Preface

This book is about constrained optimization. It begins with a thorough treat-
ment of linear programming and proceeds to convex analysis, network flows, integer
programming, quadratic programming, and convex optimization. Along the way,
dynamic programming and the linear complementarity problem are touched on as
well.

The book aims to be a first introduction to the subject. Specific examples and
concrete algorithms precede more abstract topics. Nevertheless, topics covered are
developed in some depth, a large number of numerical examples are worked out
in detail, and many recent topics are included, most notably interior-point methods.
The exercises at the end of each chapter both illustrate the theory and, in some cases, extend it.

Prerequisites. The book is divided into four parts. The first two parts assume
a background only in linear algebra. For the last two parts, some knowledge of
multivariate calculus is necessary. In particular, the student should know how to use
Lagrange multipliers to solve simple calculus problems in 2 and 3 dimensions.

Associated software. It is good to be able to solve small problems by hand,
but the problems one encounters in practice are large, requiring a computer for their
solution. Therefore, to fully appreciate the subject, one needs to solve large (prac-
tical) problems on a computer. An important feature of this book is that it comes
with software implementing the major algorithms described herein. At the time of
writing, software for the following five algorithms is available:

- The two-phase simplex method as shown in Figure 6.1.
- The self-dual simplex method as shown in Figure 7.1.
- The path-following method as shown in Figure 18.1.
- The homogeneous self-dual method as shown in Figure 22.1.
- The long-step homogeneous self-dual method as described in Exercise
  22.4.

The programs that implement these algorithms are written in C and can be
easily compiled on most hardware platforms. Students/instructors are encouraged
to install and compile these programs on their local hardware. Great pains have
been taken to make the source code for these programs readable (see Appendix A).
In particular, the names of the variables in the programs are consistent with the
notation of this book.
There are two ways to run these programs. The first is to prepare the input in a standard computer-file format, called MPS format, and to run the program using such a file as input. The advantage of this input format is that there is an archive of problems stored in this format, called the NETLIB suite, that one can download and use immediately (a link to the NETLIB suite can be found at the web site mentioned below). But, this format is somewhat archaic and, in particular, it is not easy to create these files by hand. Therefore, the programs can also be run from within a problem modeling system called AMPL. AMPL allows one to describe mathematical programming problems using an easy to read, yet concise, algebraic notation. To run the programs within AMPL, one simply tells AMPL the name of the solver-program before asking that a problem be solved. The text that describes AMPL, Fourer et al. (1993) makes an excellent companion to this book. It includes a discussion of many practical linear programming problems. It also has lots of exercises to hone the modeling skills of the student.

Several interesting computer projects can be suggested. Here are a few suggestions regarding the simplex codes:

- Incorporate the partial pricing strategy (see Section 8.7) into the two-phase simplex method and compare it with full pricing.
- Incorporate the steepest-edge pivot rule (see Section 8.8) into the two-phase simplex method and compare it with the largest-coefficient rule.
- Modify the code for either variant of the simplex method so that it can treat bounds and ranges implicitly (see Chapter 9), and compare the performance with the explicit treatment of the supplied codes.
- Implement a “warm-start” capability so that the sensitivity analyses discussed in Chapter 7 can be done.
- Extend the simplex codes to be able to handle integer programming problems using the branch-and-bound method described in Chapter 23.

As for the interior-point codes, one could try some of the following projects:

- Modify the code for the path-following algorithm so that it implements the affine-scaling method (see Chapter 21), and then compare the two methods.
- Modify the code for the path-following method so that it can treat bounds and ranges implicitly (see Section 20.3), and compare the performance against the explicit treatment in the given code.
- Modify the code for the path-following method to implement the higher-order method described in Exercise 18.5. Compare.
- Extend the path-following code to solve quadratic programming problems using the algorithm shown in Figure 24.3.
- Further extend the code so that it can solve convex optimization problems using the algorithm shown in Figure 25.2.

And, perhaps the most interesting project of all:

- Compare the simplex codes against the interior-point code and decide for yourself which algorithm is better on specific families of problems.
The software implementing the various algorithms was developed using consistent data structures and so making fair comparisons should be straightforward. The software can be downloaded from the following web site:

http://www.princeton.edu/~rvdb/LPbook/

If, in the future, further codes relating to this text are developed (for example, a self-dual network simplex code), they will be made available through this web site.

**Features.** Here are some other features that distinguish this book from others:

- The development of the simplex method leads to Dantzig’s parametric self-dual method. A randomized variant of this method is shown to be immune to the travails of degeneracy.
- The book gives a balanced treatment to both the traditional simplex method and the newer interior-point methods. The notation and analysis is developed to be consistent across the methods. As a result, the self-dual simplex method emerges as the variant of the simplex method with most connections to interior-point methods.
- From the beginning and consistently throughout the book, linear programming problems are formulated in *symmetric form*. By highlighting symmetry throughout, it is hoped that the reader will more fully understand and appreciate duality theory.
- By slightly changing the right-hand side in the Klee–Minty problem, we are able to write down an explicit dictionary for each vertex of the Klee–Minty problem and thereby uncover (as a homework problem) a simple, elegant argument why the Klee-Minty problem requires \(2^n - 1\) pivots to solve.
- The chapter on regression includes an analysis of the expected number of pivots required by the self-dual variant of the simplex method. This analysis is supported by an empirical study.
- There is an extensive treatment of modern interior-point methods, including the primal–dual method, the affine-scaling method, and the self-dual path-following method.
- In addition to the traditional applications, which come mostly from business and economics, the book features other important applications such as the optimal design of truss-like structures and \(L^1\)-regression.

**Exercises on the Web.** There is always a need for fresh exercises. Hence, I have created and plan to maintain a growing archive of exercises specifically created for use in conjunction with this book. This archive is accessible from the book’s web site:

http://www.princeton.edu/~rvdb/LPbook/

The problems in the archive are arranged according to the chapters of this book and use notation consistent with that developed herein.

**Advice on solving the exercises.** Some problems are routine while others are fairly challenging. Answers to some of the problems are given at the back of the book.
In general, the advice given to me by Leonard Gross (when I was a student) should help even on the hard problems: *follow your nose*.

**Audience.** This book evolved from lecture notes developed for my introductory graduate course in linear programming as well as my upper-level undergraduate course. A reasonable undergraduate syllabus would cover essentially all of Part 1 (Simplex Method and Duality), the first two chapters of Part 2 (Network Flows and Applications), and the first chapter of Part 4 (Integer Programming). At the graduate level, the syllabus should depend on the preparation of the students. For a well-prepared class, one could cover the material in Parts 1 and 2 fairly quickly and then spend more time on Parts 3 (Interior-Point Methods) and 4 (Extensions).

**Dependencies.** In general, Parts 2 and 3 are completely independent of each other. Both depend, however, on the material in Part 1. The first chapter in Part 4 (Integer Programming) depends only on material from Part 1, whereas the remaining chapters build on Part 3 material.

**Acknowledgments.** My interest in linear programming was sparked by Robert Garfinkel when we shared an office at Bell Labs. I would like to thank him for his constant encouragement, advice, and support. This book benefited greatly from the thoughtful comments and suggestions of David Bernstein and Michael Todd. I would also like to thank the following colleagues for their help: Ronny Ben-Tal, Leslie Hall, Yoshi Ikura, Victor Klee, Irvin Lustig, Avi Mandelbaum, Marc Meke- ton, Narcis Nabona, James Orlin, Andrzej Ruszczynski, and Henry Wolkowicz. I would like to thank Gary Folven at Kluwer and Fred Hillier, the series editor, for encouraging me to undertake this project. I would like to thank my students for finding many typos and occasionally more serious errors: John Gilmartin, Jacinta Warnie, Stephen Woolbert, Lucia Wu, and Bing Yang. My thanks to Erhan Çınlar for the many times he offered advice on questions of style. I hope this book reflects positively on his advice. Finally, I would like to acknowledge the support of the National Science Foundation and the Air Force Office of Scientific Research for supporting me while writing this book. In a time of declining resources, I am especially grateful for their support.

Princeton, NJ, USA

Robert J. Vanderbei
Linear Programming
Foundations and Extensions
Vanderbei, R.J.
2014, XXII, 414 p. 86 illus., Hardcover
ISBN: 978-1-4614-7629-0