
PRACTICAL OPTIMIZATION: A GENTLE INTRODUCTION

John W. Chinneck
Systems and Computer Engineering
Carleton University
Ottawa, Canada

Available online at
www.sce.carleton.ca/faculty/chinneck/po.html.
This version as at November 6, 2012.

TABLE OF CONTENTS

Chapter 1: Introduction. An introduction to the process of optimization and an overview of the major topics covered in the book.

Chapter 2: Introduction to Linear Programming. The basic notions of linear programming and the simplex method. The simplex method is the easiest way to provide a beginner with a solid understanding of linear programming.

Chapter 3: Towards the Simplex Method for Efficient Solution of Linear Programs. Cornerpoints and bases. Moving to improved solutions.

Chapter 4: The Mechanics of the Simplex Method. The tableau representation as a way of illustrating the process of the simplex method. Special cases such as degeneracy and unboundedness.

Chapter 5: Solving General Linear Programs. Solving non-standard linear programs. Phase 1 LPs. Feasibility and infeasibility. Unrestricted variables.

Chapter 6: Sensitivity Analysis. Simple computer-based sensitivity analysis.

Chapter 7: Linear Programming in Practice. Mention of other solution methods such as revised simplex method and interior point methods. Mention of advanced techniques used in practice such as advanced and crash start methods, infeasibility analysis, and modelling systems.

Chapter 8: An Introduction to Networks. Some basic network concepts. The shortest route problem. The minimum spanning tree problem.

Chapter 9: Maximum Flow and Minimum Cut. The maximum flow and minimum cut problems in networks.

Chapter 10: Network Flow Programming. A surprising range of problems can be solved using minimum cost network flow programming, including shortest route, maximum flow and minimum cut, etc. Variations such as generalized and processing networks are also briefly introduced.

Chapter 11: PERT for Project Planning and Scheduling. PERT is a network-based aid for project planning and scheduling. Many optimization problems involve some aspect of the timing of activities that may run sequentially or in parallel, or the timing of resource use. PERT diagrams help you to understand and formulate such problems.

Chapter 12: Integer/Discrete Programming via Branch and Bound. Branch and bound is the basic workhorse method for solving many kinds of problems that have variables that can take on only integer, binary, or discrete values.

Chapter 13: Binary and Mixed-Integer Programming. These are specialized versions of branch and bound. A binary program has only binary variables (0 or 1 only). A mixed-integer program looks like a linear program, except that some or all of the variables are integer-valued (or binary-valued), while others might be real-valued.

Chapter 14: Heuristics for Discrete Search: Genetic Algorithms and Simulated Annealing. Some problems are just too big for branch and bound, in which case you must abandon the guarantee of finding the optimum solution and instead opt for heuristic methods which can only guarantee to do fairly well most of the time. Genetic Algorithms and Simulated Annealing are two popular heuristic methods for use on very large problems.

Chapter 15: Dynamic Programming. This optimization technique builds towards a solution by first solving a small part of the whole problem, and then gradually incrementing the size in a series of stages until the whole problem is solved. Efficiency results from combining the local solution for a stage with the optimum found for a previous stage. We look at the simplest deterministic discrete cases.

Chapter 16: Introduction to Nonlinear Programming (NLP). NLP is a lot harder than linear programming. We start by looking at the reasons for this. Next we look at the simplest method for solving the simplest type of NLP: unconstrained problems that consist only of a nonlinear objective function. The method of steepest ascent/descent is described.

See the online algorithm animations linked from www.sce.carleton.ca/faculty/chinneck/po.html.