a_{ij_i} - \overline{p}_{j_i} = \max_{j \in A(i)} \{a_{ij} - \overline{p}_j\}, \qquad i = 1, \ldots, n.$$

By adding this relation over all i, we have

$$\sum_{i=1}^{n} \left(\max_{j \in A(i)} \{a_{ij} - \overline{p}_j\} + \overline{p}_{j_i} \right) = \sum_{i=1}^{n} a_{ij_i}$$

and by using Eq. (1.11), we obtain

$$\begin{aligned} \sum_{i=1}^{n} a_{ik_i} &\leq \sum_{i=1}^{n} \left(\max_{j \in A(i)} \{a_{ij} - \overline{p}_j\} + \overline{p}_{j_i} \right) \\ &= \sum_{i=1}^{n} a_{ij_i} \\ &\leq \sum_{i=1}^{n} \max_{j \in A(i)} \{a_{ij} - p_j\} + \sum_{j=1}^{n} p_j, \end{aligned}$$

for every feasible assignment $\{(i, k_i) \mid i = 1, \ldots, n\}$ and every set of prices $\{p_j \mid j = 1, \ldots, n\}$. Therefore, the assignment $\{(i, j_i) \mid i = 1, \ldots, n\}$ is optimal for the primal problem, and the set of prices $\{\overline{p}_j \mid j = 1, \ldots, n\}$ is optimal for the dual problem. Furthermore, the two optimal values are equal. **Q.E.D.**

In analogy with primal cost improvement algorithms, one may start with a price vector and try to successively obtain new price vectors with improved dual cost. The major algorithms of this type involve price changes of the form

$$p_i := \begin{cases} p_i + \gamma & \text{if } i \in \mathcal{S}, \\ p_i & \text{if } i \notin \mathcal{S}, \end{cases} \tag{1.12}$$

where $\mathcal{S}$ is a connected subset of nodes, and γ is some positive scalar that is small enough to ensure that the new price vector has an improved dual cost.

The existence of a node subset $\mathcal{S}$ that results in cost improvement at a nonoptimal price vector, as described above, will be shown in Chapter 6.

This is an important and remarkable result, which may be viewed as a dual version of the result of Prop. 1.2 (at a nonoptimal flow vector, there exists at least one unblocked simple cycle with negative cost). In fact both results are special cases of a more general theorem concerning elementary vectors of subspaces, which is central in the theory of *monotropic programming* (see Chapter 9).

Most dual cost improvement methods, simultaneously with changing p along a direction of dual cost improvement, also iterate on a flow vector x satisfying CS together with p. They terminate when x becomes feasible, at which time, by Prop. 1.3, the pair (x, p) must consist of a primal and a dual optimal solution.

In Chapter 6 we will discuss two main methods that select subsets $\mathcal{S}$ and corresponding directions of dual cost improvement in different ways:

(a) In the *primal-dual method*, the direction has a *steepest ascent property*, that is, it provides the maximal rate of improvement of the dual cost per unit change in the price vector.

(b) In the *relaxation (or coordinate ascent) method*, the direction is computed so that it has a small number of nonzero elements (i.e., the set $\mathcal{S}$ has few nodes). Such a direction may not be optimal in terms of rate of dual cost improvement, but can typically be computed much faster than the steepest ascent direction. Often the direction has only one nonzero element, in which case only one node price coordinate is changed; this motivates the name "coordinate ascent." Note, however, that coordinate ascent directions cannot be used exclusively to improve the dual cost, as is shown in Fig. 1.10.

1.3.3 Auction

Our third type of algorithm represents a significant departure from the cost improvement idea; at any one iteration, it may deteriorate both the primal and the dual cost, although in the end it does find an optimal primal solution. It is based on an approximate version of complementary slackness, called *ϵ-complementary slackness*, and while it implicitly tries to solve a dual problem, it actually attains a dual solution that is not quite optimal. This subsection introduces the main ideas underlying auction algorithms. Chapters 7 and 9 provide a detailed discussion for the minimum cost flow problem and for the separable convex cost problem, respectively.

Naive Auction

Let us return to the assignment problem, and consider a natural process for finding an equilibrium assignment and price vector. We will call this process the *naive auction algorithm*, because it has a serious flaw, as will be

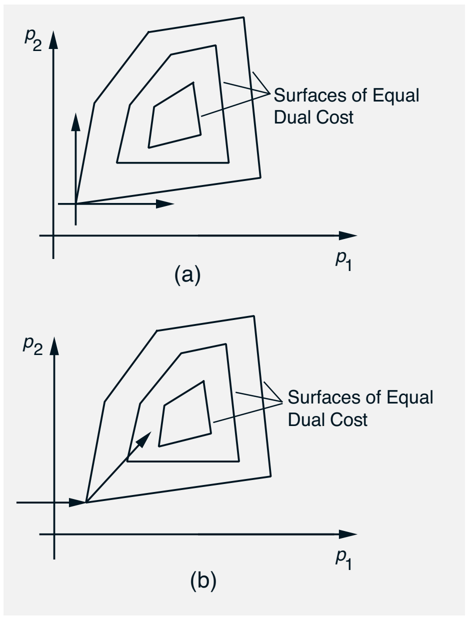


Figure 1.10: (a) The difficulty with using exclusively coordinate ascent iterations to solve the dual problem. Because the dual cost is piecewise linear, it may be impossible to improve it at some corner points by changing any *single* price coordinate. (b) As will be discussed in Chapter 6, a dual cost improvement is possible by changing several price coordinates by equal amounts, as in Eq. (1.12).

seen shortly. Nonetheless, this flaw will help motivate a more sophisticated and correct algorithm.

The naive auction algorithm proceeds in iterations and generates a sequence of price vectors and partial assignments. By a *partial assignment* we mean an assignment where only a subset of the persons have been matched with objects. A partial assignment should be contrasted with a *feasible* or *complete* assignment where all the persons have been matched with objects on a one-to-one basis. At the beginning of each iteration, the CS condition [cf. Eq. (1.10)]

$$a_{ij_i} - p_{j_i} = \max_{j \in A(i)} \{a_{ij} - p_j\}$$

is satisfied for all pairs (i, j_i) of the partial assignment. If all persons are assigned, the algorithm terminates. Otherwise some person who is unassigned, say i, is selected. This person finds an object j_i which offers maximal value, that is,

$$j_i = \arg \max_{j \in A(i)} \{a_{ij} - p_j\},$$

and then:

(a) Gets assigned to the best object j_i; the person who was assigned to j_i at the beginning of the iteration (if any) becomes unassigned.

(b) Sets the price of j_i to the level at which he/she is indifferent between j_i and the second best object; that is, he/she sets p_{j_i} to

$$p_{j_i} + \gamma_i,$$

where

$$\gamma_i = v_i - w_i, \tag{1.13}$$

v_i is the best object value,

$$v_i = \max_{j \in A(i)} \{a_{ij} - p_j\}, \tag{1.14}$$

and w_i is the second best object value,

$$w_i = \max_{j \in A(i),\, j \neq j_i} \{a_{ij} - p_j\}. \tag{1.15}$$

(Note that as p_{j_i} is increased, the value $a_{ij_i} - p_{j_i}$ offered by object j_i to person i is decreased. γ_i is the largest increment by which p_{j_i} can be increased, while maintaining the property that j_i offers maximal value to i.)

This process is repeated in a sequence of iterations until each person has been assigned to an object.

We may view this process as an auction where at each iteration the bidder i raises the price of a preferred object by the *bidding increment* γ_i. Note that γ_i cannot be negative, since $v_i \geq w_i$ [compare Eqs. (1.14)and (1.15)], so the object prices tend to increase. The choice γ_i is illustrated in Fig. 1.11. Just as in a real auction, bidding increments and price increases spur competition by making the bidder's own preferred object less attractive to other potential bidders.

ϵ-Complementary Slackness

Unfortunately, the naive auction algorithm does not always work (although it is an excellent initialization procedure for other methods, such as primal-dual or relaxation, and it is useful in other specialized contexts). The difficulty is that the bidding increment γ_i is 0 when two or more objects are tied in offering maximum value for the bidder i. As a result, a situation may be created where several persons contest a smaller number of equally desirable objects without raising their prices, thereby creating a never ending cycle; see Fig. 1.12.

To break such cycles, we introduce a perturbation mechanism, motivated by real auctions where each bid for an object must raise its price by a minimum positive increment, and bidders must on occasion take risks to win their preferred objects. In particular, let us fix a positive scalar ϵ, and

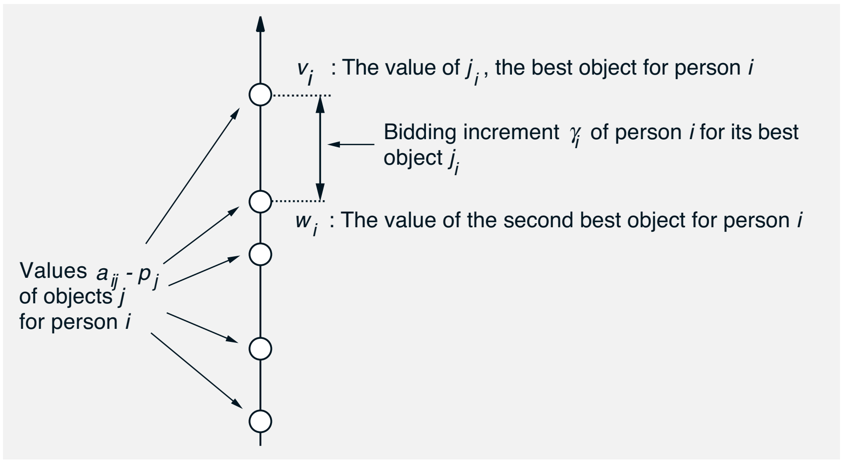

v_i : The value of j_i, the best object for person i
Bidding increment γ_i of person i for its best object j_i
w_i : The value of the second best object for person i
Values a_{ij} - p_j of objects j for person i

Figure 1.11: In the naive auction algorithm, even after the price of the best object j_i is increased by the bidding increment γ_i, j_i continues to be the best object for the bidder i, so CS is satisfied at the end of the iteration. However, we have $\gamma_i = 0$ if there is a tie between two or more objects that are most preferred by i.

say that a partial assignment and a price vector p satisfy *ϵ-complementary slackness (ϵ-CS for short)* if

$$a_{ij} - p_j \geq \max_{k \in A(i)} \{a_{ik} - p_k\} - \epsilon$$

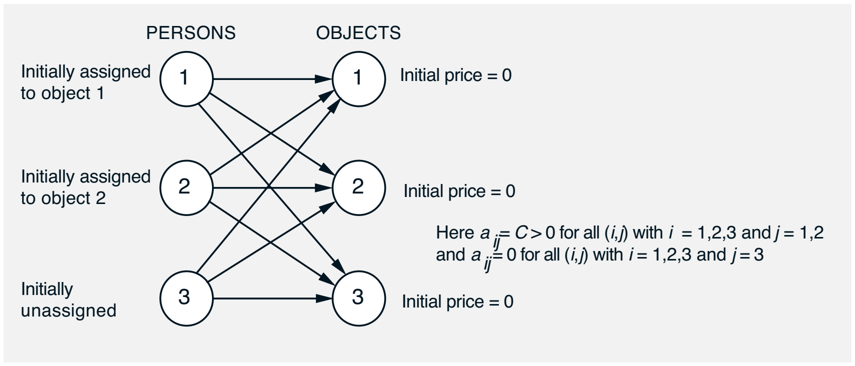
for all assigned pairs (i, j). In words, to satisfy ϵ-CS, all assigned persons of the partial assignment must be assigned to objects that are within ϵ of being best.

The Auction Algorithm

We now reformulate the previous auction process so that the bidding increment is always at least equal to ϵ. The resulting method, the *auction algorithm*, is the same as the naive auction algorithm, except that the bidding increment γ_i is

$$\gamma_i = v_i - w_i + \epsilon \tag{1.16}$$

rather than $\gamma_i = v_i - w_i$ as in Eq. (1.13). With this choice, the ϵ-CS condition is satisfied, as illustrated in Fig. 1.13. The particular increment $\gamma_i = v_i - w_i + \epsilon$ used in the auction algorithm is the maximum amount with this property. Smaller increments γ_i would also work as long as $\gamma_i \geq \epsilon$, but using the largest possible increment accelerates the algorithm. This is consistent with experience from real auctions, which tend to terminate faster when the bidding is aggressive.


PERSONS
OBJECTS
Initially assigned to object 1
Initially assigned to object 2
Initially unassigned
Initial price = 0
Initial price = 0
Initial price = 0
Here $a_{ij} = C > 0$ for all (i,j) with $i = 1,2,3$ and $j = 1,2$ and $a_{ij} = 0$ for all (i,j) with $i = 1,2,3$ and $j = 3$

At Start of Iteration #	Object Prices	Assigned Pairs	Bidder	Preferred Object	Bidding Increment
1	0,0,0	(1,1), (2,2)	3	2	0
2	0,0,0	(1,1), (3,2)	2	2	0
3	0,0,0	(1,1), (2,2)	3	2	0

Figure 1.12: Illustration of how the naive auction algorithm may never terminate for a problem involving three persons and three objects. Here objects 1 and 2 offer benefit $C > 0$ to all persons, and object 3 offers benefit 0 to all persons. The algorithm cycles as persons 2 and 3 alternately bid for object 2 without changing its price because they prefer equally object 1 and object 2.

It can be shown that this reformulated auction process terminates, necessarily with a feasible assignment and a set of prices that satisfy ϵ-CS. To get a sense of this, note that if an object receives a bid during m iterations, its price must exceed its initial price by at least $m\epsilon$. Thus, for sufficiently large m, the object will become "expensive" enough to be judged "inferior" to some object that has not received a bid so far. It follows that only for a limited number of iterations can an object receive a bid while some other object still has not yet received any bid. On the other hand, once every object has received at least one bid, the auction terminates. (This argument assumes that any person can bid for any object, but it can be generalized to the case where the set of feasible person-object pairs is limited, as long as at least one feasible assignment exists; see Prop. 7.2 in Chapter 7.) Figure 1.14 shows how the auction algorithm, based on the bidding increment $\gamma_i = v_i - w_i + \epsilon$ [see Eq. (1.16)], overcomes the cycling difficulty in the example of Fig. 1.12.

When the auction algorithm terminates, we have an assignment satisfying ϵ-CS, but is this assignment optimal? The answer depends strongly

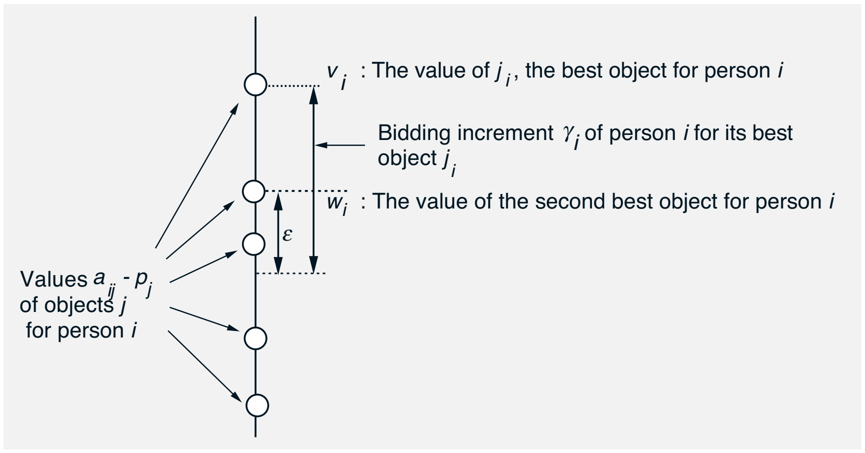

v_i : The value of j_i, the best object for person i
Bidding increment γ_i of person i for its best object j_i
w_i : The value of the second best object for person i
ε
Values $a_{ij} - p_j$ of objects j for person i

Figure 1.13: In the auction algorithm, even after the price of the preferred object j_i is increased by the bidding increment γ_i, j_i will be within ϵ of being most preferred, so the ϵ-CS condition holds at the end of the iteration.

on the size of ϵ. In a real auction, a prudent bidder would not place an excessively high bid for fear of winning the object at an unnecessarily high price. Consistent with this intuition, we can show that if ϵ is small, then the final assignment will be "almost optimal." In particular, we will show that *the total benefit of the final assignment is within $n\epsilon$ of being optimal*. The idea is that a feasible assignment and a set of prices satisfying ϵ-CS may be viewed as satisfying CS for a *slightly different* problem, where all benefits a_{ij} are the same as before except the benefits of the n assigned pairs, which are modified by no more than ϵ.

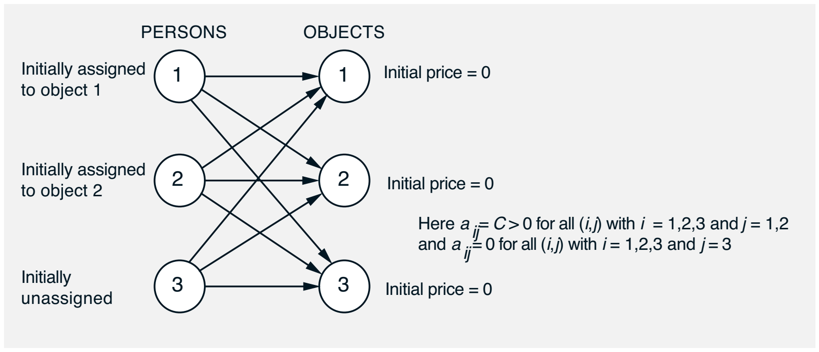
Proposition 1.4: A feasible assignment satisfying ϵ-complementary slackness, together with some price vector, attains within $n\epsilon$ the optimal primal value. Furthermore, the price vector attains within $n\epsilon$ the optimal dual cost.

Proof: Let A^* be the optimal total assignment benefit

$$A^* = \max_{\substack{k_i,\, i=1,\ldots,n \\ k_i \neq k_m \text{ if } i \neq m}} \sum_{i=1}^{n} a_{ik_i}$$

and let D^* be the optimal dual cost (cf. Prop. 1.3):

$$D^* = \min_{\substack{p_j \\ j=1,\ldots,n}} \left\{ \sum_{i=1}^{n} \max_{j \in A(i)} \{a_{ij} - p_j\} + \sum_{j=1}^{n} p_j \right\}.$$


PERSONS
OBJECTS
Initially assigned to object 1
Initially assigned to object 2
Initially unassigned
Initial price = 0
Initial price = 0
Initial price = 0
Here $a_{ij} = C > 0$ for all (i,j) with $i = 1,2,3$ and $j = 1,2$
and $a_{ij} = 0$ for all (i,j) with $i = 1,2,3$ and $j = 3$

At Start of Iteration #	Object Prices	Assigned Pairs	Bidder	Preferred Object	Bidding Increment
1	0,0,0	(1,1), (2,2)	3	2	ϵ
2	0,ϵ,0	(1,1), (3,2)	2	1	2ϵ
3	2ϵ,ϵ,0	(2,1), (3,2)	1	2	2ϵ
4	2ϵ,3ϵ,0	(1,2), (2,1)	3	1	2ϵ
5	4ϵ,3ϵ,0	(1,2), (3,1)	2	2	2ϵ
6	...	...	...	...	...

Figure 1.14: Illustration of how the auction algorithm, by making the bidding increment at least ϵ, overcomes the cycling difficulty for the example of Fig. 1.12. The table shows one possible sequence of bids and assignments generated by the auction algorithm, starting with all prices equal to 0 and with the partial assignment $\{(1,1),(2,2)\}$. At each iteration except the last, the person assigned to object 3 bids for either object 1 or 2, increasing its price by ϵ in the first iteration and by 2ϵ in each subsequent iteration. In the last iteration, after the prices of 1 and 2 reach or exceed C, object 3 receives a bid and the auction terminates.

If $\{(i,j_i) \mid i = 1,\ldots,n\}$ is the given assignment satisfying the ϵ-CS condition together with a price vector $\overline{p}$, we have

$$\max_{j\in A(i)} \{a_{ij} - \overline{p}_j\} - \epsilon \le a_{ij_i} - \overline{p}_{j_i}.$$

By adding this relation over all i, we see that

$$D^* \le \sum_{i=1}^{n} \left(\max_{j\in A(i)} \{a_{ij} - \overline{p}_j\} + \overline{p}_{j_i} \right) \le \sum_{i=1}^{n} a_{ij_i} + n\epsilon \le A^* + n\epsilon.$$

Since we showed in Prop. 1.3 that $A^* = D^*$, it follows that the total assignment benefit $\sum_{i=1}^{n} a_{ij_i}$ is within $n\epsilon$ of the optimal value A^*, while the dual cost of $\overline{p}$ is within $n\epsilon$ of the optimal dual cost. **Q.E.D.**

Suppose now that the benefits a_{ij} are all integer, which is the typical practical case. (If a_{ij} are rational numbers, they can be scaled up to integer by multiplication with a suitable common number.) Then the total benefit of any assignment is integer, so if $n\epsilon < 1$, any complete assignment that is within $n\epsilon$ of being optimal must be optimal. It follows that *if*

$$\epsilon < \frac{1}{n}$$

and the benefits a_{ij} are all integer, then the assignment obtained upon termination of the auction algorithm is optimal.

Figure 1.15 shows the sequence of generated object prices for the example of Fig. 1.12 in relation to the contours of the dual cost function. It can be seen from this figure that each bid has the effect of setting the price of the object receiving the bid nearly equal (within ϵ) to the price that minimizes the dual cost with respect to that price, with all other prices held fixed (this will be shown rigorously in Section 7.1). Successive minimization of a cost function along single coordinates is a central feature of coordinate descent and relaxation methods, which are popular for unconstrained minimization of smooth functions and for solving systems of smooth equations. Thus, the auction algorithm can be interpreted as an approximate coordinate descent method; as such, it is related to the relaxation method discussed in the previous subsection.

Scaling

Figure 1.15 also illustrates a generic feature of auction algorithms. The amount of work needed to solve the problem can depend strongly on the value of ϵ and on the maximum absolute object benefit

$$C = \max_{(i,j)\in\mathcal{A}} |a_{ij}|.$$

Basically, for many types of problems, the number of iterations up to termination tends to be proportional to C/ϵ. This can be seen from the figure, where the total number of iterations is roughly C/ϵ, starting from zero initial prices.

Note also that there is a dependence on the initial prices; if these prices are “near optimal,” we expect that the number of iterations needed to solve the problem will be relatively small. This can be seen from Fig. 1.15; if the initial prices satisfy $p_1 \approx p_3 + C$ and $p_2 \approx p_3 + C$, the number of iterations up to termination is quite small.

The preceding observations suggest the idea of *ϵ-scaling*, which consists of applying the algorithm several times, starting with a large value of ϵ and successively reducing ϵ until it is less than some critical value (for example, $1/n$, when a_{ij} are integer). Each application of the algorithm provides good initial prices for the next application. This is a common idea

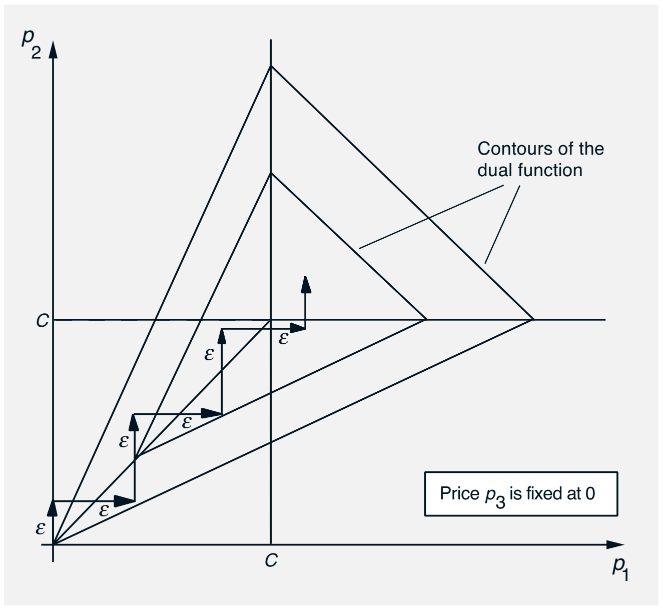


Figure 1.15: A sequence of prices p_1 and p_2 generated by the auction algorithm for the example of Figs. 1.12 and 1.14. The figure shows the equal dual cost surfaces in the space of p_1 and p_2, with p_3 fixed at 0. The arrows indicate the price iterates as given by the table of Fig. 1.14. Termination occurs when the prices reach an ϵ-neighborhood of the point (C, C), and object 3 becomes “sufficiently inexpensive” to receive a bid and to get assigned. The total number of iterations is roughly C/ϵ, starting from zero initial prices.

in nonlinear programming; it is encountered, for example, in barrier and penalty function methods (see Section 8.8). In practice, scaling is typically beneficial, and accelerates the termination of the auction algorithm.

1.3.4 Good, Bad, and Polynomial Algorithms

We have discussed several types of methods, so the natural question arises: is there a best method and what criterion should we use to rank methods?

A practitioner who has a specific type of problem to solve, perhaps repeatedly, with the data and size of the problem within some limited range, will usually be interested in one or more of the following:

(a) Fast solution time.

(b) Flexibility to use good starting solutions (which the practitioner can usually provide, based on his/her knowledge of the problem, or based on a known solution of some similar problem).

(c) The ability to perform sensitivity analysis (resolve the problem with slightly different problem data) quickly.

(d) The ability to take advantage of parallel computing hardware.

Given the diversity of these considerations, it is not surprising that there is no algorithm that will dominate the others in all or even most practical situations. Otherwise expressed, every type of algorithm that we will discuss is best given the right type of practical situation. Thus, to make intelligent choices, the practitioner needs to understand the properties of different algorithms relating to speed of convergence, flexibility, parallelization, and suitability for specific problem structures. For challenging problems, the choice of algorithm is often settled by experimentation with several candidates.

A theoretical analyst may also have difficulty ranking different algorithms for specific types of problems. The most common approach for this purpose is worst-case computational complexity analysis. For example, for the minimum cost flow problem, one tries to bound the number of elementary numerical operations needed by a given algorithm with some measure of the "problem size," that is, with some expression of the form

$$Kf(N, A, C, U, S),$$

where

N is the number of nodes,

A is the number of arcs,

C is the arc cost range $\max_{(i,j)\in\mathcal{A}} |a_{ij}|$,

U is the maximum arc flow range $\max_{(i,j)\in\mathcal{A}}(c_{ij} - b_{ij})$,

S is the supply range $\max_{i\in\mathcal{N}} |s_i|$,

f is some known function,

K is a (usually unknown) constant.

If a bound of this form can be found, we say that the *running time* or *operation count of the algorithm is* $O\big(f(N, A, C, U, S)\big)$. If $f(N, A, C, U, S)$ can be written as a polynomial function of the number of bits needed to express the problem data, the algorithm is said to be *polynomial*. Examples of polynomial complexity bounds are $O\big(N^\alpha A^\beta\big)$ and $O\big(N^\alpha A^\beta \log C\big)$, where α and β are positive integers, and the numbers a_{ij} are assumed integer. The bound $O\big(N^\alpha A^\beta\big)$ is sometimes said to be *strongly polynomial* because it involves only the graph size parameters. A bound of the form $O\big(N^\alpha A^\beta C\big)$ is not polynomial, even assuming that the a_{ij} are integer, because C is not a polynomial expression of $\log C$, the number of bits needed to express a single number a_{ij}. Bounds like $O\big(N^\alpha A^\beta C\big)$, which are polynomial in the problem data rather than in the number of bits needed to express the data, are called *pseudopolynomial*.

A common assumption in theoretical computer science is that polynomial algorithms are "better" than pseudopolynomial, and pseudopolynomial algorithms are "better" than exponential [for example, those with a bound of the form $K2^{g(N,A)}$, where g is a polynomial in N and A]. Furthermore, it is thought that two polynomial algorithms can be compared in terms of the degree of the polynomial bound; e.g., an $O(N^2)$ algorithm is "better" than an $O(N^3)$ algorithm. Unfortunately, quite often this assumption is not supported by computational practice in linear programming and network optimization. Pseudopolynomial and even exponential algorithms are often faster in practice than polynomial ones. In fact, the simplex method for general linear programs is an exponential algorithm, as shown by Klee and Minty [1972] (see also the textbooks by Chvatal [1983], or Bertsimas and Tsitsiklis [1997]), and yet it is used widely, because of its excellent practical properties.

There are two main reasons why worst-case complexity estimates may fail to predict the practical performance of network flow algorithms. First, the estimates, even if they are tight, may be very pessimistic as they may correspond to problem instances that are highly unlikely in practice. (Average complexity estimates would be more appropriate for such situations. However, obtaining these is usually hard, and the statistical assumptions underlying them may be inappropriate for many types of practical problems.) Second, worst-case complexity estimates involve the (usually unknown) constant K, which may dominate the estimate for all except for unrealistically large problem sizes. Thus, a comparison between two algorithms that is based on the size-dependent terms of running time estimates, and does not take into account the corresponding constants may be unreliable.

Despite its shortcomings, computational complexity analysis is valuable because it often illuminates the computational bottlenecks of many algorithms and motivates the use of efficient data structures. For this reason, throughout the book, we will comment on available complexity results, we will prove some of the most important estimates, and we will try to relate these estimates to computational practice. For some classes of problems, however, it turns out that the methods with the best computational complexity are impractical, because they are either too complicated or too slow in practice. In such cases, we will refer to the literature, without providing a detailed discussion.

1.4 NOTES, SOURCES, AND EXERCISES

Network problems are discussed in many books (Berge [1962], Berge and Ghouila-Houri [1962], Ford and Fulkerson [1962], Dantzig [1963], Busacker and Saaty [1965], Hu [1969], Iri [1969], Frank and Frisch 1970], Christofides

[1975], Zoutendijk [1976], Minieka [1978], Jensen and Barnes [1980], Kennington and Helgason [1980], Papadimitriou and Steiglitz [1982], Chvatal [1983], Gondran and Minoux [1984], Luenberger [1984], Rockafellar [1984], Bazaraa, Jarvis, and Sherali [1990], Bertsekas [1991a], Murty [1992], Bertsimas and Tsitsiklis [1997]). Several of these books discuss linear programming first and develop linear network optimization as a special case. An alternative approach that relies heavily on duality, is given by Rockafellar [1984]. The conformal realization theorem (Prop. 1.1) has been developed in different forms in several sources, including Ford and Fulkerson [1962], Busacker and Saaty [1965], and Rockafellar [1984].

The primal cost improvement approach for network optimization was initiated by Dantzig [1951], who specialized the simplex method to the transportation problem. The extensive subsequent work using this approach is surveyed at the end of Chapter 5.

The dual cost improvement approach was initiated by Kuhn [1955] who proposed the *Hungarian method* for the assignment problem. (The name of the algorithm honors its connection with the research of the Hungarian mathematicians Egervary [1931] and König [1931].) Work using this approach is surveyed in Chapter 6.

The auction approach was initiated in Bertsekas [1979a] for the assignment problem, and in Bertsekas [1986a], [1986b] for the minimum cost flow problem. Work using this approach is surveyed at the end of Chapter 7.

E X E R C I S E S

1.1

Consider the graph and the flow vector of Fig. 1.16.

(a) Enumerate the simple paths and the simple forward paths that start at node 1.

(b) Enumerate the simple cycles and the simple forward cycles of the graph.

(c) Is the graph connected? Is it strongly connected?

(d) Calculate the divergences of all the nodes and verify that they add to 0.

(e) Give an example of a simple path flow that starts at node 1, ends at node 5, involves four arcs, and conforms to the given flow vector.

(f) Suppose that all arcs have arc flow bounds -1 and 5. Enumerate all the simple paths that start at node 1, end at node 5, and are unblocked with respect to the given flow vector.

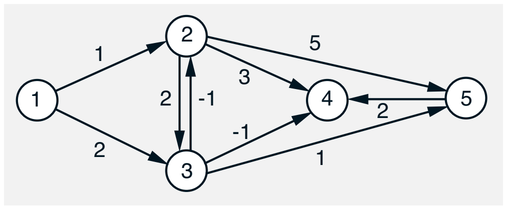


Figure 1.16: Flow vector for Exercise 1.1. The arc flows are the numbers shown next to the arcs.

1.2 (Proof of the Conformal Realization Theorem)

Prove the conformal realization theorem (Prop. 1.1) by completing the details of the following argument. Assume first that x is a circulation. Consider the following procedure by which given x, we obtain a simple cycle flow x' that conforms to x and satisfies

$$\begin{aligned}
&0 \le x'_{ij} \le x_{ij} && \text{for all arcs } (i,j) \text{ with } 0 \le x_{ij},\\
&x_{ij} \le x'_{ij} \le 0 && \text{for all arcs } (i,j) \text{ with } x_{ij} \le 0,\\
&x_{ij} = x'_{ij} && \text{for at least one arc } (i,j) \text{ with } x_{ij} \ne 0;
\end{aligned}$$

(see Fig. 1.17). Choose an arc (i,j) with $x_{ij} \ne 0$. Assume that $x_{ij} > 0$. (A similar procedure can be used when $x_{ij} < 0$.) Construct a sequence of node subsets $T_0, T_1, \ldots$, as follows: Take $T_0 = \{j\}$. For $k = 0, 1, \ldots$, given T_k, let

$$T_{k+1} = \Big\{ n \notin \cup_{p=0}^{k} T_p \mid \text{ there is a node } m \in T_k, \text{ and either an arc } (m,n) \\ \text{such that } x_{mn} > 0 \text{ or an arc } (n,m) \text{ such that } x_{nm} < 0 \Big\},$$

and mark each node $n \in T_{k+1}$ with the label "(m,n)" or "(n,m)," where m is a node of T_k such that $x_{mn} > 0$ or $x_{nm} < 0$, respectively. The procedure terminates when T_{k+1} is empty.

At the end of the procedure, trace labels backward from i until node j is reached. (How do we know that i belongs to one of the sets T_k?) In particular, let "(i_1, i)" or "(i, i_1)" be the label of i, let "(i_2, i_1)" or "(i_1, i_2)" be the label of i_1, etc., until a node i_k with label "(i_k, j)" or "(j, i_k)" is found. The cycle $C = (j, i_k, i_{k-1}, \ldots, i_1, i, j)$ is simple, it contains (i,j) as a forward arc, and is such that all its forward arcs have positive flow and all its backward arcs have negative flow. Let $a = \min_{(m,n)\in C} |x_{mn}| > 0$. Then the simple cycle flow x', where

$$x'_{ij} = \begin{cases} a & \text{if } (i,j) \in C^+,\\ -a & \text{if } (i,j) \in C^-,\\ 0 & \text{otherwise,} \end{cases}$$

has the required properties.

Now subtract x' from x. We have $x_{ij} - x'_{ij} > 0$ only for arcs (i,j) with $x_{ij} > 0$, $x_{ij} - x'_{ij} < 0$ only for arcs (i,j) with $x_{ij} < 0$, and $x_{ij} - x'_{ij} = 0$ for at

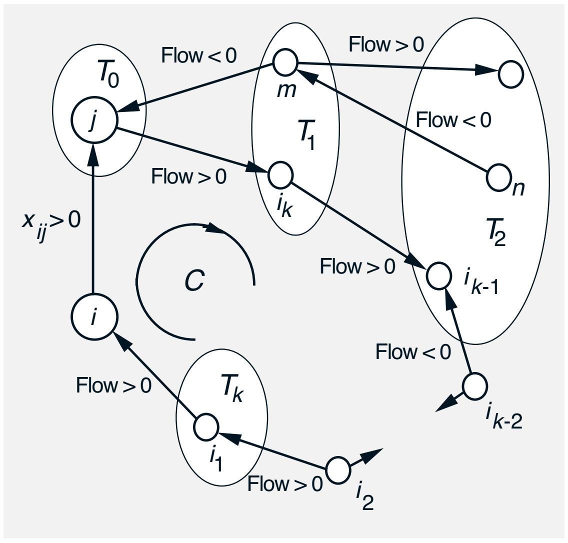


Figure 1.17: Construction of a cycle of arcs with nonzero flow used in the proof of the conformal realization theorem.

least one arc (i, j) with $x_{ij} \neq 0$. If x is integer, then x' and $x - x'$ will also be integer. We then repeat the process (for at most A times) with the circulation x replaced by the circulation $x - x'$ and so on, until the zero flow is obtained.

If x is not a circulation, we form an enlarged graph by introducing a new node s and by introducing for each node $i \in \mathcal{N}$ an arc (s, i) with flow x_{si} equal to the divergence y_i. The resulting flow vector is seen to be a circulation in the enlarged graph (why?). This circulation, by the result just shown, can be decomposed into at most $A + N$ simple cycle flows of the enlarged graph, conforming to the flow vector. Out of these cycle flows, we consider those containing node s, and we remove s and its two incident arcs while leaving the other cycle flows unchanged. As a result we obtain a set of at most $A+N$ path flows of the original graph, which add up to x. These path flows also conform to x, as required.

1.3

Use the algorithm of Exercise 1.2 to decompose the flow vector of Fig. 1.16 into conforming simple path flows.

1.4 (Path Decomposition Theorem)

(a) Use the conformal realization theorem (Prop. 1.1) to show that a forward path P can be decomposed into a (possibly empty) collection of simple forward cycles, together with a simple forward path that has the same start node and end node as P. (Here "decomposition" means that the union of the arcs of the component paths is equal to the set of arcs of P with the multiplicity of repeated arcs properly accounted for.)

(b) Suppose that a graph is strongly connected and that a length a_{ij} is given for every arc (i, j). Show that if all forward cycles have nonnegative length, then there exists a shortest path from any node s to any node t. Show also that if there exists a shortest path from some node s to some node t, then all forward cycles have nonnegative length. Why is the connectivity assumption needed?

1.5 (Cycle Decomposition - Euler Cycles)

Consider a graph such that each of the nodes has even degree.

(a) Give an algorithm to decompose the graph into a collection of simple cycles that are disjoint, in the sense that they share no arcs (although they may share some nodes). (Here "decomposition" means that the union of the arcs of the component cycles is equal to the set of arcs of the graph.) *Hint*: Given a connected graph where each of the nodes has even degree, the deletion of the arcs of any cycle creates some connected subgraphs where each of the nodes has even degree (including possibly some isolated nodes).

(b) Assume in addition that the graph is connected. Show that there is an Euler cycle, i.e., a cycle that contains all the arcs of a graph exactly once. *Hint*: Apply the decomposition of part (a), and successively merge an Euler cycle of a subgraph with a simple cycle.

1.6

In the graph of Fig. 1.16, consider the graph obtained by deleting node 1 and arcs $(1, 2)$, $(1, 3)$, and $(5, 4)$. Decompose this graph into a collection of simple cycles that are disjoint (cf. Exercise 1.5) and construct an Euler cycle.

1.7

(a) Consider an $n \times n$ chessboard, and a rook that is allowed to make the standard moves along the rows and columns. Show that the rook can start at a given square and return to that square after making each of the possible legal moves exactly once and in one direction only [of the two moves (a, b) and (b, a) only one should be made]. *Hint*: Construct an Euler cycle in a suitable graph.

(b) Consider an $n \times n$ chessboard with n even, and a bishop that is allowed to make two types of moves: legal moves (which are the standard moves along the diagonals of its color), and illegal moves (which go from any square of its color to any other square of its color). Show that the bishop can start at a given square and return to that square after making each of the possible legal moves exactly once and in one direction only, plus $n^2/4$ illegal moves.

For every square of its color, there should be exactly one illegal move that either starts or ends at that square.

1.8 (Forward Euler Cycles)

Consider a graph and the question whether there exists a forward cycle that passes through each arc of the graph exactly once. Show that such a cycle exists if and only if the graph is connected and the number of incoming arcs to each node is equal to the number of outgoing arcs from the node.

1.9

Consider an $n \times n$ chessboard with $n \geq 4$. Show that a knight starting at any square can visit every other square, with a move sequence that contains every possible move exactly once [a move (a, b) as well as its reverse (b, a) should be made]. Interpret this sequence as a forward Euler cycle in a suitable graph (cf. Exercise 1.8).

1.10 (Euler Paths)

Consider a graph and the question whether there exists a path that passes through each arc of the graph exactly once. Show that such a path exists if and only if the graph is connected, and either the degrees of all the nodes are even, or else the degrees of all the nodes except two are even.

1.11

In shatranj, the old version of chess, the firz (or vizier, the predecessor to the modern queen) can move one square diagonally in each direction. Show that starting at a corner of an $n \times n$ chessboard where n is even, the firz can reach the opposite corner after making each of the possible moves along its diagonals exactly once and in one direction only [of the two moves (a, b) and (b, a) only one should be made].

1.12

Show that the number of nodes with odd degree in a graph is even.

1.13

Assume that all the nodes of a graph have degree greater than one. Show that the graph must contain a cycle.

1.14

(a) Show that every tree with at least two nodes has at least two nodes with degree one.

(b) Show that a graph is a tree if and only if it is connected and the number of arcs is one less than the number of nodes.

1.15

Consider a volleyball net that consists of a mesh with m squares on the horizontal dimension and n squares on the vertical. What is the maximum number of strings that can be cut before the net falls apart into two pieces.

1.16 (Checking Connectivity)

Consider a graph with A arcs.

(a) Devise an algorithm with $O(A)$ running time that checks whether the graph is connected, and if it is connected, simultaneously constructs a path connecting any two nodes. *Hint*: Start at a node, mark its neighbors, and continue.

(b) Repeat part (a) for the case where we want to check strong connectedness.

(c) Devise an algorithm with $O(A)$ running time that checks whether there exists a cycle that contains two given nodes.

(d) Repeat part (c) for the case where the cycle is required to be forward.

1.17 (Inequality Constrained Minimum Cost Flows)

Consider the following variant of the minimum cost flow problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} \\ \text{subject to} \quad & \underline{s}_i \le \sum_{\{j\mid(i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j\mid(j,i)\in\mathcal{A}\}} x_{ji} \le \overline{s}_i, \qquad \forall\ i \in \mathcal{N}, \\ & b_{ij} \le x_{ij} \le c_{ij}, \qquad \forall\ (i,j) \in \mathcal{A}, \end{aligned}$$

where the bounds $\underline{s}_i$ and $\overline{s}_i$ on the divergence of node i are given. Show that this problem can be converted to a standard (equality constrained) minimum cost flow problem by adding an extra node A and an arc (A, i) from this node to every other node i, with feasible flow range $[0, \overline{s}_i - \underline{s}_i]$.

1.18 (Node Throughput Constraints)

Consider the minimum cost flow problem with the additional constraints that the total flow of the outgoing arcs from each node i must lie within a given range $[\underline{t}_i, \overline{t}_i]$, that is,

$$\underline{t}_i \le \sum_{\{j \mid (i,j)\in\mathcal{A}\}} x_{ij} \le \overline{t}_i.$$

Convert this problem into the standard form of the minimum cost flow problem by splitting each node into two nodes with a connecting arc.

1.19 (Piecewise Linear Arc Costs)

Consider the minimum cost flow problem with the difference that, instead of the linear form $a_{ij}x_{ij}$, each arc's cost function has the piecewise linear form

$$f_{ij}(x_{ij}) = \begin{cases} a_{ij}^1 x_{ij} & \text{if } b_{ij} \le x_{ij} \le m_{ij}, \\ a_{ij}^1 m_{ij} + a_{ij}^2 (x_{ij} - m_{ij}) & \text{if } m_{ij} \le x_{ij} \le c_{ij}, \end{cases}$$

where m_{ij}, a_{ij}^1, and a_{ij}^2 are given scalars satisfying $b_{ij} \le m_{ij} \le c_{ij}$ and $a_{ij}^1 \le a_{ij}^2$.

(a) Show that the problem can be converted to a linear minimum cost flow problem where each arc (i,j) is replaced by two arcs with arc cost coefficients a_{ij}^1 and a_{ij}^2, and arc flow ranges $[b_{ij}, m_{ij}]$ and $[0, c_{ij} - m_{ij}]$, respectively.

(b) Generalize to the case of piecewise linear cost functions with more than two pieces.

1.20 (Asymmetric Assignment and Transportation Problems)

Consider an assignment problem where the number of objects is larger than the number of persons, and we require that each person be assigned to one object. The associated linear program (cf. Example 1.2) is

$$\begin{aligned} &\text{maximize} \quad \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} \\ &\text{subject to} \quad \sum_{\{j \mid (i,j)\in\mathcal{A}\}} x_{ij} = 1, \qquad \forall\ i = 1,\ldots,m, \\ &\qquad\qquad \sum_{\{i \mid (i,j)\in\mathcal{A}\}} x_{ij} \le 1, \qquad \forall\ j = 1,\ldots,n, \\ &\qquad\qquad 0 \le x_{ij} \le 1, \qquad \forall\ (i,j)\in\mathcal{A}, \end{aligned}$$

where $m < n$.

(a) Show how to formulate this problem as a minimum cost flow problem by introducing extra arcs and nodes.

(b) Repeat part (a) for the case where there may be some persons that are left unassigned; that is, the constraint $\sum_{\{j \mid (i,j)\in\mathcal{A}\}} x_{ij} = 1$ is replaced by $\sum_{\{j \mid (i,j)\in\mathcal{A}\}} x_{ij} \leq 1$. Give an example of a problem with $a_{ij} > 0$ for all $(i,j) \in \mathcal{A}$, which is such that in the optimal assignment some persons are left unassigned, even though there exist feasible assignments that assign every person to some object.

(c) Formulate an asymmetric transportation problem where the total supply is less than the total demand, but some demand may be left unsatisfied, and appropriately modify your answers to parts (a) and (b).

1.21 (Bipartite Matching)

Bipartite matching problems are assignment problems where the coefficients (i,j) are all equal to 1. In such problems, we want to maximize the cardinality of the assignment, that is, the number of assigned pairs (i,j). Formulate a bipartite matching problem as an equivalent max-flow problem.

1.22 (Production Planning)

Consider a problem of scheduling production of a certain item to meet a given demand over N time periods. Let us denote:

x_i: The amount of product stored at the beginning of period i, where $i = 0, \ldots, N-1$. There is a nonnegativity constraint on x_i.

u_i: The amount of product produced during period i. There is a constraint $0 \leq u_i \leq c_i$, where the scalar c_i is given for each i.

d_i: The amount of product demanded during period i. This is a given scalar for each i.

The amount of product stored evolves according to the equation

$$x_{i+1} = x_i + u_i - d_i, \qquad i = 0, \ldots, N-1.$$

Given x_0, we want to find a feasible production sequence $\{u_0, \ldots, u_{N-1}\}$ that minimizes

$$\sum_{i=0}^{N-1} (a_i x_i + b_i u_i),$$

where a_i and b_i are given scalars for each i. Formulate this problem as a minimum cost flow problem. *Hint*: For each i, introduce a node that connects to a special artificial node.

1.23 (Capacity Expansion)

The capacity of a certain facility is to be expanded over N time periods by adding an increment $u_i \in [0, c_i]$ at time period $i = 0, \ldots, N-1$, where c_i is a given scalar. Thus, if x_i is the capacity at the beginning of period i, we have

$$x_{i+1} = x_i + u_i, \qquad i = 0, \ldots, N-1.$$

Given x_0, consider the problem of finding u_i, $i = 0, \ldots, N-1$, such that each x_i lies within a given interval $[\underline{x}_i, \bar{x}_i]$ and the cost

$$\sum_{i=0}^{N-1} (a_i x_i + b_i u_i)$$

is minimized, where a_i and b_i are given scalars for each i. Formulate the problem as a minimum cost flow problem.

1.24 (Dynamic Transhipment Problems)

Consider a transhipment context for the minimum cost flow problem where the problem is to optimally transfer flow from some supply points to some demand points over arcs of limited capacity. In a dynamic version of this context, the transfer is to be performed over N time units, and transferring flow along an arc (i, j) requires time τ_{ij}, which is a given positive integer number of time units. This means that at each time $t = 0, \ldots, N - \tau_{ij}$, we may send from node i along arc (i, j) a flow $x_{ij} \in [0, c_{ij}]$, which will arrive at node j at time $t + \tau_{ij}$. Formulate this problem as a minimum cost flow problem involving a copy of the given graph for each time period.

1.25 (Concentrator Assignment)

We have m communication terminals, each to be connected to one out of a given collection of concentrators. Suppose that there is a cost a_{ij} for connecting terminal i to concentrator j, and that each concentrator j has an upper bound b_j on the number of terminals it can be connected to. Also, each terminal i can be connected to only a given subset of concentrators.

(a) Formulate the problem of finding the minimum cost connection of terminals to concentrators as a minimum cost flow problem. *Hint*: You may use the fact that there exists an integer optimal solution to a minimum cost flow problem with integer supplies and arc flow bounds. (This will be shown in Chapter 5.)

(b) Suppose that a concentrator j can operate in an overload condition with a number of connected terminals greater than b_j, up to a number $\overline{b}_j > b_j$. In this case, however, the cost per terminal connected becomes $\overline{a}_{ij} > a_{ij}$. Repeat part (a).

(c) Suppose that when no terminals are connected to concentrator j there is a given cost savings $c_j > 0$. Can you still formulate the problem as a minimum cost flow problem?

1.26

Consider a round-robin chess tournament involving n players that play each other once. A win scores 1 for the winner and 0 for the loser, while a draw scores 1/2 for each player. We are given a set of final scores $(s_1, \ldots, s_n)$ for the players, from the range $[0, n-1]$, whose sum is $n(n-1)/2$, and we want to check whether these scores are feasible [for example, in a four-player tournament, a set of final scores of $(3, 3, 0, 0)$ is impossible]. Show that this is equivalent to checking feasibility of some transportation problem.

1.27 (k-Color Problem)

Consider the k-color problem, which is to assign one out of k colors to each node of a graph so that for every arc (i, j), nodes i and j have different colors.

(a) Suppose we want to choose the colors of countries in a world map so that no two adjacent countries have the same color. Show that if the number of available colors is k, the problem can be formulated as a k-color problem.

(b) Show that the k-color problem has a solution if and only if the number of nodes can be partitioned in k or less disjoint subsets such that there is no arc connecting a pair of nodes from the same subset.

(c) Show that when the graph is a tree, the 2-color problem has a solution. *Hint*: First color some node i and then color the remaining nodes based on their "distance" from i.

(d) Show that if each node has at most $k - 1$ neighbors, the k-color problem has a solution.

1.28 (k-Coloring and Parallel Computation)

Consider the n-dimensional vector $x = (x_1, \ldots, x_n)$ and an iteration of the form

$$x_j := f_j(x), \qquad j = 1, \ldots, n,$$

where $f = (f_1, \ldots, f_n)$ is a given function. The *dependency graph* of f has nodes $1, \ldots, n$ and an arc set such that (i, j) is an arc if the function f_j exhibits a dependence on the component x_i. Consider an ordering $j_1, \ldots, j_n$ of the indices $1, \ldots, n$, and a partition of $\{j_1, \ldots, j_n\}$ into disjoint subsets $J_1, \ldots, J_M$ such that:

(1) For all k, if $j_k \in J_m$, then $j_{k+1} \in J_m \cup \cdots \cup J_M$.

(2) If $j_p, j_q \in J_m$ and $p < q$, then f_{j_q} does not depend on x_{j_p}.

Show that such an ordering and partition exist if and only if the nodes of the dependency graph can be colored with M colors so that there exists no forward cycle with all the nodes on the cycle having the same color. *Note*: This is challenging (see Bertsekas and Tsitsiklis [1989], Section 1.2.4, for discussion and analysis). An ordering and partition of this type can be used to execute Gauss-Seidel iterations in M parallel steps.

1.29 (Replacing Arc Costs with Reduced Costs)

Consider the minimum cost flow problem and let p_j be a scalar price for each node j. Show that if the arc cost coefficients a_{ij} are replaced by $a_{ij} + p_j - p_i$, we obtain a problem that is equivalent to the original (except for a scalar shift in the cost function value).

1.30

Consider the assignment problem.

(a) Show that every k-person exchange can be accomplished with a sequence of $k - 1$ successive two-person exchanges.

(b) In light of the result of part (a), how do you explain that a nonoptimal assignment may not be improvable by any two-person exchange?

1.31 (Dual Cost Improvement Directions)

Consider the assignment problem. Let p_j denote the price of object j, let T be a subset of objects, and let

$$S = \Big\{ i \mid \text{the maximum of } a_{ij} - p_j \text{ over } j \in A(i) \\ \text{is attained by some element of } T \Big\}.$$

Assume that:

(1) For each $i \in S$, the maximum of $a_{ij} - p_j$ over $j \in A(i)$ is attained only by elements of T.

(2) S has more elements than T.

Show that the direction $d = (d_1, \ldots, d_n)$, where $d_j = 1$ if $j \in T$ and $d_j = 0$ if $j \notin T$, is a direction of dual cost improvement. *Note*: Directions of this type are used by the most common dual cost improvement algorithms for the assignment problem.

1.32

Use ϵ-CS to verify that the assignment of Fig. 1.18 is optimal and obtain a bound on how far from optimal the given price vector is. State the dual problem and verify the correctness of the bound by comparing the dual value of the price vector with the optimal dual value.

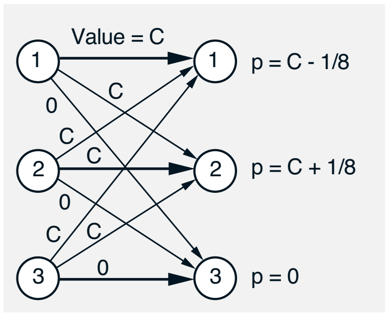


Figure 1.18: Graph of an assignment problem. Objects 1 and 2 have value C for all persons. Object 3 has value 0 for all persons. Object prices are as shown. The thick lines indicate the given assignment.

1.33 (Generic Negative Cycle Algorithm)

Consider the following minimum cost flow problem

$$
\begin{aligned}
&\text{minimize} \quad \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} \\
&\text{subject to} \quad \sum_{\{j\mid(i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j\mid(j,i)\in\mathcal{A}\}} x_{ji} = s_i, \qquad \forall\, i \in \mathcal{N}, \\
&\qquad\qquad 0 \le x_{ij} \le c_{ij}, \qquad \forall\,(i,j)\in\mathcal{A},
\end{aligned}
$$

and assume that the problem has at least one feasible solution. Consider first the circulation case where $s_i = 0$ for all $i \in \mathcal{N}$. Construct a sequence of flow vectors $x^0, x^1, \ldots$ as follows: Start with $x^0 = 0$. Given x^k, stop if x^k is optimal, and otherwise find a simple cycle C^k that is unblocked with respect to x^k and has negative cost (cf. Prop. 1.2). Increase (decrease) the flow of the forward (backward, respectively) arcs of C^k by the maximum possible increment.

(a) Show that the cost of x^{k+1} is smaller than the cost of x^k by an amount that is proportional to the cost of the cycle C^k and to the increment of the corresponding flow change.

(b) Assume that the flow increment at each iteration is greater or equal to some scalar $\delta > 0$. Show that the algorithm must terminate after a finite number of iterations with an optimal flow vector. *Note*: The assumption of existence of such a δ is essential (see Exercise 3.7 in Chapter 3).

(c) Extend parts (a) and (b) to the general case where we may have $s_i \neq 0$ for some i, by converting the problem to the circulation format (a method for doing this is given in Section 4.1.3).

1.34 (Integer Optimal Solutions of Min-Cost Flow Problems)

Consider the minimum cost flow problem of Exercise 1.33, where the upper bounds c_{ij} are given positive integers and the supplies s_i are given integers. Assume that the problem has at least one feasible solution. Show that there exists an optimal flow vector that is integer. *Hint*: Show that the flow vectors generated by the negative cycle algorithm of Exercise 1.33 are integer.

1.35 (The Original Hamiltonian Cycle)

The origins of the traveling salesman problem can be traced (among others) to the work of the Irish mathematician Sir William Hamilton. In 1856, he developed a system of commutative algebra, which inspired a puzzle marketed as the "Icosian Game." The puzzle is to find a cycle that passes exactly once through each of the 20 nodes of the graph shown in Fig. 1.19, which represents a regular dodecahedron. Find a Hamiltonian cycle on this graph using as first four nodes the ones marked 1-4 (all arcs are considered bidirectional).

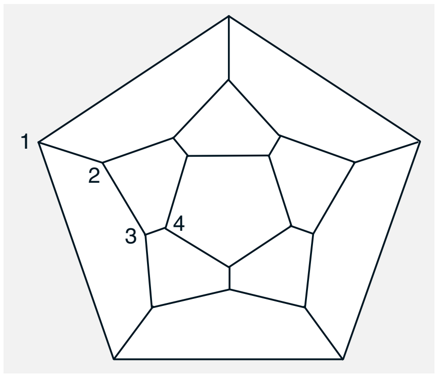


Figure 1.19: Graph for the Icosian Game (cf. Exercise 1.35). The arcs and nodes correspond to the edges and vertices of the regular dodecahedron, respectively. The name "icosian" comes from the Greek word "icosi," which means twenty. Adjacent nodes of the dodecahedron correspond to adjacent faces of the regular icosahedron.

1.36 (Hamiltonian Cycle on the Hypercube)

The hypercube of dimension n is a graph with 2^n nodes, each corresponding to an n-bit string where each bit is either a 0 or a 1. There is a bidirectional arc connecting every pair of nodes whose n-bit strings differ by a single bit. Show that for every $n \geq 2$, the hypercube contains a Hamiltonian cycle. *Hint*: Use induction.

1.37 (Hardy's Theorem)

Let $\{a_1, \ldots, a_n\}$ and $\{b_1, \ldots, b_n\}$ be monotonically nondecreasing sequences of numbers. Consider the problem of associating with each $i = 1, \ldots, n$ a distinct index j_i in a way that maximizes $\sum_{i=1}^{n} a_i b_{j_i}$. Formulate this as an assignment problem and show that it is optimal to select $j_i = i$ for all i. *Hint*: Use the complementary slackness conditions with prices defined by $p_1 = 0$ and $p_k = p_{k-1} + a_k(b_k - b_{k-1})$ for $k = 2, \ldots, n$.

References

Aarts, E., and Lenstra, J. K., 1997. Local Search in Combinatorial Optimization, Wiley, N. Y.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B., 1989. "Network Flows," in Handbooks in Operations Research and Management Science, Vol. 1, Optimization, Nemhauser, G. L., et.al (eds.), North-Holland, Amsterdam, pp. 211-369; available at http://www.dtic.mil/dtic/tr/fulltext/u2/a594171.pdf.

Ahuja, R. K., Mehlhorn, K., Orlin, J. B., and Tarjan, R. E., 1990. "Faster Algorithms for the Shortest Path Problem," J. ACM, Vol. 37, 1990, pp. 213-223.

Ahuja, R. K., and Orlin, J. B., 1987. Private Communication.

Ahuja, R. K., and Orlin, J. B., 1989. "A Fast and Simple Algorithm for the Maximum Flow Problem," Operations Research, Vol. 37, pp. 748-759.

Amini, M. M., 1994. "Vectorization of an Auction Algorithm for Linear Cost Assignment Problem," Comput. Ind. Eng., Vol. 26, pp. 141-149.

Arezki, Y., and Van Vliet, D., 1990. "A Full Analytical Implementation of the PARTAN/Frank-Wolfe Algorithm for Equilibrium Assignment," Transportation Science, Vol. 24, pp. 58-62.

Assad, A. A., and Golden, B. L., 1995. "Arc Routing Methods and Applications," Handbooks in OR and MS, Ball, M. O., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L., (eds.), Vol. 8, North-Holland, Amsterdam, pp. 375-483.

Atkinson, D. S., and Vaidya, P. M., 1995. "A Cutting Plane Algorithm for Convex Programming that Uses Analytic Centers," Math. Programming, Vol. 69, pp. 1-44.

Auchmuty, G., 1989. "Variational Principles for Variational Inequalities," Numer. Functional Analysis and Optimization, Vol. 10, pp. 863-874.

Auslender, A., 1976. Optimization: Methodes Numeriques, Mason, Paris.

Balas, E., Miller, D., Pekny, J., and Toth, P., 1991. "A Parallel Shortest Path Algorithm for the Assignment Problem," J. ACM, Vol. 38, pp. 985-1004.

Balas, E., and Toth, P., 1985. "Branch and Bound Methods," in The Traveling Salesman Problem, Lawler, E., Lenstra, J. K., Rinnoy Kan, A. H. G., and Shmoys, D. B. (eds.), Wiley, N. Y., pp. 361-401.

Balinski, M. L., 1985. "Signature Methods for the Assignment Problem," Operations Research, Vol. 33, pp. 527-537.

Balinski, M. L., 1986. "A Competitive (Dual) Simplex Method for the Assignment Problem," Math. Programming, Vol. 34, pp. 125-141.

Ball, M. O., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L., 1995a. Network Models, Handbooks in OR and MS, Vol. 7, North-Holland, Amsterdam.

Ball, M. O., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L., 1995b. Network Routing, Handbooks in OR and MS, Vol. 8, North-Holland, Amsterdam.

Bar-Shalom, Y., and Fortman, T. E., 1988. Tracking and Data Association, Academic Press, N. Y.

Barnhart, C., Hane, C. H., and Vance, P. H., 1997. "Integer Multicommodity Flow Problems," in Network Optimization, Pardalos, P. M., Hearn, D. W., and Hager, W. W. (eds.), Springer-Verlag, N. Y., pp. 17-31.

Barr, R., Glover, F., and Klingman, D., 1977. "The Alternating Basis Algorithm for Assignment Problems," Math. Programming, Vol. 13, pp. 1-13.

Barr, R., Glover, F., and Klingman, D., 1978. "Generalized Alternating Path Algorithm for Transportation Problems," European J. of Operations Research, Vol. 2, pp. 137-144.

Barr, R., Glover, F., and Klingman, D., 1979. "Enhancement of Spanning Tree Labeling Procedures for Network Optimization," INFOR, Vol. 17, pp. 16-34.

Barr, R., and Hickman, B. L., 1994. "Parallel Simplex for Large Pure Network Problems - Computational Testing and Sources of Speedup," Operations Research, Vol. 42, pp. 65-80.

Bazaraa, M. S., Jarvis, J. J., and Sherali, H. D., 1990. Linear Programming and Network Flows (2nd Ed.), Wiley, N. Y.

Bazaraa, M. S., Sherali, H. D., and Shetty, C. M., 1993. Nonlinear Programming Theory and Algorithms (2nd Ed.), Wiley, N. Y.

Bell, G. J., and Lamar, B. W., 1997. "Solution Methods for Nonconvex Network Flow Problems," in Network Optimization, Pardalos, P. M., Hearn,

D. W., and Hager, W. W. (eds.), Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, N. Y., pp. 32-50.

Bellman, R., 1957. Dynamic Programming, Princeton Univ. Press, Princeton, N. J.

Benders, J. F., 1962. "Partitioning Procedures for Solving Mixed Variables Programming Problems," Numer. Math., Vol. 4, pp. 238-252.

Beraldi, P., and Guerriero, F., 1997. "A Parallel Asynchronous Implementation of the Epsilon-Relaxation Method for the Linear Minimum Cost Flow Problem," Parallel Computing, Vol. 23, pp. 1021-1044.

Beraldi, P., Guerriero, F., and Musmanno, R., 1996. "Parallel Algorithms for Solving the Convex Minimum Cost Flow Problem," Tech. Report PARCOLAB No. 8/96, Dept. of Electronics, Informatics, and Systems, Univ. of Calabria.

Beraldi, P., Guerriero, F., and Musmanno, R., 1997. "Efficient Parallel Algorithms for the Minimum Cost Flow Problem," J. of Optimization Theory and Applications, Vol. 95, pp. 501-530.

Berge, C., 1962. The Theory of Graphs and its Applications, Wiley, N. Y.

Berge, C., and Ghouila-Houri, A., 1962. Programming, Games, and Transportation Networks, Wiley, N. Y.

Bertsekas, D. P., 1975a. "Nondifferentiable Optimization via Approximation," Math. Programming Studies, Vol. 3, North-Holland, Amsterdam, pp. 1-25.

Bertsekas, D. P., 1975b. "Necessary and Sufficient Conditions for a Penalty Method to be Exact," Math. Programming, Vol. 9, pp. 87-99.

Bertsekas, D. P., 1979a. "A Distributed Algorithm for the Assignment Problem," Lab. for Information and Decision Systems Working Paper, M.I.T., Cambridge, MA.

Bertsekas, D. P., 1979b. "Algorithms for Nonlinear Multicommodity Network Flow Problems," in International Symposium on Systems Optimization and Analysis, Bensoussan, A., and Lions, J. L. (eds.), Springer-Verlag, N. Y., pp. 210-224.

Bertsekas, D. P., 1980. "A Class of Optimal Routing Algorithms for Communication Networks," Proc. of the Fifth International Conference on Computer Communication, Atlanta, Ga., pp. 71-76.

Bertsekas, D. P., 1981. "A New Algorithm for the Assignment Problem," Math. Programming, Vol. 21, pp. 152-171.

Bertsekas, D. P., 1982. Constrained Optimization and Lagrange Multiplier Methods, Academic Press, N. Y. (republished in 1996 by Athena Scientific, Belmont, MA).

Bertsekas, D. P., 1985. "A Unified Framework for Minimum Cost Network Flow Problems," Math. Programming, Vol. 32, pp. 125-145.

Bertsekas, D. P., 1986a. "Distributed Asynchronous Relaxation Methods for Linear Network Flow Problems," Lab. for Information and Decision Systems Report P-1606, M.I.T., Cambridge, MA.

Bertsekas, D. P., 1986b. "Distributed Relaxation Methods for Linear Network Flow Problems," Proceedings of 25th IEEE Conference on Decision and Control, Athens, Greece, pp. 2101-2106.

Bertsekas, D. P., 1988. "The Auction Algorithm: A Distributed Relaxation Method for the Assignment Problem," Annals of Operations Research, Vol. 14, pp. 105-123.

Bertsekas, D. P., 1990. "The Auction Algorithm for Assignment and Other Network Flow Problems: A Tutorial," Interfaces, Vol. 20, pp. 133-149.

Bertsekas, D. P., 1991a. Linear Network Optimization: Algorithms and Codes, MIT Press, Cambridge, MA.

Bertsekas, D. P., 1991b. "An Auction Algorithm for Shortest Paths," SIAM J. on Optimization, Vol. 1, pp. 425-447.

Bertsekas, D. P., 1992a. "Auction Algorithms for Network Flow Problems: A Tutorial Introduction," Computational Optimization and Applications, Vol. 1, pp. 7-66.

Bertsekas, D. P., 1992b. "Modified Auction Algorithms for Shortest Paths," Lab. for Information and Decision Systems Report P-2150, M.I.T., Cambridge, MA.

Bertsekas, D. P., 1992c. "An Auction Sequential Shortest Path Algorithm for the Minimum Cost Network Flow Problem," Lab. for Information and Decision Systems Report P-2146, M.I.T.

Bertsekas, D. P., 1993a. "A Simple and Fast Label Correcting Algorithm for Shortest Paths," Networks, Vol. 23, pp. 703-709.

Bertsekas, D. P., 1993b. "Mathematical Equivalence of the Auction Algorithm for Assignment and the ϵ-Relaxation (Preflow-Push) Method for Min Cost Flow," in Large Scale Optimization: State of the Art, Hager, W. W., Hearn, D. W., and Pardalos, P. M. (eds.), Kluwer, Boston, pp. 27-46.

Bertsekas, D. P., 1995a. Dynamic Programming and Optimal Control, Vols. I and II, Athena Scientific, Belmont, MA.

Bertsekas, D. P., 1995b. Nonlinear Programming, Athena Scientific, Belmont, MA.

Bertsekas, D. P., 1995c. "An Auction Algorithm for the Max-Flow Problem," J. of Optimization Theory and Applications, Vol. 87, pp. 69-101.

Bertsekas, D. P., 1996. "Thevenin Decomposition and Network Optimization," J. of Optimization Theory and Applications, Vol. 89, pp. 1-15.

Bertsekas, D. P., and Castañon, D. A., 1989. "The Auction Algorithm for Transportation Problems," Annals of Operations Research, Vol. 20, pp. 67-96.

Bertsekas, D. P., and Castañon, D. A., 1991. "Parallel Synchronous and Asynchronous Implementations of the Auction Algorithm," Parallel Computing, Vol. 17, pp. 707-732.

Bertsekas, D. P., and Castañon, D. A., 1992. "A Forward/Reverse Auction Algorithm for Asymmetric Assignment Problems," Computational Optimization and Applications, Vol. 1, pp. 277-297.

Bertsekas, D. P., and Castañon, D. A., 1993a. "Asynchronous Hungarian Methods for the Assignment Problem," ORSA J. on Computing, Vol. 5, pp. 261-274.

Bertsekas, D. P., and Castañon, D. A., 1993b. "Parallel Primal-Dual Methods for the Minimum Cost Flow Problem," Computational Optimization and Applications, Vol. 2, pp. 317-336.

Bertsekas, D. P., and Castañon, D. A., 1993c. "A Generic Auction Algorithm for the Minimum Cost Network Flow Problem," Computational Optimization and Applications, Vol. 2, pp. 229-260.

Bertsekas, D. P., and Castañon, D. A., 1998. "Solving Stochastic Scheduling Problems Using Rollout Algorithms," Lab. for Information and Decision Systems Report P-12413, M.I.T., Cambridge, MA.

Bertsekas, D. P., Castañon, D. A., Eckstein, J., and Zenios, S., 1995. "Parallel Computing in Network Optimization," Handbooks in OR and MS, Ball, M. O., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L. (eds.), Vol. 7, North-Holland, Amsterdam, pp. 331-399.

Bertsekas, D. P., Castañon, D. A., and Tsaknakis, H., 1993. "Reverse Auction and the Solution of Inequality Constrained Assignment Problems," SIAM J. on Optimization, Vol. 3, pp. 268-299.

Bertsekas, D. P., and El Baz, D., 1987. "Distributed Asynchronous Relaxation Methods for Convex Network Flow Problems," SIAM J. on Control and Optimization, Vol. 25, pp. 74-85.

Bertsekas, D. P., and Eckstein, J., 1987. "Distributed Asynchronous Relaxation Methods for Linear Network Flow Problems," Proc. of IFAC '87, Munich, Germany.

Bertsekas, D. P., and Eckstein, J., 1988. "Dual Coordinate Step Methods for Linear Network Flow Problems," Math. Programming, Series B, Vol. 42, pp. 203-243.

Bertsekas, D. P., and Gafni, E. M., 1982. "Projection Methods for Variational Inequalities with Application to the Traffic Assignment Problem," Math. Progr. Studies, Vol. 17, North-Holland, Amsterdam, pp. 139-159.

Bertsekas, D. P., and Gafni, E. M., 1983. "Projected Newton Methods and Optimization of Multicommodity Flows," IEEE Trans. on Auto. Control, Vol. 28, pp. 1090-1096.

Bertsekas, D. P., Gafni, E. M., and Gallager, R. G., 1984. "Second Derivative Algorithms for Minimum Delay Distributed Routing in Networks," IEEE Trans. on Communications, Vol. 32, pp. 911-919.

Bertsekas, D. P., and Gallager, R. G., 1992. Data Networks, (2nd Ed.), Prentice-Hall, Englewood Cliffs, N. J.

Bertsekas, D. P., Guerriero, F., and Musmanno, R., 1996. "Parallel Asynchronous Label Correcting Methods for Shortest Paths," J. of Optimization Theory and Applications, Vol. 88, pp. 297-320.

Bertsekas, D. P., Hosein, P., and Tseng, P., 1987. "Relaxation Methods for Network Flow Problems with Convex Arc Costs," SIAM J. on Control and Optimization, Vol. 25, pp. 1219-1243.

Bertsekas, D. P, and Mitter, S. K., 1971. "Steepest Descent for Optimization Problems with Nondifferentiable Cost Functionals," Proc. 5th Annual Princeton Confer. Inform. Sci. Systems, Princeton, N. J., pp. 347-351.

Bertsekas, D. P., and Mitter, S. K., 1973. "Descent Numerical Methods for Optimization Problems with Nondifferentiable Cost Functions," SIAM J. on Control, Vol. 11, pp. 637-652.

Bertsekas, D. P., Pallottino, S., and Scutellà, M. G., 1995. "Polynomial Auction Algorithms for Shortest Paths," Computational Optimization and Applications, Vol. 4, pp. 99-125.

Bertsekas, D. P., Polymenakos, L. C., and Tseng, P., 1997a. "An ϵ-Relaxation Method for Separable Convex Cost Network Flow Problems," SIAM J. on Optimization, Vol. 7, pp. 853-870.

Bertsekas, D. P., Polymenakos, L. C., and Tseng, P., 1997b. "Epsilon-Relaxation and Auction Methods for Separable Convex Cost Network Flow Problems," in Network Optimization, Pardalos, P. M., Hearn, D. W., and Hager, W. W. (eds.), Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, N. Y., pp. 103-126.

Bertsekas, D. P., and Tseng, P., 1988a. "Relaxation Methods for Minimum Cost Ordinary and Generalized Network Flow Problems," Operations Research, Vol. 36, pp. 93-114.

Bertsekas, D. P., and Tseng, P., 1988b. "RELAX: A Computer Code for Minimum Cost Network Flow Problems," Annals of Operations Research, Vol. 13, pp. 127-190.

Bertsekas, D. P., and Tseng, P., 1990. “RELAXT-III: A New and Improved Version of the RELAX Code,” Lab. for Information and Decision Systems Report P-1990, M.I.T., Cambridge, MA.

Bertsekas, D. P., and Tseng, P., 1994. “RELAX-IV: A Faster Version of the RELAX Code for Solving Minimum Cost Flow Problems,” Laboratory for Information and Decision Systems Report P-2276, M.I.T., Cambridge, MA.

Bertsekas, D. P., and Tsitsiklis, J. N., 1989. Parallel and Distributed Computation: Numerical Methods, Prentice-Hall, Englewood Cliffs, N. J. (republished in 1997 by Athena Scientific, Belmont, MA).

Bertsekas, D. P., and Tsitsiklis, J. N., 1996. Neuro-Dynamic Programming, Athena Scientific, Belmont, MA.

Bertsekas, D. P., Tsitsiklis, J. N., and Wu, C., 1997. “Rollout Algorithms for Combinatorial Optimization,” Heuristics, Vol. 3, pp. 245-262.

Bertsimas, D., and Tsitsiklis, J. N., 1993. “Simulated Annealing,” Stat. Sci., Vol. 8, pp. 10-15.

Bertsimas, D., and Tsitsiklis, J. N., 1997. Introduction to Linear Optimization, Athena Scientific, Belmont, MA.

Birkhoff, G., and Diaz, J. B., 1956. “Nonlinear Network Problems,” Quart. Appl. Math., Vol. 13, pp. 431-444.

Bland, R. G., and Jensen, D. L., 1985. “On the Computational Behavior of a Polynomial-Time Network Flow Algorithm,” Tech. Report 661, School of Operations Research and Industrial Engineering, Cornell University.

Blackman, S. S., 1986. Multi-Target Tracking with Radar Applications, Artech House, Dehdam, MA.

Bogart, K. P., 1990. Introductory Combinatorics, Harcourt Brace Jovanovich, Inc., New York, N. Y.

Bradley, G. H., Brown, G. G., and Graves, G. W., 1977. “Design and Implementation of Large-Scale Primal Transshipment Problems,” Management Science, Vol. 24, pp. 1-38.

Brannlund, U., 1993. On Relaxation Methods for Nonsmooth Convex Optimization, Doctoral Thesis, Royal Institute of Technology, Stockhorm, Sweden.

Brown, G. G., and McBride, R. D., 1984. “Solving Generalized Networks,” Management Science, Vol. 30, pp. 1497-1523.

Burkard, R. E., 1990. “Special Cases of Traveling Salesman Problems and Heuristics,” Acta Math. Appl. Sin., Vol. 6, pp. 273-288.

Busacker, R. G., and Gowen, P. J., 1961. “A Procedure for Determining a

Family of Minimal-Cost Network Flow Patterns," O.R.O. Technical Report No. 15, Operational Research Office, John Hopkins University, Baltimore, MD.

Busacker, R. G., and Saaty, T. L., 1965. Finite Graphs and Networks: An Introduction with Applications, McGraw-Hill, N. Y.

Cameron, P. J., 1994. Combinatorics: Topics, Techniques, Algorithms, Cambridge Univ. Press, Cambridge, England.

Cantor, D. G., and Gerla, M., 1974. "Optimal Routing in a Packet Switched Computer Network," IEEE Trans. on Computers, Vol. 23, pp. 1062-1069.

Carpaneto, G., Martello, S., and Toth, P., 1988. "Algorithms and Codes for the Assignment Problem," Annals of Operations Research, Vol. 13, pp. 193-223.

Carraresi, P., and Sodini, C., 1986. "An Efficient Algorithm for the Bipartite Matching Problem," Eur. J. Operations Research, Vol. 23, pp. 86-93.

Castañon, D. A., 1990. "Efficient Algorithms for Finding the K Best Paths Through a Trellis," IEEE Trans. on Aerospace and Electronic Systems, Vol. 26, pp. 405-410.

Castañon, D. A., 1993. "Reverse Auction Algorithms for Assignment Problems," in Algorithms for Network Flows and Matching, Johnson, D. S., and McGeoch, C. C. (eds.), American Math. Soc., Providence, RI, pp. 407-429.

Censor, Y., and Zenios, S. A., 1992. "The Proximal Minimization Algorithm with D-Functions," J. Opt. Theory and Appl., Vol. 73, pp. 451-464.

Censor, Y., and Zenios, S. A., 1997. Parallel Optimization: Theory, Algorithms, and Applications, Oxford University Press, N. Y.

Cerny, V., 1985. "A Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm," J. Opt. Theory and Applications, Vol. 45, pp. 41-51.

Cerulli, R., De Leone, R., and Piacente, G., 1994. "A Modified Auction Algorithm for the Shortest Path Problem," Optimization Methods and Software, Vol. 4, pp. 209-224.

Cerulli, R., Festa, P., and Raiconi, G., 1997a. "Graph Collapsing in Shortest Path Auction Algorithms," Univ. of Salerno Tech. Report n. 6/97.

Cerulli, R., Festa, P., and Raiconi, G., 1997b. "An Efficient Auction Algorithm for the Shortest Path Problem Using Virtual Source Concept," Univ. of Salerno Tech. Report n. 6/97.

Chajakis, E. D., and Zenios, S. A., 1991. "Synchronous and Asynchronous Implementations of Relaxation Algorithms for Nonlinear Network Optimization," Parallel Computing, Vol. 17, pp. 873-894.

Chen, G., and Teboulle, M., 1993. "Convergence Analysis of a Proximal-Like Minimization Algorithm Using Bregman Functions," SIAM J. on Optimization, Vol. 3, pp. 538-543.

Chen, Z. L., and Powell, W. B., 1997. "A Note on Bertsekas' Small-Label-First Strategy," Networks, Vol. 29, pp. 111-116.

Cheney, E. W., and Goldstein, A. A., 1959. "Newton's Method for Convex Programming and Tchebycheff Approximation," Numer. Math., Vol. I, pp. 253-268.

Cheriyan, J., and Maheshwari, S. N., 1989. "Analysis of Preflow Push Algorithms for Maximum Network Flow," SIAM J. Computing, Vol. 18, pp. 1057-1086.

Cherkasky, R. V., 1977. "Algorithm for Construction of Maximum Flow in Networks with Complexity of $O(V^2\sqrt{E})$ Operations," Mathematical Methods of Solution of Economical Problems, Vol. 7, pp. 112-125.

Christofides, N., 1975. Graph Theory: An Algorithmic Approach, Academic Press, N. Y.

Chvatal, V., 1983. Linear Programming, W. H. Freeman and Co., N. Y.

Connors, D. P., and Kumar, P. R., 1989. "Simulated Annealing Type Markov Chains and their Order Balance Equations," SIAM J. on Control and Optimization, Vol. 27, pp. 1440-1461.

Cook, W., Cunningham, W., Pulleyblank, W., and Schrijver, A., 1998. Combinatorial Optimization, Wiley, N. Y.

Cornuejols, G., Fonlupt, J., and Naddef, D., 1985. "The Traveling Salesman Problem on a Graph and Some Related Polyhedra," Math. Programming, Vol. 33, pp. 1-27.

Cottle, R. W., and Pang, J. S., 1982. "On the Convergence of a Block Successive Over-Relaxation Method for a Class of Linear Complementarity Problems," Math. Progr. Studies, Vol. 17, pp. 126-138.

Croes, G. A., 1958. "A Method for Solving Traveling Salesman Problems," Operations Research, Vol. 6, pp. 791-812.

Cunningham, W. H., 1976. "A Network Simplex Method," Math. Programming, Vol. 4, pp. 105-116.

Cunningham, W. H., 1979. "Theoretical Properties of the Network Simplex Method," Math. of Operations Research, Vol. 11, pp. 196-208.

Dafermos, S., 1980. "Traffic Equilibrium and Variational Inequalities," Transportation Science, Vol. 14, pp. 42-54.

Dafermos, S., 1982. "Relaxation Algorithms for the General Asymmetric Traffic Equilibrium Problem," Transportation Science, Vol. 16, pp. 231-240.

Dafermos, S., and Sparrow, F. T., 1969. "The Traffic Assignment Problem for a General Network," J. Res. Nat. Bureau of Standards, Vol. 73B, pp. 91-118.

Dantzig, G. B., 1951. "Application of the Simplex Method to a Transportation Problem," in Activity Analysis of Production and Allocation, T. C. Koopmans (ed.), Wiley, N. Y., pp. 359-373.

Dantzig, G. B., 1960. "On the Shortest Route Problem Through a Network," Management Science, Vol. 6, pp. 187-190.

Dantzig, G. B., 1963. Linear Programming and Extensions, Princeton Univ. Press, Princeton, N. J.

Dantzig, G. B., 1967. "All Shortest Routes in a Graph," in Theory of Graphs, P. Rosenthier (ed.), Gordan and Breach, N. Y., pp. 92-92.

Dantzig, G. B., and Fulkerson, D. R., 1956. "On the Max-Flow Min-Cut Theorem of Networks," in Linear Inequalities and Related Systems, Kuhn, H. W., and Tucker, A. W. (eds.), Annals of Mathematics Study 38, Princeton Univ. Press, pp. 215-221.

Dantzig, G. B., and Wolfe, P., 1960. "Decomposition Principle for Linear Programs," Operations Research, Vol. 8, pp. 101-111.

Dantzig, G. B., Fulkerson, D. R., and Johnson, S. M., 1954. "Solution of a Large-Scale Traveling-Salesman Problem," Operations Research, Vol. 2, pp. 393-410.

De Leone, R., Meyer, R. R., and Zakarian, A., 1995. "An ϵ-Relaxation Algorithm for Convex Network Flow Problems," Computer Sciences Department Technical Report, University of Wisconsin, Madison, WI.

Dembo, R. S., 1987. "A Primal Truncated Newton Algorithm for Large-Scale Unconstrained Optimization," Math. Programming Studies, Vol. 31, pp. 43-72.

Dembo, R. S., and Klincewicz, J. G., 1981. "A Scaled Reduced Gradient Algorithm for Network Flow Problems with Convex Separable Costs," Math. Programming Studies, Vol. 15, pp. 125-147.

Dembo, R. S., and Tulowitzki, U., 1988. "Computing Equilibria on Large Multicommodity Networks: An Application of Truncated Quadratic Programming Algorithms," Networks, Vol. 18, pp. 273-284.

Denardo, E. V., and Fox, B. L., 1979. "Shortest-Route Methods: 1. Reaching, Pruning and Buckets," Operations Research, Vol. 27, pp. 161-186.

Dennis, J. B., 1959. Mathematical Programming and Electical Circuits, Technology Press of M.I.T., Cambridge, MA.

Deo, N., and Kumar, N., 1997. "Computation of Constrained Spanning Trees: A Unified Approach," in Network Optimization, Pardalos, P. M., Hearn, D. W., and Hager, W. W. (eds.), Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, N. Y., pp. 194-220.

Deo, N., and Pang, C., 1984. "Shortest Path Algorithms: Taxonomy and Annotation," Networks, Vol. 14, pp. 275-323.

Derigs, U., 1985. "The Shortest Augmenting Path Method for Solving Assignment Problems – Motivation and Computational Experience," Annals of Operations Research, Vol. 4, pp. 57-102.

Derigs, U., and Meier, W., 1989. "Implementing Goldberg's Max-Flow Algorithm – A Computational Investigation," Zeitschrif fur Operations Research, Vol. 33, pp. 383-403.

Desrosiers, J., Dumas, Y., Solomon, M. M., and Soumis, F., 1995. "Time Constrained Routing and Scheduling," Handbooks in OR and MS, Ball, M. O., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L. (eds.), Vol. 8, North-Holland, Amsterdam, pp. 35-139.

Dial, R. B., 1969. "Algorithm 360: Shortest Path Forest with Topological Ordering," Comm. ACM, Vol. 12, pp. 632-633.

Dial, R., Glover, F., Karney, D., and Klingman, D., 1979. "A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shortest Path Trees," Networks, Vol. 9, pp. 215-248.

Dijkstra, E., 1959. "A Note on Two Problems in Connexion with Graphs," Numer. Math., Vol. 1, pp. 269-271.

Dinic, E. A., 1970. "Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation," Soviet Math. Doklady, Vol. 11, pp. 1277-1280.

Dreyfus, S. E., 1969. "An Appraisal of Some Shortest-Path Algorithms," Operations Research, Vol. 17, pp. 395-412.

Duffin, R. J., 1947. "Nonlinear Networks. IIa," Bull. Amer. Math. Soc., Vol. 53, pp. 963-971.

Eastman, W. L., 1958. Linear Programming with Pattern Constraints, Ph.D. Thesis, Harvard University, Cambridge, MA.

Eckstein, J., 1994. "Nonlinear Proximal Point Algorithms Using Bregman Functions, with Applications to Convex Programming," Math. of Operations Research, Vol. 18, pp. 202-226.

Edmonds, J., 1965. "Paths, Trees, and Flowers," Canadian J. of Math., Vol. 17, pp. 449-467.

Edmonds, J., and Johnson, E. L., 1973. "Matching, Euler Tours, and the Chinese Postman," Math. Programming, Vol. 5, pp. 88-124.

Edmonds, J., and Karp, R. M., 1972. "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," J. ACM, Vol. 19, pp. 248-264.

Eiselt, H. A., Gendreau, M., and Laporte, G., 1995a. "Arc Routing Problems, Part 1: The Chinese Postman Problem," Operations Research, Vol. 43, pp. 231-242.

Eiselt, H. A., Gendreau, M., and Laporte, G., 1995b. "Arc Routing Problems, Part 2: The Rural Postman Problem," Operations Research, Vol. 43, pp. 399-414.

Elias, P., Feinstein, A., and Shannon, C. E., 1956. "Note on Maximum Flow Through a Network," IRE Trans. Info. Theory, Vol. IT-2, pp. 117-119.

Egervary, J., 1931. "Matrixok Kombinatoricus Tulajonsagairol," Mat. Es Fiz. Lapok, Vol. 38, pp. 16-28.

El Baz, D., 1989. "A Computational Experience with Distributed Asynchronous Iterative Methods for Convex Network Flow Problems," Proc. of the 28th IEEE Conference on Decision and Control, Tampa, Fl., pp. 590-591.

El Baz, D., 1996. "Asynchronous Gradient Algorithms for a Class of Convex Separable Network Flow Problems," Computational Optimization and Applications, Vol. 5, pp. 187-205.

El Baz, D., Spiteri, P., Miellou, J. C., and Gazen, D., 1996. "Asynchronous Iterative Algorithms with Flexible Communication for Nonlinear Network Flow Problems," J. of Parallel and Distributed Computing, Vol. 38, pp. 1-15.

Elam, J., Glover, F., and Klingman, D., 1979. "A Strongly Convergent Primal Simplex Algorithm for Generalized Networks," Math. of Operations Research, Vol. 4, pp. 39-59.

Elmaghraby, S. E., 1978. Activity Networks: Project Planning and Control by Network Models, Wiley, N. Y.

Elzinga, J., and Moore, T. G., 1975. "A Central Cutting Plane Algorithm for the Convex Programming Problem," Math. Programming, Vol. 8, pp. 134-145.

Engquist, M., 1982. "A Successive Shortest Path Algorithm for the Assignment Problem," INFOR, Vol. 20, pp. 370-384.

Ephremides, A., 1986. "The Routing Problem in Computer Networks," in Communication and Networks, Blake, I. F., and Poor, H. V. (eds.), Springer-Verlag, N. Y., pp. 299-325.

Ephremides, A., and Verdu, S., 1989. "Control and Optimization Methods in Communication Network Problems," IEEE Trans. on Automatic Control, Vol. 34, pp. 930-942.

Esau, L. R., and Williams, K. C., 1966. "On Teleprocessing System Design. A Method for Approximating the Optimal Network," IBM System J., Vol. 5, pp. 142-147.

Escudero, L. F., 1985. "Performance Evaluation of Independent Superbasic Sets on Nonlinear Replicated Networks," Eur. J. Operations Research, Vol. 23, pp. 343-355.

Everett, H., 1963. "Generalized Lagrange Multiplier Method for Solving Problems of Optimal Allocation of Resources," Operations Research, Vol. 11, pp. 399-417.

Falcone, M., 1987. "A Numerical Approach to the Infinite Horizon Problem of Deterministic Control Theory," Appl. Math. Opt., Vol. 15, pp. 1-13.

Federgruen, A., and Simchi-Levi, D., 1995. "Analysis of Vehicle and Inventory-Routing Problems," Handbooks in OR and MS, Ball, M. O., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L. (eds.), Vol. 8, North-Holland, Amsterdam, pp. 297-373.

Ferris, M. C., 1991. "Finite Termination of the Proximal Point Algorithm," Math. Programming, Vol. 50, pp. 359-366.

Fisher, M., 1995. "Vehicle Routing," Handbooks in OR and MS, Ball, M. O., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L. (eds.), Vol. 8, North-Holland, Amsterdam, pp. 1-33.

Florian, M., Guélat, J., and Spiess, H., 1987. "An Efficient Implementation of the "PARTAN" Variant of the Linear Approximation Method for the Network Equilibrium Problem," Networks, Vol. 17, pp. 319-339.

Florian, M. S., and Hearn, D., 1995. "Network Equilibrium Models and Algorithms," Handbooks in OR and MS, Ball, M. O., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L. (eds.), Vol. 8, North-Holland, Amsterdam, pp. 485-550.

Florian, M. S., and Nguyen, S., 1974. "A Method for Computing Network Equilibrium with Elastic Demands," Transportation Science, Vol. 8, pp. 321-332.

Florian, M. S., and Nguyen, S., 1976. "An Application and Validation of Equilibrium Trip Assignment Methods," Transportation Science, Vol. 10, pp. 374-390.

Florian, M. S., Nguyen, S., and Pallottino, S., 1981. "A Dual Simplex Algorithm for Finding All Shortest Paths," Networks, Vol. 11, pp. 367-378.

Floudas, C. A., 1995. Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications, Oxford University Press, N. Y.

Floyd, R. W., 1962. "Algorithm 97: Shortest Path," Comm. ACM, Vol. 5, pp. 345.

Ford, L. R., Jr., 1956. "Network Flow Theory," Report P-923, The Rand Corporation, Santa Monica, CA.

Ford, L. R., Jr., and Fulkerson, D. R., 1956a. "Solving the Transportation Problem," Management Science, Vol. 3, pp. 24-32.

Ford, L. R., Jr., and Fulkerson, D. R., 1956b. "Maximal Flow Through a Network," Can. J. of Math., Vol. 8, pp. 339-404.

Ford, L. R., Jr., and Fulkerson, D. R., 1957. "A Primal-Dual Algorithm for the Capacitated Hitchcock Problem," Naval Res. Logist. Quart., Vol. 4, pp. 47-54.

Ford, L. R., Jr., and Fulkerson, D. R., 1962. Flows in Networks, Princeton Univ. Press, Princeton, N. J.

Fox, B. L., 1993. "Integrating and Accelerating Tabu Search, Simulated Annealing, and Genetic Algorithms," Annals of Operations Research, Vol. 41, pp. 47-67.

Fox, B. L., 1995. "Faster Simulated Annealing," SIAM J. Optimization, Vol. 41, pp. 47-67.

Frank, H., and Frisch, I. T., 1970. Communication, Transmission, and Transportation Networks, Addison-Wesley, Reading, MA.

Fratta, L., Gerla, M., and Kleinrock, L., 1973. "The Flow-Deviation Method: An Approach to Store-and-Forward Computer Communication Network Design," Networks, Vol. 3, pp. 97-133.

Fredman, M. L., and Tarjan, R. E., 1984. "Fibonacci Heaps and their Uses in Improved Network Optimization Algorithms," Proc. 25th Annual Symp. on Found. of Comp. Sci., pp. 338-346.

Fukushima, M., 1984a. "A Modified Frank-Wolfe Algorithm for Solving the Traffic Assignment Problem," Transportation Research, Vol. 18B, pp. 169–177.

Fukushima, M., 1984b. "On the Dual Approach to the Traffic Assignment Problem," Transportation Research, Vol. 18B, pp. 235-245.

Fukushima, M., 1992. "Equivalent Differentiable Optimization Problems and Descent Methods for Asymmetric Variational Inequalities," Math. Programming, Vol. 53, pp. 99-110.

Fulkerson, D. R., 1961. "An Out-of-Kilter Method for Minimal Cost Flow Problems," SIAM J. Appl. Math., Vol. 9, pp. 18-27.

Fulkerson, D. R., and Dantzig, G. B., 1955. "Computation of Maximum Flow in Networks," Naval Res. Log. Quart., Vol. 2, pp. 277-283.

Gafni, E. M., 1979. "Convergence of a Routing Algorithm," M.S. Thesis, Dept. of Electrical Engineering, Univ. of Illinois, Urbana, Ill.

Gafni, E. M., and Bertsekas, D. P., 1984. "Two-Metric Projection Methods for Constrained Optimization," SIAM J. on Control and Optimization, Vol. 22, pp. 936-964.

Gale, D., 1957. "A Theorem of Flows in Networks," Pacific J. Math., Vol. 7, pp. 1073-1082.

Gale, D., Kuhn, H. W., and Tucker, A. W., 1951. "Linear Programming and the Theory of Games," in Activity Analysis of Production and Allocation, T. C. Koopmans (ed.), Wiley, N. Y.

Galil, Z., 1980. "$O\big(V^{5/3}E^{2/3}\big)$ Algorithm for the Maximum Flow Problem," Acta Informatica, Vol. 14, pp. 221-242.

Galil, Z., and Naamad, A., 1980. "$O\big(VE\log^2 V\big)$ Algorithm for the Maximum Flow Problem," J. of Comput. Sys. Sci., Vol. 21, pp. 203-217.

Gallager, R. G., 1977. "A Minimum Delay Routing Algorithm Using Distributed Computation," IEEE Trans. on Communications, Vol. 23, pp. 73-85.

Gallo, G. S., and Pallottino, S., 1982. "A New Algorithm to Find the Shortest Paths Between All Pairs of Nodes," Discrete Applied Mathematics, Vol. 4, pp. 23-35.

Gallo, G. S., and Pallottino, S., 1986. "Shortest Path Methods: A Unified Approach," Math. Programming Studies, Vol. 26, pp. 38-64.

Gallo, G. S., and Pallottino, S., 1988. "Shortest Path Algorithms," Annals of Operations Research, Vol. 7, pp. 3-79.

Garey, M. R., and Johnson, D. S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Co., San Francisco, Ca.

Gartner, N. H., 1980a. "Optimal Traffic Assignment with Elastic Demands: A Review. Part I. Analysis Framework," Transportation Science, Vol. 14, pp. 174-191.

Gartner, N. H., 1980b. "Optimal Traffic Assignment with Elastic Demands: A Review. Part II. Algorithmic Approaches," Transportation Science, Vol. 14, pp. 192-208.

Gavish, B., Schweitzer, P., and Shlifer, E., 1977. "The Zero Pivot Phenomenon in Transportation Problems and its Computational Implications," Math. Programming, Vol. 12, pp. 226-240.

Gelfand, S. B., and Mitter, S. K., 1989. "Simulated Annealing with Noisy or Imprecise Measurements," J. Opt. Theory and Applications, Vol. 69, pp. 49-62.

Geoffrion, A. M., 1970. “Elements of Large-Scale Mathematical Programming, I, II,” Management Science, Vol. 16, pp. 652-675, 676-691.

Geoffrion, A. M., 1974. “Lagrangian Relaxation for Integer Programming,” Math. Programming Studies, Vol. 2, pp. 82-114.

Gerards, A. M. H., 1995. “Matching,” Handbooks in OR and MS, Ball, M. O., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L. (eds.), Vol. 7, North-Holland, Amsterdam, pp. 135-224.

Gibby, D., Glover, F., Klingman, D., and Mead, M., 1983. “A Comparison of Pivot Selection Rules for Primal Simplex Based Network Codes,” Operations Research Letters, Vol. 2, pp. 199-202.

Gill, P. E., Murray, W., and Wright, M. H., 1981. Practical Optimization, Academic Press, N. Y.

Gilmore, P. C., Lawler, E. L., and Shmoys, D. B., 1985. “Well-Solved Special Cases,” in The Traveling Salesman Problem, Lawler, E., Lenstra, J. K., Rinnoy Kan, A. H. G., and Shmoys, D. B. (eds.), Wiley, N. Y., pp. 87-143.

Glover, F., 1986. “Future Paths for Integer Programming and Links to Artificial Intelligence,” Computers and Operations Research, Vol. 13, pp. 533-549.

Glover, F., 1989. “Tabu Search: Part I,” ORSA J. on Computing, Vol. 1, pp. 190-206.

Glover, F., 1990. “Tabu Search: Part II,” ORSA J. on Computing, Vol. 2, pp. 4-32.

Glover, F., Glover, R., and Klingman, D., 1986. “The Threshold Shortest Path Algorithm,” Math. Programming Studies, Vol. 26, pp. 12-37.

Glover, F., Glover, R., and Klingman, D., 1986. “Threshold Assignment Algorithm,” Math. Programming Studies, Vol. 26, pp. 12-37.

Glover, F., Karney, D., and Klingman, D., 1974. “Implementation and Computational Comparisons of Primal, Dual, and Primal-Dual Computer Codes for Minimum Cost Network Flow Problem,” Networks, Vol. 4, pp. 191-212.

Glover, F., Karney, D., Klingman, D., and Napier, A., 1974. “A Computation Study on Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems,” Management Science, Vol. 20, pp. 793-819.

Glover, F., Klingman, D., Mote, J., and Whitman, D., 1984. “A Primal Simplex Variant for the Maximum Flow Problem,” Naval Res. Logist. Quart., Vol. 31, pp. 41-61.

Glover, F., Klingman, D., and Phillips, N., 1985. “A New Polynomially

Bounded Shortest Path Algorithm," Operations Research, Vol. 33, pp. 65-73.

Glover, F., Klingman, D., and Phillips, N., 1992. Network Models in Optimization and Their Applications in Practice, Wiley, N. Y.

Glover, F., Klingman, D., Phillips, N., and Schneider, R. F., 1985. "New Polynomial Shortest Path Algorithms and Their Computational Attributes," Management Science, Vol. 31, pp. 1106-1128.

Glover, F., Klingman, D., and Stutz, J., 1973. "Extension of the Augmented Predecessor Index Method to Generalized Netork Problems," Transportation Science, Vol. 7, pp. 377-384.

Glover, F., Klingman, D., and Stutz, J., 1974. "Augmented Threaded Index Method for Network Optimization," INFOR, Vol. 12, pp. 293-298.

Glover, F., and Laguna, M., 1997. Tabu Search, Kluwer, Boston.

Glover, F., Taillard, E., and de Verra, D., 1993. "A User's Guide to Tabu Search," Annals of Operations Research, Vol. 41, pp. 3-28.

Goffin, J. L., 1977. "On Convergence Rates of Subgradient Optimization Methods," Math. Programming, Vol. 13, pp. 329-347.

Goffin, J. L., Haurie, A., and Vial, J. P., 1992. "Decomposition and Non-differentiable Optimization with the Projective Algorithm," Management Science, Vol. 38, pp. 284-302.

Goffin, J. L., and Kiwiel, K. C, 1996. 'Convergence of a Simple Subgradient Level Method," Unpublished Report, to appear in Math. Programming.

Goffin, J. L., Luo, Z.-Q., and Ye, Y., 1993. "On the Complexity of a Column Generation Algorithm for Convex or Quasiconvex Feasibility Problems," in Large Scale Optimization: State of the Art, Hager, W. W., Hearn, D. W., and Pardalos, P. M. (eds.), Kluwer.

Goffin, J. L., Luo, Z.-Q., and Ye, Y., 1996. "Further Complexity Analysis of a Primal-Dual Column Generation Algorithm for Convex or Quasiconvex Feasibility Problems," SIAM J. on Optimization, Vol. 6, pp. 638-652.

Goffin, J. L., and Vial, J. P., 1990. "Cutting Planes and Column Generation Techniques with the Projective Algorithm," J. Opt. Th. and Appl., Vol. 65, pp. 409-429.

Goldberg, A. V., 1987. "Efficient Graph Algorithms for Sequential and Parallel Computers," Tech. Report TR-374, Laboratory for Computer Science, M.I.T., Cambridge, MA.

Goldberg, A. V., 1993. "An Efficient Implementation of a Scaling Minimum-Cost Flow Algorithm," Proc. 3rd Integer Progr. and Combinatorial Optimization Conf., pp. 251-266.

Goldberg, A. V., and Tarjan, R. E., 1986. "A New Approach to the Maximum Flow Problem," Proc. 18th ACM STOC, pp. 136-146.

Goldberg, A. V., and Tarjan, R. E., 1990. "Solving Minimum Cost Flow Problems by Successive Approximation," Math. of Operations Research, Vol. 15, pp. 430-466.

Goldberg, D. E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, Reading, MA.

Goldfarb, D., 1985. "Efficient Dual Simplex Algorithms for the Assignment Problem," Math. Programming, Vol. 33, pp. 187-203.

Goldfarb, D., and Hao, J., 1990. "A Primal Simplex Algorithm that Solves the Maximum Flow Problem in at Most nm Pivots and $O(n^2m)$ Time," Math. Programming, Vol. 47, pp. 353-365.

Goldfarb, D., Hao, J., and Kai, S., 1990a. "Anti-Stalling Pivot Rules for the Network Simplex Algorithm," Networks, Vol. 20, pp. 79-91.

Goldfarb, D., Hao, J., and Kai, S., 1990b. "Efficient Shortest Path Simplex Algorithms," Operations Research, Vol. 38, pp. 624-628.

Goldfarb, D., and Reid, J. K., 1977. "A Practicable Steepest Edge Simplex Algorithm," Math. Programming, Vol. 12, pp. 361-371.

Goldstein, A. A., 1967. Constructive Real Analysis, Harper and Row, N. Y.

Gondran, M., and Minoux, M., 1984. Graphs and Algorithms, Wiley, N. Y.

Gonzalez, R., and Rofman, E., 1985. "On Deterministic Control Problems: An Approximation Procedure for the Optimal Cost, Parts I, II," SIAM J. on Control and Optimization, Vol. 23, pp. 242-285.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G., 1979. "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey," Annals of Discrete Math., Vol. 5, pp. 287-326.

Grötschel, M., Monma, C. L., and Stoer, M., 1995. "Design of Survivable Networks," Handbooks in OR and MS, Ball, M. O., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L. (eds.), Vol. 7, North-Holland, Amsterdam, pp. 617-672.

Grötschel, M., and Padberg, M. W., 1985. "Polyhedral Theory," in The Traveling Salesman Problem, Lawler, E., Lenstra, J. K., Rinnoy Kan, A. H. G., and Shmoys, D. B. (eds.), Wiley, N. Y., pp. 251-305.

Guerriero, F., Lacagnina, V., Musmanno, R., and Pecorella, A., 1996. "Efficient Node Selection Strategies in Label-Correcting Methods for the K Shortest Paths Problem," Technical Report PARCOLAB No. 6/96, Department of Electronics, Informatics and Systems, University of Calabria.

Guler, O., 1992. "New Proximal Point Algorithms for Convex Minimization," SIAM J. on Optimization, Vol. 2, pp. 649-664.

Hajek, B., 1988. "Cooling Schedules for Optimal Annealing," Math. of Operations Research, Vol. 13, pp. 311-329.

Hall, M., Jr., 1956. "An Algorithm for Distinct Representatives," Amer. Math. Monthly, Vol. 51, pp. 716-717.

Hansen, P., 1986. "The Steepest Ascent Mildest Descent Heuristic for Combinatorial Optimization," Presented at the Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy.

Hearn, D. W., and Lawphongpanich, S., 1990. "A Dual Ascent Algorithm for Traffic Assignment Problems," Transportation Research, Vol. 24B, pp. 423-430.

Hearn, D. W., Lawphongpanish, S., and Nguyen, S., 1984. "Convex Programming Formulation of the Asymmetric Traffic Assignment Problem," Transportation Research, Vol. 18B, pp. 357-365.

Hearn, D. W., Lawphongpanish, S., and Ventura, J. A., 1985. "Finiteness in Restricted Simplicial Decomposition," Operations Research Letters, Vol. 4, pp. 125-130.

Hearn, D. W., Lawphongpanish, S., and Ventura, J. A., 1987. "Restricted Simplicial Decomposition: Computation and Extensions," Math. Programming Studies, Vol. 31, pp. 99-118.

Held, M., and Karp, R. M., 1970. "The Traveling Salesman Problem and Minimum Spanning Trees," Operations Research, Vol. 18, pp. 1138-1162.

Held, M., and Karp, R. M., 1971. "The Traveling Salesman Problem and Minimum Spanning Trees: Part II," Math. Programming, Vol. 1, pp. 6-25.

Helgason, R. V., and Kennington, J. L., 1977. "An Efficient Procedure for Implementing a Dual-Simplex Network Flow Algorithm," AIIE Transactions, Vol. 9, pp. 63-68.

Helgason, R. V., and Kennington, J. L., 1995. "Primal-Simplex Algorithms for Minimum Cost Network Flows," Handbooks in OR and MS, Ball, M. O., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L. (eds.), Vol. 7, North-Holland, Amsterdam, pp. 85-133.

Helgason, R. V., Kennington, J. L., and Stewart, B. D., 1993. "The One-to-One Shortest-Path Problem: An Empirical Analysis with the Two-Tree Dijkstra Algorithm," Computational Optimization and Applications, Vol. 1, pp. 47-75.

Hiriart-Urruty, J.-B., and Lemarechal, C., 1993. Convex Analysis and Minimization Algorithms, Vols. I and II, Springer-Verlag, Berlin and N. Y.

Hochbaum, D. S., and Shantikumar, J. G., 1990. "Convex Separable Optimization is not Much Harder than Linear Optimization," J. ACM, Vol. 37, pp. 843-862.

Hoffman, A. J., 1960. "Some Recent Applications of the Theory of Linear Inequalities to Extremal Combinatorial Analysis," Proc. Symp. Appl. Math., Vol. 10, pp. 113-128.

Hoffman, A. J., and Kuhn, H. W., 1956. "Systems of Distinct Representatives and Linear Programming," Amer. Math. Monthly, Vol. 63, pp. 455-460.

Hoffman, K., and Kunze, R., 1971. Linear Algebra, Prentice-Hall, Englewood Cliffs, N. J.

Holloway, C. A., 1974. "An Extension of the Frank and Wolfe Method of Feasible Directions," Math. Programming, Vol. 6, pp. 14-27.

Hopcroft, J. E., and Karp, R. M., 1973. "A $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs," SIAM J. on Computing, Vol. 2, pp. 225-231.

Horst, R., Pardalos, P. M., and Thoai, N. V., 1995. Introduction to Global Optimization, Kluwer Academic Publishers, N. Y.

Hu, T. C., 1969. Integer Programming and Network Flows, Addison-Wesley, Reading, MA.

Hung, M., 1983. "A Polynomial Simplex Method for the Assignment Problem," Operations Research, Vol. 31, pp. 595-600.

Ibaraki, T., and Katoh, N., 1988. Resource Allocation Problems: Algorithmic Approaches, M.I.T. Press, Cambridge, MA.

Iri, M., 1969. Network Flows, Transportation, and Scheduling, Academic Press, N. Y.

Iusem, A. N., Svaiter, B., and Teboulle, M., 1994. "Entropy-Like Proximal Methods in Convex Programming," Math. Operations Research, Vol. 19, pp. 790-814.

Jensen, P. A., and Barnes, J. W., 1980. Network Flow Programming, Wiley, N. Y.

Jewell, W. S., 1962. "Optimal Flow Through Networks with Gains," Operations Research, Vol. 10, pp. 476-499.

Johnson, D. B., 1977. "Efficient Algorithms for Shortest Paths in Sparse Networks," J. ACM, Vol. 24, pp. 1-13.

Johnson, D. S., and Papadimitriou, C. H., 1985. "Computational Complexity," in The Traveling Salesman Problem, Lawler, E., Lenstra, J. K., Rinnoy Kan, A. H. G., and Shmoys, D. B. (eds.), Wiley, N. Y., pp. 37-85.

Johnson, D. S., and McGeoch, L., 1997. "The Traveling Salesman Problem:

A Case Study," in Local Search in Combinatorial Optimization, Aarts, E., and Lenstra, J. K. (eds.), Wiley, N. Y.

Johnson, E. L., 1966. "Networks and Basic Solutions," Operations Research, Vol. 14, pp. 619-624.

Johnson, E. L., 1972. "On Shortest Paths and Sorting," Proc. 25th ACM Annual Conference, pp. 510-517.

Jonker, R., and Volgenant, A., 1986. "Improving the Hungarian Assignment Algorithm," Operations Research Letters, Vol. 5, pp. 171-175.

Jonker, R., and Volgenant, A., 1987. "A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems," Computing, Vol. 38, pp. 325-340.

Junger, M., Reinelt, G., and Rinaldi, G., 1995. "The Traveling Salesman Problem," Handbooks in OR and MS, Ball, M. O., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L. (eds.), Vol. 7, North-Holland, Amsterdam, pp. 225-330.

Karzanov, A. V., 1974. "Determining the Maximal Flow in a Network with the Method of Preflows," Soviet Math Dokl., Vol. 15, pp. 1277-1280.

Karzanov, A. V., and McCormick, S. T., 1997. "Polynomial Methods for Separable Convex Optimization in Unimodular Linear Spaces with Applications to Circulations and Co-circulations in Network," SIAM J. on Computing, Vol. 26, pp. 1245-1275.

Kelley, J. E., 1960. "The Cutting-Plane Method for Solving Convex Programs," J. Soc. Indust. Appl. Math., Vol. 8, pp. 703-712.

Kennington, J., and Helgason, R., 1980. Algorithms for Network Programming, Wiley, N. Y.

Kennington, J., and Shalaby, M., 1977. "An Effective Subgradient Procedure for Minimal Cost Multicommodity Flow Problems," Management Science, Vol. 23, pp. 994-1004.

Kernighan, B. W., and Lin, S., 1970. "An Efficient Heuristic Procedure for Partitioning Graphs," Bell System Tech. Journal, Vol. 49, pp. 291-307.

Kershenbaum, A., 1981. "A Note on Finding Shortest Path Trees," Networks, Vol. 11, pp. 399-400.

Kershenbaum, A., 1993. Network Design Algorithms, McGraw-Hill, N. Y.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., 1983. "Optimization by Simulated Annealing," Science, Vol. 220, pp. 621-680.

Kiwiel, K. C., 1997a. "Proximal Minimization Methods with Generalized Bregman Functions," SIAM J. on Control and Optimization, Vol. 35, pp. 1142-1168.

Kiwiel, K. C., 1997b. "Efficiency of the Analytic Center Cutting Plane Method for Convex Minimization," SIAM J. on Optimization, Vol. 7, pp. 336-346.

Klee, V., and Minty, G. J., 1972. "How Good is the Simplex Algorithm?," in Inequalities III, O. Shisha (ed.), Academic Press, N. Y., pp. 159-175.

Klein, M., 1967. "A Primal Method for Minimal Cost Flow with Applications to the Assignment and Transportation Problems," Management Science, Vol. 14, pp. 205-220.

Klessig, R. W., 1974. "An Algorithm for Nonlinear Multicommodity Flow Problems," Networks, Vol. 4, pp. 343-355.

Klincewitz, J. C., 1989. "Implementing an Exact Newton Method for Separable Convex Transportation Problems," Networks, Vol. 19, pp. 95-105.

König, D., 1931. "Graphok es Matrixok," Mat. Es Fiz. Lapok, Vol. 38, pp. 116-119.

Korst, J., Aarts, E. H., and Korst, A., 1989. Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing, Wiley, N. Y.

Kortanek, K. O., and No, H., 1993. "A Central Cutting Plane Algorithm for Convex Semi-Infinite Programming Problems," SIAM J. on Optimization, Vol. 3, pp. 901-918.

Kuhn, H. W., 1955. "The Hungarian Method for the Assignment Problem," Naval Research Logistics Quarterly, Vol. 2, pp. 83-97.

Kumar, V., Grama, A., Gupta, A., and Karypis, G., 1994. Introduction to Parallel Computing, Benjamin/Cummings, Redwood City, CA.

Kushner, H. J., 1990. "Numerical Methods for Continuous Control Problems in Continuous Time," SIAM J. on Control and Optimization, Vol. 28, pp. 999-1048.

Kushner, H. J., and Dupuis, P. G., 1992. Numerical Methods for Stochastic Control Problems in Continuous Time, Springer-Verlag, N. Y.

Kwan Mei-Ko, 1962. "Graphic Programming Using Odd or Even Points," Chinese Math., Vol. 1, pp. 273-277.

Lamar, B. W., 1993. "An Improved Branch and Bound Algorithm for Minimum Concave Cost Network Flow Problems," in Network Optimization Problems, Du, D.-Z., and Pardalos, P. M. (eds.), World Scientific Publ., Singapore, pp. 261-287.

Land, A. H., and Doig, A. G., 1960. "An Automatic Method for Solving Discrete Programming Problems," Econometrica, Vol. 28, pp. 497-520.

Larsson, T., and Patricksson, M., 1992. "Simplicial Decomposition with

Disaggregated Representation for the Traffic Assignment Problem," Transportation Science, Vol. 26, pp. 4-17.

Lasdon, L. S., 1970. Optimization Theory for Large Systems, Macmillian, N. Y.

Lawphongpanich, S., and Hearn, D., 1984. "Simplicial Decomposition of the Asymmetric Traffic Assignment Problems," Transportation Research, Vol. 18B, pp. 123-133.

Lawphongpanich, S., and Hearn, D. W., 1986. "Restricted Simplicial Decomposition with Application to the Traffic Assignment Problem," Ricerca Operativa, Vol. 38, pp. 97-120.

Lawler, E., 1976. Combinatorial Optimization: Networks and Matroids, Holt, Reinhart, and Winston, N. Y.

Lawler, E., Lenstra, J. K., Rinnoy Kan, A. H. G., and Shmoys, D. B., 1985. The Traveling Salesman Problem, Wiley, N. Y.

LeBlanc, L. J., Helgason, R. V., and Boyce, D. E., 1985. "Improved Efficiency of the Frank-Wolfe Algorithm for Convex Network Programs," Transportation Science, Vol. 19, pp. 445–462.

LeBlanc, L. J., Morlok, E. K., and Pierskalla, W. P., 1974. "An Accurate and Efficient Approach to Equilibrium Traffic Assignment on Congested Networks," Transportation Research Record, TRB-National Academy of Sciences, Vol. 491, pp. 12-23.

LeBlanc, L. J., Morlok, E. K., and Pierskalla, W. P., 1975. "An Efficient Approach to Solving the Road Network Equilibrium Traffic Assignment Problem," Transportation Research, Vol. 9, pp. 309-318.

Leventhal, T., Nemhauser, G., and Trotter, Jr., L., 1973. "A Column Generation Algorithm for Optimal Traffic Assignment," Transportation Science, Vol. 7, pp. 168-176.

Lemarechal, C., 1974. "An Algorithm for Minimizing Convex Functions," in Information Processing '74, Rosenfeld, J. L. (ed.), North Holland Publ. Co., Amsterdam, pp. 552-556.

Little, J. D. C., Murty, K. G., Sweeney, D. W., and Karel, C., 1963. "An Algorithm for the Traveling Salesman Problem," Operations Research, Vol. 11, pp. 972-989.

Lovasz, L., and Plummer, M. D., 1985. Matching Theory, North-Holland, Amsterdam.

Luenberger, D. G., 1969. Optimization by Vector Space Methods, Wiley, N. Y.

Luenberger, D. G., 1984. Linear and Nonlinear Programming, Addison-Wesley, Reading, MA.

Luo, Z.-Q., 1997. "Analysis of a Cutting Plane Method that Uses Weighted Analytic Center and Multiple Cuts," SIAM J. of Optimization, Vol. 7, pp. 697-716.

Luo, Z.-Q., and Tseng, P., 1994. "On the Rate of Convergence of a Distributed Asynchronous Routing Algorithm," IEEE Trans. on Automatic Control, Vol. 39, pp. 1123-1129.

Malhotra, V. M., Kumar, M. P., and Maheshwari, S. N., 1978. "An $O(|V|^3)$ Algorithm for Finding Maximum Flows in Networks," Inform. Process. Lett., Vol. 7, pp. 277-278.

Marcotte, P., 1985. "A New Algorithm for Solving Variational Inequalities with Application to the Traffic Assignment Problem," Math. Programming Studies, Vol. 33, pp. 339-351.

Marcotte, P., and Dussault, J.-P., 1987. "A Note on a Globally Convergent Newton Method for Solving Monotone Variational Inequalities," Operations Research Letters, Vol. 6, pp. 35-42.

Marcotte, P., and Guélat, J., 1988. "Adaptation of a Modified Newton Method for Solving the Asymmetric Traffic Equilibrium Problem," Transportation Science, Vol. 22, pp. 112-124.

Martello, S., and Toth, P., 1990. Knapsack Problems, Wiley, N. Y.

Martinet, B., 1970. "Regularisation d'Inequations Variationnelles par Approximations Successives," Rev. Francaise Inf. Rech. Oper., Vol. 4, pp. 154-159.

McGinnis, L. F., 1983. "Implementation and Testing of a Primal-Dual Algorithm for the Assignment Problem," Operations Research, Vol. 31, pp. 277-291.

Mendelssohn, N. S., and Dulmage, A. L., 1958. "Some Generalizations of Distinct Representatives," Canad. J. Math., Vol. 10, pp. 230-241.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E., 1953. "Equation of State Calculations by Fast Computing Machines," J. of Chemical Physisc, Vol. 21, pp. 1087-1092.

Meyer, R. R., 1979. "Two-Segment Separable Programming," Management Science, Vol. 25, pp. 385-395.

Miller, D., Pekny, J., and Thompson, G. L., 1990. "Solution of Large Dense Transportation Problems Using a Parallel Primal Algorithm," Operations Research Letters, Vol. 9, pp. 319-324.

Minty, G. J., 1957. "A Comment on the Shortest Route Problem," Operations Research, Vol. 5, p. 724.

Minty, G. J., 1960. "Monotone Networks," Proc. Roy. Soc. London, A, Vol. 257, pp. 194-212.

Minieka, E., 1978. Optimization Algorithms for Networks and Graphs, Marcel Dekker, N. Y.

Minoux, M., 1986a. Mathematical Programming: Theory and Algorithms, Wiley, N. Y.

Minoux, M., 1986b. "Solving Integer Minimum Cost Flows with Separable Convex Cost Objective Polynomially," Math. Programming Studies, Vol. 26, pp. 237-239.

Minoux, M., 1989. "Network Synthesis and Optimum Network Design Problems: Models, Solution Methods,and Applications," Networks, Vol. 19, pp. 313-360.

Monma, C. L., and Sheng, D. D., 1986. "Backbone Network Design and Performance Analysis: A Methodology for Packet Switching Networks," IEEE J. Select. Areas Comm., Vol. SAC-4, pp. 946-965.

Mulvey, J., 1978a. "Pivot Strategies for Primal-Simplex Network Codes," J. ACM, Vol. 25, pp. 266-270.

Mulvey, J., 1978b. "Testing a Large-Scale Network Optimization Program," Math. Programming, Vol. 15, pp. 291-314.

Murty, K. G., 1992. Network Programming, Prentice-Hall, Englewood Cliffs, N. J.

Nagurney, A., 1988. "An Equilibration Scheme for the Traffic Assignment Problem with Elastic Demands," Transportation Research, Vol. 22B, pp. 73-79.

Nagurney, A., 1993. Network Economics: A Variational Inequality Approach, Kluwer, Dordrecht, The Netherlands.

Nemhauser, G. L., and Wolsey, L. A., 1988. Integer and Combinatorial Optimization, Wiley, N. Y.

Nesterov, Y., 1995. "Complexity Estimates of Some Cutting Plane Methods Based on Analytic Barrier," Math. Programming, Vol. 69, pp. 149-176.

Nesterov, Y., and Nemirovskii, A., 1994. Interior Point Polynomial Algorithms in Convex Programming, SIAM, Phila., PA.

Nguyen, S., 1974. "An Algorithm for the Traffic Assignment Problem," Transportation Science, Vol. 8, pp. 203-216.

Nicholson, T., 1966. "Finding the Shortest Route Between Two Points in a Network," The Computer Journal, Vol. 9, pp. 275-280.

Nilsson, N. J., 1971. Problem-Solving Methods in Artificial Intelligence, McGraw-Hill, N. Y.

Nilsson, N. J., 1980. Principles of Artificial Intelligence, Tioga, Palo Alto, CA.

O'hEigeartaigh, M., Lenstra, S. K., and Rinnoy Kan, A. H. G. (eds.), 1985. Combinatorial Optimization: Annotated Bibliographies, Wiley, N. Y.

Ortega, J. M., and Rheinboldt, W. C., 1970. Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, N. Y.

Osman, I. H., and Laporte, G., 1996. "Metaheuristics: A Bibliography," Annals of Operations Research, Vol. 63, pp. 513-628.

Padberg, M. W., and Grötschel, M., 1985. "Polyhedral Computations," in The Traveling Salesman Problem, Lawler, E., Lenstra, J. K., Rinnoy Kan, A. H. G., and Shmoys, D. B. (eds.), Wiley, N. Y., pp. 307-360.

Pallottino, S., 1984. "Shortest Path Methods: Complexity, Interrelations and New Propositions," Networks, Vol. 14, pp. 257-267.

Pallottino, S., and Scutellà, M. G., 1991. "Strongly Polynomial Algorithms for Shortest Paths," Ricerca Operativa, Vol. 60, pp. 33-53.

Pallottino, S., and Scutellà, M. G., 1997a. "Shortest Path Algorithms in Transportation Models: Classical and Innovative Aspects," Proc. of the International Colloquium on Equilibrium in Transportation Models, Montreal, Canada.

Pallottino, S., and Scutellà, M. G., 1997b. "Dual Algorithms for the Shortest Path Tree Problem," Networks, Vol. 29, pp. 125-133.

Pang, J.-S., 1984. "Solution of the General Multicommodity Spatial Equilibrium Problem by Variational and Complementarity Methods," J. of Regional Science, Vol. 24, pp. 403-414.

Pang, J.-S., and Yu, C.-S., 1984. "Linearized Simplicial Decomposition Methods for Computing Traffic Equilibria on Networks," Networks, Vol. 14, pp. 427-438.

Papadimitriou, C. H., and Steiglitz, K., 1982. Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, Englewood Cliffs, N. J.

Pape, U., 1974. "Implementation and Efficiency of Moore - Algorithms for the Shortest Path Problem," Math. Programming, Vol. 7, pp. 212-222.

Pardalos, P. M., and Rosen, J. B., 1987. Constrained Global Optimization: Algorithms and Applications, Springer-Verlag, N. Y.

Patricksson, M., 1991. "Algorithms for Urban Traffic Network Equilibria," Linköping Studies in Science and Technology, Department of Mathematics, Thesis No. 263, Linköping University, Linköping, Sweden.

Pattipati, K. R., and Alexandridis, M. G., 1990. "Application of Heuristic Search and Information Theory to Sequential Fault Diagnosis," IEEE Trans. on Systems, Man, and Cybernetics, Vol. 20, pp. 872-887.

Pattipati, K. R., Deb, S., Bar-Shalom, Y., and Washburn, R. B., 1992. "A

New Relaxation Algorithm and Passive Sensor Data Association," IEEE Trans. Automatic Control, Vol. 37, pp. 198-213.

Pearl, J., 1984. Heuristics, Addison-Wesley, Reading, MA.

Peters, J., 1990. "The Network Simplex Method on a Multiprocessor," Networks, Vol. 20, pp. 845-859.

Phillips, C., and Zenios, S. A., 1989. "Experiences with Large Scale Network Optimization on the Connection Machine," in The Impact of Recent Computing Advances on Operations Research, Vol. 9, Elsevier, Amsterdam, The Netherlands, pp. 169-180.

Pinar, M. C., and Zenios, S. A., 1992. "Parallel Decomposition of Multicommodity Network Flows Using a Linear-Quadratic Penalty Algorithm," ORSA J. on Computing, Vol. 4, pp. 235-249.

Pinar, M. C., and Zenios, S. A., 1993. "Solving Nonlinear Programs with Embedded Network Structures," in Network Optimization Problems, Du, D.-Z., and Pardalos, P. M. (eds.), World Scientific Publ., Singapore, pp. 177-202.

Pinar, M. C., and Zenios, S. A., 1994. "On Smoothing Exact Penalty Functions for Convex Constrained Optimization," SIAM J. on Optimization, Vol. 4, pp. 486-511.

Pinedo, M., 1995. Scheduling: Theory, Algorithms, and Systems, Prentice-Hall, Englewood Cliffs, N. J.

Poljak, B. T., 1987. Introduction to Optimization, Optimization Software Inc., N. Y.

Polymenakos, L. C., 1995. "ϵ-Relaxation and Auction Algorithms for the Convex Cost Network Flow Problem," Ph.D. Thesis, Electrical Engineering and Computer Science Dept, M.I.T., Cambridge, MA.

Polymenakos, L. C., and Bertsekas, D. P., 1994. "Parallel Shortest Path Auction Algorithms," Parallel Computing, Vol. 20, pp. 1221-1247.

Polymenakos, L. C., Bertsekas, D. P., and Tsitsiklis, J. N., 1998. "Efficient Algorithms for Continuous-Space Shortest Path Problems," IEEE Trans. on Automatic Control, Vol. AC-43, pp. 278-283.

Poore, A. B., 1994. "Multidimensional Assignment Formulation of Data Association Problems Arising from Multitarget Tracking and Multisensor Data Fusion," Computational Optimization and Applications, Vol. 3, pp. 27-57.

Poore, A. B., and Robertson, A. J. A., 1997. New Lagrangian Relaxation Based Algorithm for a Class of Multidimensional Assignment Problems," Computational Optimization and Applications, Vol. 8, pp. 129-150.

Powell, W. B., Jaillet, P., and Odoni, A., 1995. "Stochastic and Dynamic

Networks and Routing," Handbooks in OR and MS, Ball, M. O., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L. (eds.), Vol. 8, North-Holland, Amsterdam, pp. 141-295.

Powell, W. B., Berkkam, E., and Lustig, I. J., 1993. "On Algorithms for Nonlinear Dynamic Networks," in Network Optimization Problems, Du, D.-Z., and Pardalos, P. M. (eds.), World Scientific Publ., Singapore, pp. 177-202.

Pulleyblank, W., 1983. "Polyhedral Combinatorics," in Mathematical Programming: The State of the Art - Bonn 1982, by Bachem, A., Grötschel, M., and Korte, B., (eds.), Springer, Berlin, pp. 312-345.

Pulleyblank, W., Cook, W., Cunningham, W., and Schrijver, A., 1993. An Introduction to Combinatorial Optimization, Wiley, N. Y.

Resende, M. G. C., and Veiga, G., 1993. "An Implementation of the Dual Affine Scaling Algorithm for Minimum-Cost Flow on Bipartite Uncapacitated Networks," SIAM J. on Optimization, Vol. 3, pp. 516-537.

Resende, M. G. C., and Pardalos, P. M., 1996. "Interior Point Algorithms for Network Flow Problems," Advances in Linear and Integer Programming, Oxford Lecture Ser. Math. Appl., Vol. 4, Oxford Univ. Press, New York, pp. 145-185.

Rockafellar, R. T., 1967. "Convex Programming and Systems of Elementary Monotonic Relations," J. of Math. Analysis and Applications, Vol. 19, pp. 543-564.

Rockafellar, R. T., 1969. "The Elementary Vectors of a Subspace of R^N," in Combinatorial Mathematics and its Applications, by Bose, R. C., and Dowling, T. A. (eds.), University of North Carolina Press, pp. 104-127.

Rockafellar, R. T., 1970. Convex Analysis, Princeton Univ. Press, Princeton, N. J.

Rockafellar, R. T., 1976. "Monotone Operators and the Proximal Point Algorithm," SIAM J. on Control and Optimization, Vol. 14, pp. 877-898.

Rockafellar, R. T., 1981. "Monotropic Programming: Descent Algorithms and Duality," in Nonlinear Programming 4, by Mangasarian, O. L., Meyer, R. R., and Robinson, S. M. (eds.), Academic Press, N. Y., pp. 327-366.

Rockafellar, R. T., 1984. Network Flows and Monotropic Programming, Wiley, N. Y.

Rudin, W., 1976. Real Analysis, McGraw Hill, N. Y.

Sahni, S., and Gonzalez, T., 1976. "P-Complete Approximation Problems," J. ACM, Vol. 23, pp. 555-565.

Schwartz, B. L., 1994. "A Computational Analysis of the Auction Algorithm," Eur. J. of Operations Research, Vol. 74, pp. 161-169.

Sheffi, Y., 1985. Urban Transportation Networks. Equilibrium Analysis with Mathematical Programming Methods, Prentice-Hall, Englewood Cliffs, N. J.

Shier, D. R., 1979. "On Algorithms for Finding the K Shortest Paths in a Network," Networks, Vol. 9, pp. 195-214.

Shier, D. R., and Witzgall, C., 1981. "Properties of Labeling Methods for Determining Shortest Path Trees," J. Res. Natl. Bureau of Standards, Vol. 86, pp. 317-330.

Shiloach, Y., and Vishkin, U., 1982. "An $O(n^2 \log n)$ Parallel Max-Flow Algorithm," J. Algorithms, Vol. 3, pp. 128-146.

Schrijver, A., 1986. Theory of Linear and Integer Programming, Wiley, N. Y.

Shapiro, J. E., 1979. Mathematical Programming Structures and Algorithms, Wiley, N. Y.

Shor, N. Z., 1985. Minimization Methods for Nondifferentiable Functions, Springer-Verlag, Berlin.

Srinivasan, V., and Thompson, G. L., 1973. "Benefit-Cost Analysis of Coding Techniques for Primal Transportation Algorithm," J. ACM, Vol. 20, pp. 194-213.

Strang, G., 1976. Linear Algebra and Its Applications, Academic Press, N. Y.

Suchet, C., 1949. Electrical Engineering, Vol. 68, pp. 843-844.

Tabourier, Y., 1973. "All Shortest Distances in a Graph: An Improvement to Dantzig's Inductive Algorithm," Disc. Math., Vol. 4, pp. 83-87.

Tardos, E., 1985. "A Strongly Polynomial Minimum Cost Circulation Algorithm," Combinatorica, Vol. 5, pp. 247-255.

Teboulle, M., 1992. "Entropic Proximal Mappings with Applications to Nonlinear Programming," Math. of Operations Research, Vol. 17, pp. 1-21.

Toint, P. L., and Tuyttens, D., 1990. "On Large Scale Nonlinear Network Optimization," Math. Programming, Vol. 48, pp. 125-159.

Tseng, P., 1986. "Relaxation Methods for Monotropic Programming Problems," Ph.D. Thesis, Dept. of Electrical Engineering and Computer Science, M.I.T., Cambridge, MA.

Tseng, P., 1991. "Relaxation Method for Large Scale Linear Programming Using Decomposition," Math. of Operations Research, Vol. 17, pp. 859-880.

Tseng, P., 1998. "An ϵ-Out-of-Kilter Method for Monotropic Programming," Department of Mathematics Report, Univ. of Washington, Seattle, Wash.

Tseng, P., and Bertsekas, D. P., 1987. "Relaxation Methods for Linear Programs," Math. of Operations Research, Vol. 12, pp. 569-596.

Tseng, P., and Bertsekas, D. P., 1990. "Relaxation Methods for Monotropic Programs," Math. Programming, Vol. 46, 1990, pp. 127-151.

Tseng, P., and Bertsekas, D. P., 1993. "On the Convergence of the Exponential Multiplier Method for Convex Programming," Math. Programming, Vol. 60, pp. 1-19.

Tseng, P., and Bertsekas, D. P., 1996. "An Epsilon-Relaxation Method for Separable Convex Cost Generalized Network Flow Problems," Lab. for Information and Decision Systems Report P-2374, M.I.T., Cambridge, MA.

Tseng, P., Bertsekas, D. P., and Tsitsiklis, J. N., 1990. "Partially Asynchronous Parallel Algorithms for Network Flow and Other Problems," SIAM J. on Control and Optimization, Vol. 28, pp. 678-710.

Tsitsiklis, J. N., 1989. "Markov Chains with Rare Transitions and Simulated Annealing," Math. of Operations Research, Vol. 14, pp. 70-90.

Tsitsiklis, J. N., 1992. "Special Cases of Traveling Salesman and Repairman Problems with Time Windows," Networks, Vol. 22, pp. 263-282.

Tsitsiklis, J. N., 1995. "Efficient Algorithms for Globally Optimal Trajectories," IEEE Trans. on Automatic Control, Vol. 40, pp. 1528-1538.

Tsitsiklis, J. N., and Bertsekas, D. P., 1986. "Distributed Asynchronous Optimal Routing in Data Networks," IEEE Trans. on Automatic Control, Vol. 31, pp. 325-331.

Ventura, J. A., and Hearn, D. W., 1993. "Restricted Simplicial Decomposition for Convex Constrained Problems," Math. Programming, Vol. 59, pp. 71-85.

Voß, S., 1992. "Steiner's Problem in Graphs: Heuristic Methods,", Discrete Applied Math., Vol. 40, pp. 45-72.

Von Randow, R., 1982. Integer Programming and Related Areas: A Classified Bibliography 1978-1981, Lecture Notes in Economics and Mathematical Systems, Vol. 197, Springer-Verlag, N. Y.

Von Randow, R., 1985. Integer Programming and Related Areas: A Classified Bibliography 1982-1984, Lecture Notes in Economics and Mathematical Systems, Vol. 243, Springer-Verlag, N. Y.

Warshall, S., 1962. "A Theorem on Boolean Matrices," J. ACM, Vol. 9, pp. 11-12.

Wein, J., and Zenios, S. A., 1991. "On the Massively Parallel Solution of the Assignment Problem," J. of Parallel and Distributed Computing, Vol.

13, pp. 228-236.

Whitting, P. D., and Hillier, J. A., 1960. “A Method for Finding the Shortest Route Through a Road Network,” Operations Research Quart., Vol. 11, pp. 37-40.

Winter, P., 1987. “Steiner Problem in Networks: A Survey,” Networks, Vol. 17, pp. 129-167.

Wright, S. J., 1997. Primal-Dual Interior Point Methods, SIAM, Phila., PA.

Ye, Y., 1992. “A Potential Reduction Algorithm Allowing Column Generation,” SIAM J. on Optimization, Vol. 2, pp. 7-20.

Ye, Y., 1997. Interior Point Algorithms: Theory and Analysis, Wiley, N. Y.

Zadeh, N., 1973a. “A Bad Network Problem for the Simplex Method and Other Minimum Cost Flow Algorithms,” Math. Programming, Vol. 5, pp. 255-266.

Zadeh, N., 1973b. “More Pathological Examples for Network Flow Problems,” Math. Programming, Vol. 5, pp. 217-224.

Zadeh, N., 1979. “Near Equivalence of Network Flow Algorithms,” Technical Report No. 26, Dept. of Operations Research, Stanford University, CA.

Zenios, S. A., and Mulvey, J. M., 1986. “Relaxation Techniques for Strictly Convex Network Problems,” Annals of Operations Research, Vol. 5, pp. 517-538.

Zoutendijk, G., 1976. Mathematical Programming Methods, North Holland, Amsterdam.