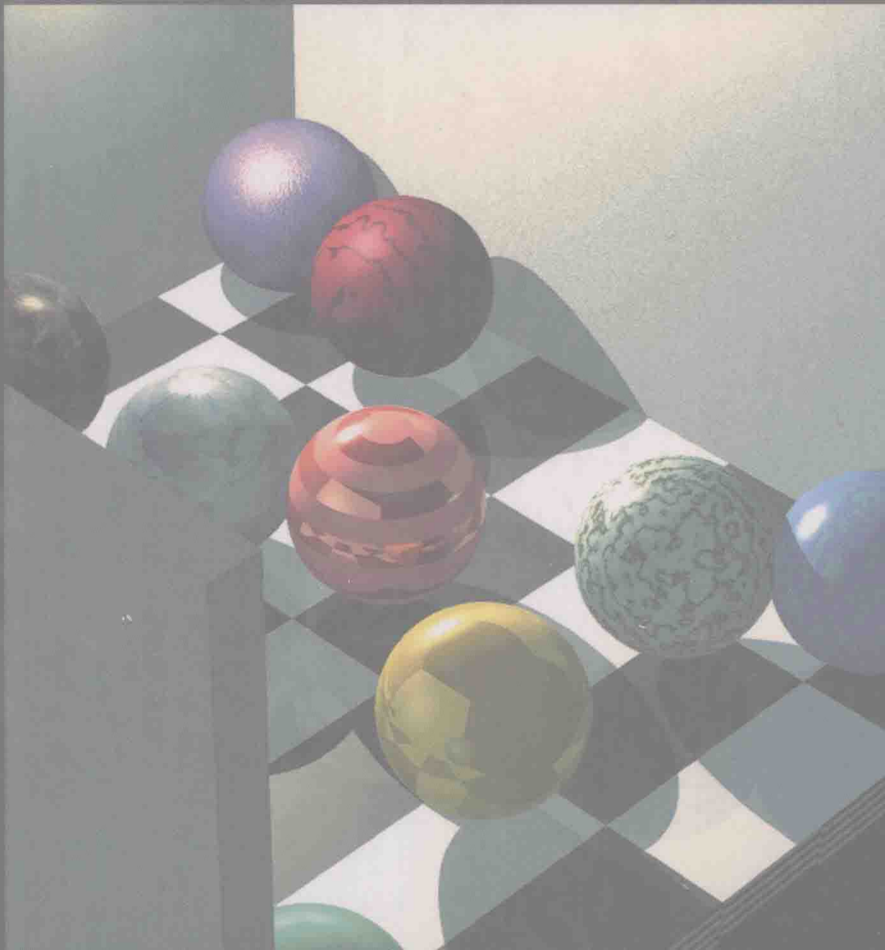
Introduction to LINEAR OPTIMIZATION

Dimitris Bertsimas
John N. Tsitsiklis



Introduction to Linear Optimization

Dimitris Bertsimas
John N. Tsitsiklis

Massachusetts Institute of Technology

Athena Scientific, Belmont, Massachusetts

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

Email: athenasc@world.std.com
WWW information and orders: http://world.std.com/˜athenasc/

Cover Design: *Ann Gallager*

Publisher's Cataloging-in-Publication Data

Bertsimas, Dimitris, Tsitsiklis, John N.
Introduction to Linear Optimization
Includes bibliographical references and index
1. Linear programming. 2. Mathematical optimization.
3. Integer programming. I. Title.
T57.74.B465 1997 519.7 96-78786

ISBN 1-886529-19-1

Preface

The purpose of this book is to provide a unified, insightful, and modern treatment of linear optimization, that is, linear programming, network flow problems, and discrete linear optimization. We discuss both classical topics, as well as the state of the art. We give special attention to theory, but also cover applications and present case studies. Our main objective is to help the reader become a sophisticated practitioner of (linear) optimization, or a researcher. More specifically, we wish to develop the ability to formulate fairly complex optimization problems, provide an appreciation of the main classes of problems that are practically solvable, describe the available solution methods, and build an understanding of the qualitative properties of the solutions they provide.

Our general philosophy is that insight matters most. For the subject matter of this book, this necessarily requires a geometric view. On the other hand, problems are solved by algorithms, and these can only be described algebraically. Hence, our focus is on the beautiful interplay between algebra and geometry. We build understanding using figures and geometric arguments, and then translate ideas into algebraic formulas and algorithms. Given enough time, we expect that the reader will develop the ability to pass from one domain to the other without much effort.

Another of our objectives is to be comprehensive, but economical. We have made an effort to cover and highlight all of the principal ideas in this field. However, we have not tried to be encyclopedic, or to discuss every possible detail relevant to a particular algorithm. Our premise is that once mature understanding of the basic principles is in place, further details can be acquired by the reader with little additional effort.

Our last objective is to bring the reader up to date with respect to the state of the art. This is especially true in our treatment of interior point methods, large scale optimization, and the presentation of case studies that stretch the limits of currently available algorithms and computers.

The success of any optimization methodology hinges on its ability to deal with large and important problems. In that sense, the last chapter, on the art of linear optimization, is a critical part of this book. It will, we hope, convince the reader that progress on challenging problems requires both problem specific insight, as well as a deeper understanding of the underlying theory.

In any book dealing with linear programming, there are some important choices to be made regarding the treatment of the simplex method. Traditionally, the simplex method is developed in terms of the full simplex tableau, which tends to become the central topic. We have found that the full simplex tableau is a useful device for working out numerical examples. But other than that, we have tried not to overemphasize its importance.

Let us also mention another departure from many other textbooks. Introductory treatments often focus on standard form problems, which is sufficient for the purposes of the simplex method. On the other hand, this approach often leaves the reader wondering whether certain properties are generally true, and can hinder the deeper understanding of the subject. We depart from this tradition: we consider the general form of linear programming problems and define key concepts (e.g., extreme points) within this context. (Of course, when it comes to algorithms, we often have to specialize to the standard form.) In the same spirit, we separate the structural understanding of linear programming from the particulars of the simplex method. For example, we include a derivation of duality theory that does not rely on the simplex method.

Finally, this book contains a treatment of several important topics that are not commonly covered. These include a discussion of the column geometry and of the insights it provides into the efficiency of the simplex method, the connection between duality and the pricing of financial assets, a unified view of delayed column generation and cutting plane methods, stochastic programming and Benders decomposition, the auction algorithm for the assignment problem, certain theoretical implications of the ellipsoid algorithm, a thorough treatment of interior point methods, and a whole chapter on the practice of linear optimization. There are also several noteworthy topics that are covered in the exercises, such as Leontief systems, strict complementarity, options pricing, von Neumann's algorithm, submodular function minimization, and bounds for a number of integer programming problems.

Here is a chapter by chapter description of the book.

Chapter 1: Introduces the linear programming problem, together with a number of examples, and provides some background material on linear algebra.

Chapter 2: Deals with the basic geometric properties of polyhedra, focusing on the definition and the existence of extreme points, and emphasizing the interplay betwen the geometric and the algebraic viewpoints.

Chapter 3: Contains more or less the classical material associated with the simplex method, as well as a discussion of the column geometry. It starts with a high-level and geometrically motivated derivation of the simplex method. It then introduces the revised simplex method, and concludes with the simplex tableau. The usual topics of Phase I and anticycling are also covered.

Chapter 4: It is a comprehensive treatment of linear programming duality. The duality theorem is first obtained as a corollary of the simplex method. A more abstract derivation is also provided, based on the separating hyperplane theorem, which is developed from first principles. It ends with a deeper look into the geometry of polyhedra.

Chapter 5: Discusses sensitivity analysis, that is, the dependence of solutions and the optimal cost on the problem data, including parametric programming. It also develops a characterization of dual optimal solutions as subgradients of a suitably defined optimal cost function.

Chapter 6: Presents the complementary ideas of delayed column generation and cutting planes. These methods are first developed at a high level, and are then made concrete by discussing the cutting stock problem, Dantzig-Wolfe decomposition, stochastic programming, and Benders decomposition.

Chapter 7: Provides a comprehensive review of the principal results and methods for the different variants of the network flow problem. It contains representatives from all major types of algorithms: primal descent (the simplex method), dual ascent (the primal-dual method), and approximate dual ascent (the auction algorithm). The focus is on the major algorithmic ideas, rather than on the refinements that can lead to better complexity estimates.

Chapter 8: Includes a discussion of complexity, a development of the ellipsoid method, and a proof of the polynomiality of linear programming. It also discusses the equivalence of separation and optimization, and provides examples where the ellipsoid algorithm can be used to derive polynomial time results for problems involving an exponential number of constraints.

Chapter 9: Contains an overview of all major classes of interior point methods, including affine scaling, potential reduction, and path following (both primal and primal-dual) methods. It includes a discussion of the underlying geometric ideas and computational issues, as well as convergence proofs and complexity analysis.

Chapter 10: Introduces integer programming formulations of discrete optimization problems. It provides a number of examples, as well as some intuition as to what constitutes a "strong" formulation.

Chapter 11: Covers the major classes of integer programming algorithms, including exact methods (branch and bound, cutting planes, dynamic programming), approximation algorithms, and heuristic methods (local search and simulated annealing). It also introduces a duality theory for integer programming.

Chapter 12: Deals with the art in linear optimization, i.e., the process of modeling, exploiting problem structure, and fine tuning of optimization algorithms. We discuss the relative performance of interior point methods and different variants of the simplex method, in a realistic large scale setting. We also give some indication of the size of problems that can be currently solved.

An important theme that runs through several chapters is the modeling, complexity, and algorithms for problems with an exponential number constraints. We discuss modeling in Section 10.3, complexity in Section 8.5, algorithmic approaches in Chapter 6 and 8.5, and we conclude with a case study in Section 12.5.

There is a fair number of exercises that are given at the end of each chapter. Most of them are intended to deepen the understanding of the subject, or to explore extensions of the theory in the text, as opposed to routine drills. However, several numerical exercises are also included. Starred exercises are supposed to be fairly hard. A solutions manual for qualified instructors can be obtained from the authors.

We have made a special effort to keep the text as modular as possible, allowing the reader to omit certain topics without loss of continuity. For example, much of the material in Chapters 5 and 6 is rarely used in the rest of the book. Furthermore, in Chapter 7 (on network flow problems), a reader who has gone through the problem formulation (Sections 7.1-7.2) can immediately move to any later section in that chapter. Also, the interior point algorithms of Chapter 9 are not used later, with the exception of some of the applications in Chapter 12. Even within the core chapters (Chapters 1-4), there are many sections that can be skipped during a first reading. Some sections have been marked with a star indicating that they contain somewhat more advanced material that is not usually covered in an introductory course.

The book was developed while we took turns teaching a first-year graduate course at M.I.T., for students in engineering and operations research. The only prerequisite is a working knowledge of linear algebra. In fact, it is only a small subset of linear algebra that is needed (e.g., the concepts of subspaces, linear independence, and the rank of a matrix). However, these elementary tools are sometimes used in subtle ways, and some mathematical maturity on the part of the reader can lead to a better appreciation of the subject.

The book can be used to teach several different types of courses. The first two suggestions below are one-semester variants that we have tried at M.I.T., but there are also other meaningful alternatives, depending on the students' background and the course's objectives.

(a) Cover most of Chapters 1-7, and if time permits, cover a small number of topics from Chapters 9-12.

(b) An alternative could be the same as above, except that interior point algorithms (Chapter 9) are fully covered, replacing network flow problems (Chapter 7).

(c) A broad overview course can be constructed by concentrating on the easier material in most of the chapters. The core of such a course could consist of Chapter 1, Sections 2.1-2.4, 3.1-3.5, 4.1-4.3, 5.1, 7.1-7.3, 9.1, 10.1, some of the easier material in Chapter 11, and an application from Chapter 12.

(d) Finally, the book is also suitable for a half-course on integer programming, based on parts of Chapters 1 and 8, as well as Chapters 10-12.

There is a truly large literature on linear optimization, and we make no attempt to provide a comprehensive bibliography. To a great extent, the sources that we cite are either original references of historical interest, or recent texts where additional information can be found. For those topics, however, that touch upon current research, we also provide pointers to recent journal articles.

We would like to express our thanks to a number of individuals. We are grateful to our colleagues Dimitri Bertsekas and Rob Freund, for many discussions on the subjects in this book, as well as for reading parts of the manuscript. Several of our students, colleagues, and friends have contributed by reading parts of the manuscript, providing critical comments, and working on the exercises: Jim Christodouleas, Thalia Chryssikou, Austin Frakt, David Gamarnik, Leon Hsu, Spyros Kontogiorgis, Peter Marbach, Gina Mourtzinou, Yannis Paschalidis, Georgia Perakis, Lakis Polymenakos, Jay Sethuraman, Sarah Stock, Paul Tseng, and Ben Van Roy. But mostly, we are grateful to our families for their patience, love, and support in the course of this long project.

Dimitris Bertsimas
John N. Tsitsiklis
Cambridge, January 1997

Contents

Chapter 1

Introduction

Contents

In this chapter, we introduce *linear programming*, the problem of minimizing a linear cost function subject to linear equality and inequality constraints. We consider a few equivalent forms and then present a number of examples to illustrate the applicability of linear programming to a wide variety of contexts. We also solve a few simple examples and obtain some basic geometric intuition on the nature of the problem. The chapter ends with a review of linear algebra and of the conventions used in describing the computational requirements (operation count) of algorithms.

1.1 Variants of the linear programming problem

In this section, we pose the linear programming problem, discuss a few special forms that it takes, and establish some standard notation that we will be using. Rather than starting abstractly, we first state a concrete example, which is meant to facilitate understanding of the formal definition that will follow. The example we give is devoid of any interpretation. Later on, in Section 1.2, we will have ample opportunity to develop examples that arise in practical settings.

Example 1.1 The following is a linear programming problem:

$$\begin{array}{rrcrcrcrcl}
\text{minimize} & 2x_1 & - & x_2 & + & 4x_3 & & & & \\
\text{subject to} & x_1 & + & x_2 & & & + & x_4 & \leq & 2 \\
 & & & 3x_2 & - & x_3 & & & = & 5 \\
 & & & & & x_3 & + & x_4 & \geq & 3 \\
 & x_1 & & & & & & & \geq & 0 \\
 & & & & & x_3 & & & \leq & 0.
\end{array}$$

Here x_1, x_2, x_3, and x_4 are variables whose values are to be chosen to minimize the linear cost function $2x_1 - x_2 + 4x_3$, subject to a set of linear equality and inequality constraints. Some of these constraints, such as $x_1 \geq 0$ and $x_3 \leq 0$, amount to simple restrictions on the sign of certain variables. The remaining constraints are of the form $\mathbf{a}'\mathbf{x} \leq b$, $\mathbf{a}'\mathbf{x} = b$, or $\mathbf{a}'\mathbf{x} \geq b$, where $\mathbf{a} = (a_1, a_2, a_3, a_4)$ is a given vector[1], $\mathbf{x} = (x_1, x_2, x_3, x_4)$ is the vector of decision variables, $\mathbf{a}'\mathbf{x}$ is their inner product $\sum_{i=1}^4 a_i x_i$, and b is a given scalar. For example, in the first constraint, we have $\mathbf{a} = (1, 1, 0, 1)$ and $b = 2$.

We now generalize. In a *general* linear programming problem, we are given a cost vector $\mathbf{c} = (c_1, \ldots, c_n)$ and we seek to minimize a linear cost function $\mathbf{c}'\mathbf{x} = \sum_{i=1}^n c_i x_i$ over all n-dimensional vectors $\mathbf{x} = (x_1, \ldots, x_n)$,

[1] As discussed further in Section 1.5, all vectors are assumed to be column vectors, and are treated as such in matrix-vector products. Row vectors are indicated as transposes of (column) vectors. However, whenever we refer to a vector $\mathbf{x}$ inside the text, we use the more economical notation $\mathbf{x} = (x_1, \ldots, x_n)$, even though $\mathbf{x}$ is a column vector. The reader who is unfamiliar with our notation may wish to consult Section 1.5 before continuing.

subject to a set of linear equality and inequality constraints. In particular, let M_1, M_2, M_3 be some finite index sets, and suppose that for every i in any one of these sets, we are given an n-dimensional vector $\mathbf{a}_i$ and a scalar b_i, that will be used to form the ith constraint. Let also N_1 and N_2 be subsets of $\{1,\ldots,n\}$ that indicate which variables x_j are constrained to be nonnegative or nonpositive, respectively. We then consider the problem

$$\begin{array}{lrcll} \text{minimize} & \mathbf{c}'\mathbf{x} & & & \\ \text{subject to} & \mathbf{a}_i'\mathbf{x} & \geq & b_i, & i \in M_1, \\ & \mathbf{a}_i'\mathbf{x} & \leq & b_i, & i \in M_2, \\ & \mathbf{a}_i'\mathbf{x} & = & b_i, & i \in M_3, \\ & x_j & \geq & 0, & j \in N_1, \\ & x_j & \leq & 0, & j \in N_2. \end{array} \tag{1.1}$$

The variables $x_1,\ldots,x_n$ are called *decision variables*, and a vector $\mathbf{x}$ satisfying all of the constraints is called a *feasible solution* or *feasible vector*. The set of all feasible solutions is called the *feasible set* or *feasible region*. If j is in neither N_1 nor N_2, there are no restrictions on the sign of x_j, in which case we say that x_j is a *free* or *unrestricted* variable. The function $\mathbf{c}'\mathbf{x}$ is called the *objective function* or *cost function*. A feasible solution $\mathbf{x}^*$ that minimizes the objective function (that is, $\mathbf{c}'\mathbf{x}^* \leq \mathbf{c}'\mathbf{x}$, for all feasible $\mathbf{x}$) is called an *optimal feasible solution* or, simply, an *optimal solution*. The value of $\mathbf{c}'\mathbf{x}^*$ is then called the *optimal cost*. On the other hand, if for every real number K we can find a feasible solution $\mathbf{x}$ whose cost is less than K, we say that the optimal cost is $-\infty$ or that the cost is *unbounded below*. (Sometimes, we will abuse terminology and say that the problem is *unbounded*.) We finally note that there is no need to study maximization problems separately, because maximizing $\mathbf{c}'\mathbf{x}$ is equivalent to minimizing the linear cost function $-\mathbf{c}'\mathbf{x}$.

An equality constraint $\mathbf{a}_i'\mathbf{x} = b_i$ is equivalent to the two constraints $\mathbf{a}_i'\mathbf{x} \leq b_i$ and $\mathbf{a}_i'\mathbf{x} \geq b_i$. In addition, any constraint of the form $\mathbf{a}_i'\mathbf{x} \leq b_i$ can be rewritten as $(-\mathbf{a}_i)'\mathbf{x} \geq -b_i$. Finally, constraints of the form $x_j \geq 0$ or $x_j \leq 0$ are special cases of constraints of the form $\mathbf{a}_i'\mathbf{x} \geq b_i$, where $\mathbf{a}_i$ is a unit vector and $b_i = 0$. We conclude that the feasible set in a general linear programming problem can be expressed exclusively in terms of inequality constraints of the form $\mathbf{a}_i'\mathbf{x} \geq b_i$. Suppose that there is a total of m such constraints, indexed by $i = 1,\ldots,m$, let $\mathbf{b} = (b_1,\ldots,b_m)$, and let $\mathbf{A}$ be the $m \times n$ matrix whose rows are the row vectors $\mathbf{a}_1',\ldots,\mathbf{a}_m'$, that is,

$$\mathbf{A} = \begin{bmatrix} - & \mathbf{a}_1' & - \\ & \vdots & \\ - & \mathbf{a}_m' & - \end{bmatrix}.$$

Then, the constraints $\mathbf{a}_i'\mathbf{x} \geq b_i$, $i = 1,\ldots,m$, can be expressed compactly in the form $\mathbf{A}\mathbf{x} \geq \mathbf{b}$, and the linear programming problem can be written as

$$\begin{array}{ll} \text{minimize} & \mathbf{c}'\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} \geq \mathbf{b}. \end{array} \tag{1.2}$$

Inequalities such as $\mathbf{Ax} \geq \mathbf{b}$ will always be interpreted componentwise; that is, for every i, the ith component of the vector $\mathbf{Ax}$, which is $\mathbf{a}_i'\mathbf{x}$, is greater than or equal to the ith component b_i of the vector $\mathbf{b}$.

Example 1.2 The linear programming problem in Example 1.1 can be rewritten as

$$\begin{array}{lrcrcrcrcr} \text{minimize} & 2x_1 & - & x_2 & + & 4x_3 & & & & \\ \text{subject to} & -x_1 & - & x_2 & & & - & x_4 & \geq & -2 \\ & & & 3x_2 & - & x_3 & & & \geq & 5 \\ & & - & 3x_2 & + & x_3 & & & \geq & -5 \\ & & & & & x_3 & + & x_4 & \geq & 3 \\ & x_1 & & & & & & & \geq & 0 \\ & & & & - & x_3 & & & \geq & 0, \end{array}$$

which is of the same form as the problem (1.2), with $\mathbf{c} = (2, -1, 4, 0)$,

$$\mathbf{A} = \begin{bmatrix} -1 & -1 & 0 & -1 \\ 0 & 3 & -1 & 0 \\ 0 & -3 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix},$$

and $\mathbf{b} = (-2, 5, -5, 3, 0, 0)$.

Standard form problems

A linear programming problem of the form

$$\begin{array}{lrcl} \text{minimize} & \mathbf{c}'\mathbf{x} & & \\ \text{subject to} & \mathbf{Ax} & = & \mathbf{b} \\ & \mathbf{x} & \geq & \mathbf{0}, \end{array} \tag{1.3}$$

is said to be in *standard form.* We provide an interpretation of problems in standard form. Suppose that $\mathbf{x}$ has dimension n and let $\mathbf{A}_1, \ldots, \mathbf{A}_n$ be the columns of $\mathbf{A}$. Then, the constraint $\mathbf{Ax} = \mathbf{b}$ can be written in the form

$$\sum_{i=1}^{n} \mathbf{A}_i x_i = \mathbf{b}.$$

Intuitively, there are n available resource vectors $\mathbf{A}_1, \ldots, \mathbf{A}_n$, and a target vector $\mathbf{b}$. We wish to "synthesize" the target vector $\mathbf{b}$ by using a non-negative amount x_i of each resource vector $\mathbf{A}_i$, while minimizing the cost $\sum_{i=1}^{n} c_i x_i$, where c_i is the unit cost of the ith resource. The following is a more concrete example.

Example 1.3 (The diet problem) Suppose that there are n different foods and m different nutrients, and that we are given the following table with the nutritional content of a unit of each food:

	food 1	$\cdots$	food n
nutrient 1	a_{11}	$\cdots$	a_{1n}
$\vdots$	$\vdots$		$\vdots$
nutrient m	a_{m1}	$\cdots$	a_{mn}

Let $\mathbf{A}$ be the $m \times n$ matrix with entries a_{ij}. Note that the jth column $\mathbf{A}_j$ of this matrix represents the nutritional content of the jth food. Let $\mathbf{b}$ be a vector with the requirements of an ideal diet or, equivalently, a specification of the nutritional contents of an "ideal food." We then interpret the standard form problem as the problem of mixing nonnegative quantities x_i of the available foods, to synthesize the ideal food at minimal cost. In a variant of this problem, the vector $\mathbf{b}$ specifies the *minimal* requirements of an adequate diet; in that case, the constraints $\mathbf{Ax} = \mathbf{b}$ are replaced by $\mathbf{Ax} \geq \mathbf{b}$, and the problem is not in standard form.

Reduction to standard form

As argued earlier, any linear programming problem, including the standard form problem (1.3), is a special case of the general form (1.1). We now argue that the converse is also true and that a general linear programming problem can be transformed into an equivalent problem in standard form. Here, when we say that the two problems are equivalent, we mean that given a feasible solution to one problem, we can construct a feasible solution to the other, with the same cost. In particular, the two problems have the same optimal cost and given an optimal solution to one problem, we can construct an optimal solution to the other. The problem transformation we have in mind involves two steps:

(a) *Elimination of free variables:* Given an unrestricted variable x_j in a problem in general form, we replace it by $x_j^+ - x_j^-$, where x_j^+ and x_j^- are new variables on which we impose the sign constraints $x_j^+ \geq 0$ and $x_j^- \geq 0$. The underlying idea is that any real number can be written as the difference of two nonnegative numbers.

(b) *Elimination of inequality constraints:* Given an inequality constraint of the form

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i,$$