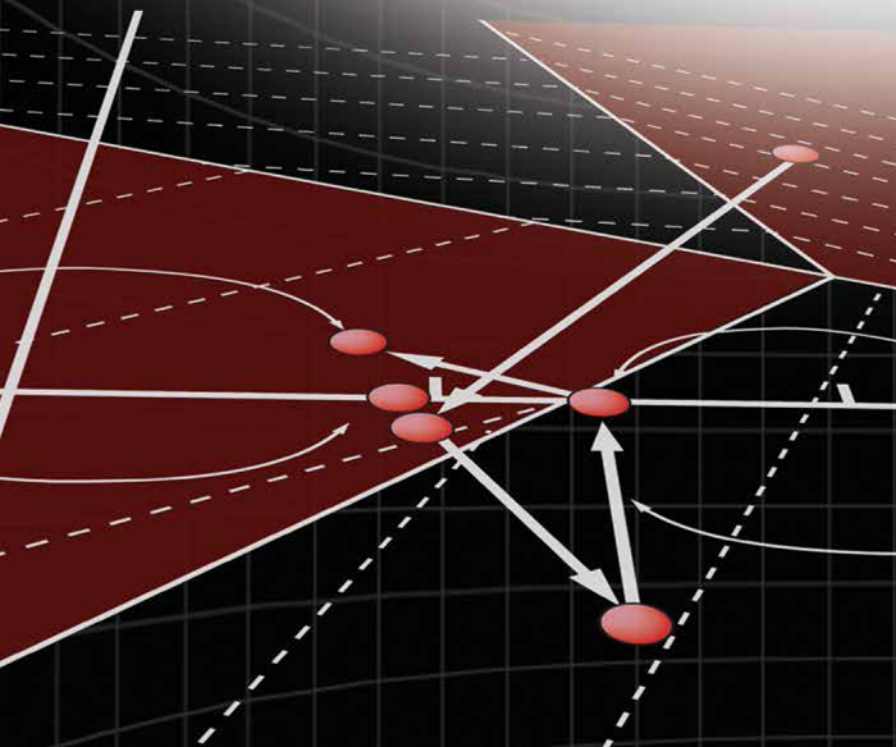


RONALD L. RARDIN

OPTIMIZATION IN OPERATIONS RESEARCH

SECOND EDITION



PRIMERS

| | | |
|---|--|-----|
| 1 | Vectors | 89 |
| 2 | Derivatives and Partial Derivatives | 110 |
| 3 | Extreme Points and Directions of Polyhedral Sets | 203 |
| 4 | Matrices, Matrix Arithmetic, and Transposes | 213 |
| 5 | Simultaneous Equations, Singularity, and Bases | 223 |
| 6 | Identity and Inverse Matrices | 261 |
| 7 | Second Derivatives and Hessian Matrices | 938 |
| 8 | Positive and Negative (Semi) Definite Matrices | 945 |

ALGORITHMS

| | | |
|----|--|-----|
| 3A | Continuous Improving Search | 106 |
| 3B | Two-Phase Improving Search | 129 |
| 5A | Rudimentary Simplex Search for Linear Programs | 235 |
| 5B | Two-Phase Simplex Search | 247 |
| 5C | Revised Simplex Search for Linear Programs | 271 |
| 5D | Lower- and Upper-Bounded Revised Simplex | 278 |
| 6A | Dual Simplex Search for Linear Programs | 363 |
| 6B | Primal-Dual Simplex Search for Linear Programs | 369 |
| 7A | Affine Scaling Search for Linear Programs | 407 |
| 7B | Newton Step Barrier Search for Linear Programs | 419 |
| 7C | Primal-Dual Interior-Point LP Search | 424 |
| 9A | One to All (No Negative Dicycles); Bellman–Ford Shortest Paths | 496 |
| 9B | All-to-All (No Negative Dicycles); Floyd–Warshall Shortest Paths | 502 |
| 9C | One to All (Nonnegative Costs); Dijkstra Shortest Paths | 509 |
| 9D | Shortest One to All Paths (Acyclic Digraph) Shortest Paths | 518 |

| | | |
|-----|---|------|
| 9E | CPM Early Start Scheduling | 525 |
| 10A | Rudimentary Cycle Direction Network Search | 582 |
| 10B | Cycle Cancelling for Network Flows | 587 |
| 10C | Network Simplex Search | 599 |
| 10D | Hungarian Algorithm for Linear Assignment | 612 |
| 10E | Maxflow-Mincut Search | 622 |
| 10F | Greedy Search for a Min/Max Spanning Tree | 634 |
| 12A | LP-Based Branch and Bound (0-1 ILPs) | 761 |
| 12B | Branch and Cut (0-1 ILP's) | 779 |
| 13A | Delayed Column Generation | 816 |
| 13B | Branch and Price Search (0-1 ILPs) | 820 |
| 13C | Subgradient Lagrangian Search | 835 |
| 13D | Dantzig–Wolfe Decomposition | 841 |
| 13E | Benders Decomposition | 846 |
| 15A | Rudimentary Constructive Search | 880 |
| 15B | Discrete Improving Search | 887 |
| 15C | Tabu Search | 895 |
| 15D | Simulated Annealing Search | 898 |
| 15E | Genetic Algorithm Search | 904 |
| 16A | Golden Section Search | 927 |
| 16B | Three-Point Pattern | 931 |
| 16C | Quadratic Fit Search | 934 |
| 16D | Gradient Search | 955 |
| 16E | Newton's Method | 962 |
| 16F | BFGS Quasi-Newton Search | 968 |
| 16G | Nelder–Mead Derivative-Free Search | 974 |
| 17A | Sequential Unconstrained Penalty Technique (SUMT) | 1034 |
| 17B | Sequential Unconstrained Barrier Technique | 1038 |
| 17C | Reduced Gradient Search | 1048 |
| 17D | Active Set Method for Quadratic Programs | 1059 |
| 17E | Sequential Quadratic Programming (SQP) | 1063 |

Optimization in Operations Research

This page intentionally left blank

Optimization in Operations Research

SECOND EDITION

RONALD L. RARDIN

University of Arkansas

PEARSON

Boston Columbus Indianapolis Hoboken New York San Francisco Amsterdam
Cape Town Dubai London Madrid Milan Munich Paris Montréal Toronto
Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Vice President and Editorial
Director, ECS: Marcia J. Horton
Executive Editor: Holly Stark
Editorial Assistant: Amanda Brands
Field Marketing Manager: Demetrius Hall
Senior Product Marketing Manager:
Bram van Kempen
Marketing Assistant: Jon Bryant
Senior Managing Editor: Scott Disanno
Program Manager: Erin Ault
Production Project Manager: Greg Dulles

Director of Operations: Nick Sklitsis
Operations Specialist: Maura Zaldivar-Garcia
Cover Designer: Black Horse Designs
Manager, Rights and Permissions:
Rachel Youdelman
Associate Project Manager, Rights and
Permissions: William Opaluch
Composition: Integra Software Services, Inc.
Printer/Binder: RR Donnelley/Crawfordsville
Cover Printer: Phoenix Color/Hagerstown
Typeface: 10/12 Times Ten LT Std

Copyright © 2017, 1998 by Pearson Higher Education, Inc., Hoboken, NJ 07030.

All rights reserved. Manufactured in the United States of America. This publication is protected by Copyright and permissions should be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise. For information regarding permissions, request forms and the appropriate contacts within the Pearson Education Global Rights & Permissions department, please visit www.pearsoned.com/permissions/.

Many of the designations by manufacturers and seller to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed in initial caps or all caps.

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages with, or arising out of, the furnishing, performance, or use of these programs.

Library of Congress Cataloging-in-Publication Data

Rardin, Ronald L.

Optimization in operations research / Ronald L. Rardin, Purdue University.—Second edition.
pages cm

Includes bibliographical references and index.

ISBN 978-0-13-438455-9—ISBN 0-13-438455-5

1. Operations research. 2. Mathematical optimization. 3. Programming (Mathematics) I. Title.

T57.7.R37 2016

519.7'2—dc23

2015019627

10 9 8 7 6 5 4 3 2 1

PEARSON

ISBN 10: 0-13-438455-5

ISBN 13: 978-0-13-438455-9

Contents

PREFACE *xxix*

ABOUT THE AUTHOR *xxxii*

CHAPTER 1 *PROBLEM SOLVING WITH MATHEMATICAL MODELS* **1**

- 1.1 OR Application Stories 1
 - 1.2 Optimization and the Operations Research Process 3
 - Decisions, Constraints, and Objectives* 4
 - Optimization and Mathematical Programming* 4
 - Constant-Rate Demand Assumption* 5
 - Back of Envelope Analysis* 5
 - Constant-Rate Demand Model* 7
 - Feasible and Optimal Solutions* 7
 - 1.3 System Boundaries, Sensitivity Analysis, Tractability, and Validity 9
 - EOQ Under Constant-Rate Demand* 9
 - System Boundaries and Sensitivity Analysis* 10
 - Closed-Form Solutions* 11
 - Tractability versus Validity* 11
 - 1.4 Descriptive Models and Simulation 12
 - Simulation over MM's History* 12
 - Simulation Model Validity* 12
 - Descriptive versus Prescriptive Models* 14
 - 1.5 Numerical Search and Exact Versus Heuristic Solutions 14
 - Numerical Search* 14
 - A Different Start* 15
 - Exact versus Heuristic Optimization* 16
 - 1.6 Deterministic Versus Stochastic Models 16
 - Random Variables and Realizations* 17
 - Stochastic Simulation* 17
 - Tradeoffs between Deterministic and Stochastic Models* 19
 - 1.7 Perspectives 19
 - Other Issues* 20
 - The Rest of This Book* 20
- Exercises** 20

CHAPTER 2 DETERMINISTIC OPTIMIZATION MODELS IN OPERATIONS RESEARCH 23

- 2.1 Decision Variables, Constraints, and Objective Functions 23
 - Decision Variables* 24
 - Variable-Type Constraints* 24
 - Main Constraints* 25
 - Objective Functions* 25
 - Standard Model* 26
- 2.2 Graphic Solution and Optimization Outcomes 27
 - Graphic Solution* 27
 - Feasible Sets* 27
 - Graphing Constraints and Feasible Sets* 27
 - Graphing Objective Functions* 30
 - Optimal Solutions* 33
 - Optimal Values* 34
 - Unique versus Alternative Optimal Solutions* 35
 - Infeasible Models* 36
 - Unbounded Models* 38
- 2.3 Large-Scale Optimization Models and Indexing 40
 - Indexing* 40
 - Indexed Decision Variables* 41
 - Indexed Symbolic Parameters* 42
 - Objective Functions* 43
 - Indexed Families of Constraints* 43
 - Pi Hybrids Application Model* 45
 - How Models Become Large* 46
- 2.4 Linear and Nonlinear Programs 46
 - General Mathematical Programming Format* 46
 - Right-Hand Sides* 47
 - Linear Functions* 48
 - Linear and Nonlinear Programs Defined* 50
 - Two Crude and Pi Hybrids Models are LPs* 51
 - Indexing, Parameters, and Decision Variables for E-mart* 51
 - Nonlinear Response* 51
 - E-mart Application Model* 52
- 2.5 Discrete or Integer Programs 53
 - Indexes and Parameters of the Bethlehem Application* 53
 - Discrete versus Continuous Decision Variables* 53
 - Constraints with Discrete Variables* 55
 - Bethlehem Ingot Mold Application Model* 56
 - Integer and Mixed-Integer Programs* 56
 - Integer Linear versus Integer Nonlinear Programs* 57
 - Indexing, Parameters, and Decision Variables for Purdue Finals Application* 59

| | | |
|-----|---|----|
| | <i>Nonlinear Objective Function</i> | 59 |
| | <i>Purdue Final Exam Scheduling Application Model</i> | 60 |
| 2.6 | Multiobjective Optimization Models | 60 |
| | <i>Multiple Objectives</i> | 61 |
| | <i>Constraints of the DuPage Land Use Application</i> | 62 |
| | <i>DuPage Land Use Application Model</i> | 63 |
| | <i>Conflict among Objectives</i> | 64 |
| 2.7 | Classification Summary | 65 |
| 2.8 | Computer Solution and AMPL | 65 |
| | <i>Solvers versus Modeling Languages</i> | 66 |
| | <i>Indexing, Summations, and Symbolic Parameters</i> | 67 |
| | <i>Nonlinear and Integer Models</i> | 70 |
| | Exercises | 73 |
| | References | 86 |

CHAPTER 3 IMPROVING SEARCH 87

| | | |
|-----|---|-----|
| 3.1 | Improving Search, Local, and Global Optima | 87 |
| | <i>Solutions</i> | 88 |
| | <i>Solutions as Vectors</i> | 88 |
| | <i>Example of an Improving Search</i> | 93 |
| | <i>Neighborhood Perspective</i> | 94 |
| | <i>Local Optima</i> | 95 |
| | <i>Local Optima and Improving Search</i> | 95 |
| | <i>Local versus Global Optima</i> | 95 |
| | <i>Dealing with Local Optima</i> | 97 |
| 3.2 | Search with Improving and Feasible Directions | 98 |
| | <i>Direction-Step Paradigm</i> | 98 |
| | <i>Improving Directions</i> | 100 |
| | <i>Feasible Directions</i> | 102 |
| | <i>Step Size: How Far?</i> | 104 |
| | <i>Search of the DClub Example</i> | 105 |
| | <i>When Improving Search Stops</i> | 107 |
| | <i>Detecting Unboundedness</i> | 108 |
| 3.3 | Algebraic Conditions for Improving and Feasible Directions | 109 |
| | <i>Gradients</i> | 109 |
| | <i>Gradient Conditions for Improving Directions</i> | 112 |
| | <i>Objective Function Gradients as Move Directions</i> | 114 |
| | <i>Active Constraints and Feasible Directions</i> | 115 |
| | <i>Linear Constraints</i> | 117 |
| | <i>Conditions for Feasible Directions with Linear Constraints</i> | 118 |

| | | |
|-----|---|------------|
| 3.4 | Tractable Convex and Linear Cases | 120 |
| | <i>Special Tractability of Linear Objective Functions</i> | 120 |
| | <i>Constraints and Local Optima</i> | 121 |
| | <i>Convex Feasible Sets</i> | 121 |
| | <i>Algebraic Description of Line Segments</i> | 123 |
| | <i>Convenience of Convex Feasible Sets for Improving Search</i> | 124 |
| | <i>Global Optimality of Linear Objectives over Convex Feasible Sets</i> | 125 |
| | <i>Convexity of Linearly Constrained Feasible Sets</i> | 126 |
| | <i>Global Optimality of Improving Search for Linear Programs</i> | 127 |
| | <i>Blocking Constraints in Linear Programs</i> | 127 |
| 3.5 | Searching for Starting Feasible Solutions | 129 |
| | <i>Two-Phase Method</i> | 129 |
| | <i>Two Crude Model Application Revisited</i> | 129 |
| | <i>Artificial Variables</i> | 130 |
| | <i>Phase I Models</i> | 130 |
| | <i>Starting Artificial Solution</i> | 131 |
| | <i>Phase I Outcomes</i> | 132 |
| | <i>Concluding Infeasibility from Phase I</i> | 133 |
| | <i>Big-M Method</i> | 135 |
| | <i>Big-M Outcomes</i> | 136 |
| | Exercises | 138 |
| | References | 141 |

CHAPTER 4 *LINEAR PROGRAMMING MODELS* 143

| | | |
|-----|--|-----|
| 4.1 | Allocation Models | 144 |
| | <i>Allocation Decision Variables</i> | 145 |
| | <i>Forest Service Allocation Model</i> | 145 |
| 4.2 | Blending Models | 147 |
| | <i>Ingredient Decision Variables</i> | 148 |
| | <i>Composition Constraints</i> | 148 |
| | <i>Swedish Steel Example Model</i> | 150 |
| | <i>Ratio Constraints</i> | 150 |
| 4.3 | Operations Planning Models | 152 |
| | <i>Tubular Products Operations Planning Model</i> | 153 |
| | <i>CFPL Decision Variables</i> | 156 |
| | <i>Continuous Variables for Integer Quantities</i> | 157 |
| | <i>CFPL Objective Function</i> | 157 |
| | <i>CFPL Constraints</i> | 158 |
| | <i>Balance Constraints</i> | 158 |
| | <i>CFPL Application Model</i> | 160 |

- 4.4 Shift Scheduling and Staff Planning Models 162
 - ONB Decision Variables and Objective Function* 163
 - ONB Constraints* 164
 - Covering Constraints* 164
 - ONB Shift Scheduling Application Model* 165
- 4.5 Time-Phased Models 166
 - Time-Phased Decision Variables* 167
 - Time-Phased Balance Constraints* 168
 - IFS Cash Flow Model* 169
 - Time Horizons* 170
- 4.6 Models with Linearizable Nonlinear Objectives 171
 - Maxisum Highway Patrol Application Model* 172
 - Minimax and Maximin Objective Functions* 173
 - Nonlinear Maximin Highway Patrol Application Model* 173
 - Linearizing Minimax and Maximin Objective Functions* 173
 - Linearized Maximin Highway Patrol Example Model* 174
 - Nonlinear VP Location Model* 175
 - Min Deviation Objective Functions* 176
 - Linearizing Min Deviation Objective Functions* 176
 - Linearized VP Location Model* 177
- 4.7 Stochastic Programming 179
 - Deterministic Model of QA Example* 180
 - Stochastic Programming with Recourse* 181
 - Stochastic Programming Modeling of the QA Application* 182
 - Extensive Form versus Large-Scale Techniques* 184
- Exercises 185**
- References 200**

CHAPTER 5 SIMPLEX SEARCH FOR LINEAR PROGRAMMING 201

- 5.1 LP Optimal Solutions and Standard Form 201
 - Global Optima in Linear Programs* 203
 - Interior, Boundary, and Extreme Points* 204
 - Optimal Points in Linear Programs* 207
 - LP Standard Form* 208
 - Converting Inequalities to Nonnegativities with Slack Variables* 209
 - Converting Nonpositive and Unrestricted Variables to Nonnegative* 211
 - Standard Notation for LPs* 213
- 5.2 Extreme-Point Search and Basic Solutions 216
 - Determining Extreme Points with Active Constraints* 216
 - Adjacent Extreme Points and Edges* 216

| | | |
|-----|---|-----|
| | <i>Basic Solutions</i> | 219 |
| | <i>Existence of Basic Solutions</i> | 221 |
| | <i>Basic Feasible Solutions and Extreme Points</i> | 225 |
| 5.3 | The Simplex Algorithm | 227 |
| | <i>Standard Display</i> | 227 |
| | <i>Initial Basic Solution</i> | 228 |
| | <i>Simplex Directions</i> | 228 |
| | <i>Improving Simplex Directions and Reduced Costs</i> | 231 |
| | <i>Step Size and the Minimum Ratio Rule</i> | 232 |
| | <i>Updating the Basis</i> | 234 |
| | <i>Rudimentary Simplex Algorithm</i> | 235 |
| | <i>Rudimentary Simplex Solution of Top Brass Example</i> | 236 |
| | <i>Stopping and Global Optimality</i> | 236 |
| | <i>Extreme-Point or Extreme-Direction</i> | 238 |
| 5.4 | Dictionary and Tableau Representations of Simplex | 238 |
| | <i>Simplex Dictionaries</i> | 239 |
| | <i>Simplex Tableaux</i> | 241 |
| | <i>Simplex Algorithm with Dictionaries or Tableaux</i> | 242 |
| | <i>Correspondence to the Improving Search Paradigm</i> | 242 |
| | <i>Comparison of Formats</i> | 243 |
| 5.5 | Two Phase Simplex | 243 |
| | <i>Starting Basis in the Two Phase Simplex</i> | 245 |
| | <i>Three Possible Outcomes for Linear Programs</i> | 247 |
| | <i>Clever Clyde Infeasible Case</i> | 247 |
| | <i>Clever Clyde Optimal Case</i> | 250 |
| | <i>Clever Clyde Unbounded Case</i> | 252 |
| 5.6 | Degeneracy and Zero-Length Simplex Steps | 253 |
| | <i>Degenerate Solutions</i> | 253 |
| | <i>Zero-Length Simplex Steps</i> | 255 |
| | <i>Progress through Changing of Bases</i> | 256 |
| 5.7 | Convergence and Cycling with Simplex | 257 |
| | <i>Finite Convergence with Positive Steps</i> | 257 |
| | <i>Degeneracy and Cycling</i> | 258 |
| 5.8 | Doing it Efficiently: Revised Simplex | 260 |
| | <i>Computations with Basis Inverses</i> | 260 |
| | <i>Updating the Representation of B^{-1}</i> | 264 |
| | <i>Basic Variable Sequence in Revised Simplex</i> | 266 |
| | <i>Computing Reduced Costs by Pricing</i> | 267 |
| | <i>Revised Simplex Search of Top Brass Application</i> | 269 |
| 5.9 | Simplex with Simple Upper and Lower Bounds | 272 |
| | <i>Lower- and Upper-Bounded Standard Form</i> | 272 |
| | <i>Basic Solutions with Lower and Upper Bounds</i> | 274 |
| | <i>Unrestricted Variables with No Bounds</i> | 274 |
| | <i>Increasing and Decreasing Nonbasic Variable Values</i> | 275 |
| | <i>Step Size with Increasing and Decreasing Values</i> | 276 |

| | |
|--|------------|
| <i>Case with No Basis Change</i> | 277 |
| <i>Lower- and Upper-Bounded Simplex Algorithm</i> | 277 |
| <i>Lower- and Upper-Bounded Simplex on Top Brass Application</i> | 277 |
| Exercises | 280 |
| References | 285 |

CHAPTER 6 DUALITY, SENSITIVITY, AND OPTIMALITY IN LINEAR PROGRAMMING 287

| | | |
|-----|---|-----|
| 6.1 | Generic Activities Versus Resources Perspective | 288 |
| | <i>Objective Functions as Costs and Benefits</i> | 288 |
| | <i>Choosing a Direction for Inequality Constraints</i> | 288 |
| | <i>Inequalities as Resource Supplies and Demands</i> | 288 |
| | <i>Equality Constraints as Both Supplies and Demands</i> | 289 |
| | <i>Variable-Type Constraints</i> | 290 |
| | <i>Variables as Activities</i> | 290 |
| | <i>LHS Coefficients as Activity Inputs and Outputs</i> | 290 |
| 6.2 | Qualitative Sensitivity to Changes in Model Coefficients | 293 |
| | <i>Relaxing versus Tightening Constraints</i> | 293 |
| | <i>Swedish Steel Application Revisited</i> | 293 |
| | <i>Effects of Changes in Right-Hand Sides</i> | 294 |
| | <i>Effects of Changes in LHS Constraint Coefficients</i> | 296 |
| | <i>Effects of Adding or Dropping Constraints</i> | 297 |
| | <i>Effects of Unmodeled Constraints</i> | 297 |
| | <i>Changing Rates of Constraint Coefficient Impact</i> | 298 |
| | <i>Effects of Objective Function Coefficient Changes</i> | 299 |
| | <i>Changing Rates of Objective Function Coefficient Impact</i> | 301 |
| | <i>Effects of Adding or Dropping Variables</i> | 303 |
| 6.3 | Quantifying Sensitivity to Changes in LP Model Coefficients: A Dual Model | 304 |
| | <i>Primals and Duals Defined</i> | 304 |
| | <i>Dual Variables</i> | 304 |
| | <i>Dual Variable Types</i> | 305 |
| | <i>Two Crude Application Again</i> | 306 |
| | <i>Dual Variables as Implicit Marginal Resource Prices</i> | 307 |
| | <i>Implicit Activity Pricing in Terms of Resources Produced and Consumed</i> | 308 |
| | <i>Main Dual Constraints to Enforce Activity Pricing</i> | 309 |
| | <i>Optimal Value Equality between Primal and Dual</i> | 310 |
| | <i>Primal Complementary Slackness between Primal Constraints and Dual Variable Values</i> | 311 |
| | <i>Dual Complementary Slackness between Dual Constraints and Primal Variable Values</i> | 312 |

| | | |
|-----|--|-----|
| 6.4 | Formulating Linear Programming Duals | 313 |
| | <i>Form of the Dual for Nonnegative Primal Variables</i> | 314 |
| | <i>Duals of LP Models with Nonpositive and Unrestricted Variables</i> | 316 |
| | <i>Dual of the Dual is the Primal</i> | 317 |
| 6.5 | Computer Outputs and What If Changes of Single Parameters | 318 |
| | <i>CFPL Example Primal and Dual</i> | 318 |
| | <i>Constraint Sensitivity Outputs</i> | 320 |
| | <i>Right-Hand-Side Ranges</i> | 322 |
| | <i>Constraint What If's</i> | 324 |
| | <i>Variable Sensitivity Outputs</i> | 326 |
| | <i>Objective Coefficient Ranges</i> | 328 |
| | <i>Variable What If's</i> | 330 |
| | <i>Dropping and Adding Constraint What If's</i> | 332 |
| | <i>Dropping and Adding Variable What If's</i> | 333 |
| 6.6 | Bigger Model Changes, Reoptimization, and Parametric Programming | 335 |
| | <i>Ambiguity at Limits of the RHS and Objective Coefficient Ranges</i> | 335 |
| | <i>Connection between Rate Changes and Degeneracy</i> | 337 |
| | <i>Reoptimization to Make Sensitivity Exact</i> | 338 |
| | <i>Parametric Variation of One Coefficient</i> | 338 |
| | <i>Assessing Effects of Multiple Parameter Changes</i> | 340 |
| | <i>Parametric Multiple-RHS Change</i> | 341 |
| | <i>Parametric Change of Multiple Objective Function Coefficients</i> | 343 |
| 6.7 | Duality and Optimality in Linear Programming | 344 |
| | <i>Dual of the Dual</i> | 345 |
| | <i>Weak Duality between Objective Values</i> | 345 |
| | <i>Unbounded and Infeasible Cases</i> | 347 |
| | <i>Complementary Slackness and Optimality</i> | 349 |
| | <i>Strong Duality and Karush-Kuhn-Tucker (KKT) Optimality Conditions for Linear Programs</i> | 351 |
| | <i>Models in Standard Form</i> | 352 |
| | <i>Standard Form LPs in Partitioned Basic Format</i> | 354 |
| | <i>Basic Solutions in Partitioned Form</i> | 355 |
| | <i>Complementary Dual Basic Solutions</i> | 355 |
| | <i>Primal Simplex Optimality and Necessity of KKT Conditions</i> | 357 |
| 6.8 | Dual Simplex Search | 359 |
| | <i>Choosing an Improving Direction</i> | 361 |
| | <i>Determining a Dual Step Size to Retain Dual Feasibility</i> | 361 |
| | <i>Changing the Primal Solution and Basis Update</i> | 362 |

- 6.9 Primal-Dual Simplex Search 365
 Choosing an Improving Dual Direction 367
 Determining a Dual Step Size 368
 Exercises 371
 References 384

CHAPTER 7 *INTERIOR POINT METHODS FOR LINEAR PROGRAMMING* 385

- 7.1 Searching through the Interior 385
 Interior Points 386
 Objective as a Move Direction 386
 Boundary Strategy of Interior Point Methods 387
 Interior in LP Standard Form 389
 Projecting to Deal with Equality Constraints 390
 Improvement with Projected Directions 394
- 7.2 Scaling with the Current Solution 396
 Affine Scaling 396
 Diagonal Matrix Formalization of Affine Scaling 396
 Affine-Scaled Standard Form 399
 Projecting on Affine-Scaled Equality Constraints 401
 Computational Effort in Interior Point Computations 402
- 7.3 Affine Scaling Search 402
 Affine Scaling Move Directions 402
 Feasibility and Improvement of Affine Scaling Directions 404
 Affine Scaling Step Size 404
 Termination in Affine Scaling Search 407
 Affine Scaling Search of the Frannie's Firewood Application 408
- 7.4 Log Barrier Methods for Interior Point Search 408
 Barrier Objective Functions 408
 Problems with Gradient Directions 411
 Newton Steps for Barrier Search 412
 Newton Step Barrier Search Step Sizes 415
 Impact of the Barrier Multiplier μ 417
 Barrier Algorithm Multiplier Strategy 418
 Newton Step Barrier Algorithm 418
 Newton Barrier Solution of Frannie's Firewood Application 419
- 7.5 Primal-Dual Interior-Point Search 421
 KKT Optimality Conditions 421
 Strategy of Primal-Dual Interior-Point Search 422
 Feasible Move Directions 422
 Management of Complementary Slackness 423
 Step Size 423
 Solving the Conditions for Move Directions 423

| | | |
|-----|---|------------|
| 7.6 | Complexity of Linear Programming Search | 428 |
| | <i>Length of Input for LP Instances</i> | 428 |
| | <i>Complexity of Simplex Algorithms for LP</i> | 429 |
| | <i>Complexity of Interior-Point Algorithms for LP</i> | 430 |
| | Exercises | 430 |
| | References | 435 |

CHAPTER 8 *MULTIOBJECTIVE OPTIMIZATION AND GOAL PROGRAMMING* 437

| | | |
|-----|--|-----|
| 8.1 | Multiobjective Optimization Models | 437 |
| | <i>Bank Three Example Objectives</i> | 438 |
| | <i>Bank Three Example Model</i> | 439 |
| | <i>Dynamometer Ring Design Model</i> | 440 |
| | <i>Hazardous Waste Disposal Model</i> | 442 |
| 8.2 | Efficient Points and the Efficient Frontier | 443 |
| | <i>Efficient Points</i> | 443 |
| | <i>Identifying Efficient Points Graphically</i> | 444 |
| | <i>Efficient Frontier</i> | 445 |
| | <i>Plots in Objective Value Space</i> | 446 |
| | <i>Constructing the Efficient Frontier</i> | 446 |
| 8.3 | Preemptive Optimization and Weighted Sums of Objectives | 448 |
| | <i>Preemptive Optimization</i> | 448 |
| | <i>Preemptive Optimization of the Bank Three Application</i> | 448 |
| | <i>Preemptive Optimization and Efficient Points</i> | 451 |
| | <i>Preemptive Optimization and Alternative Optima</i> | 451 |
| | <i>Weighted Sums of Objectives</i> | 451 |
| | <i>Weighted-Sum Optimization of the Hazardous Waste Application</i> | 452 |
| | <i>Weighted-Sum Optimization and Efficient Points</i> | 453 |
| 8.4 | Goal Programming | 454 |
| | <i>Goal or Target Levels</i> | 454 |
| | <i>Goal Form of Bank Three Application</i> | 454 |
| | <i>Soft Constraints</i> | 455 |
| | <i>Deficiency Variables</i> | 455 |
| | <i>Expressing Soft Constraints in Mathematical Programs</i> | 456 |
| | <i>Goal Program Objective Function: Minimizing (Weighted) Deficiency</i> | 457 |
| | <i>Goal Linear Program Model of the Bank Three Application</i> | 457 |
| | <i>Alternative Deficiency Weights in the Objective</i> | 458 |
| | <i>Preemptive Goal Programming</i> | 459 |
| | <i>Preemptive Goal Programming of the Bank Three Application</i> | 459 |

Preemptive Goal Programming by Weighting the Objective 461
Practical Advantage of Goal Programming in Multiobjective Problems 461
Goal Programming and Efficient Points 462
Modified Goal Program Formulation to Assure Efficient Points 464
Exercises 465
References 475

CHAPTER 9 *SHORTEST PATHS AND DISCRETE DYNAMIC PROGRAMMING* 477

9.1 Shortest Path Models 477
Nodes, Arcs, Edges, and Graphs 478
Paths 479
Shortest Path Problems 481
Classification of Shortest Path Models 481
Undirected and Directed Graphs (Digraphs) 482
Two Ring Application Model 485

9.2 Dynamic Programming Approach to Shortest Paths 485
Families of Shortest Path Models 485
Functional Notation 486
Optimal Paths and Subpaths 487
Negative Dicycles Exception 488
Principle of Optimality 489
Functional Equations 489
Functional Equations for One Node to All Others 489
Sufficiency of Functional Equations in the One to All Case 490
Functional Equations for All Nodes to All Others 493
Solving Shortest Path Problems by Linear Programming 494

9.3 Shortest Paths from One Node to All Others:
 Bellman–Ford 494
 Solving the Functional Equations 495
 Repeated Evaluation Algorithm: Bellman–Ford 495
 Bellman–Ford Solution of the Two Ring Circus Application 496
 Justification of the Bellman–Ford Algorithm 498
 Recovering Optimal Paths 499
 Encountering Negative Dicycles with Bellman–Ford 500

9.4 Shortest Paths from All Nodes to All Others:
 Floyd–Warshall 501
 Floyd–Warshall Algorithm 501
 Floyd–Warshall Solution of the Littleville Application 503
 Recovering Optimal Paths 507
 Detecting Negative Dicycles with Floyd–Warshall 507

| | | |
|------|--|------------|
| 9.5 | Shortest Path from One Node to All Others with Costs | |
| | Nonnegative: Dijkstra | 509 |
| | <i>Permanently and Temporarily Labeled Nodes</i> | 509 |
| | <i>Least Temporary Criterion for Next Permanent Node</i> | 510 |
| | <i>Dijkstra Algorithm Solution of the Texas Transfer</i> | |
| | Application | 510 |
| | <i>Recovering Paths</i> | 514 |
| | <i>Justification of the Dijkstra Algorithm</i> | 514 |
| 9.6 | Shortest Paths from One Node to All Others in Acyclic Digraphs | 515 |
| | <i>Acyclic Digraphs</i> | 515 |
| | <i>Shortest Path Algorithm for Acyclic Digraphs</i> | 518 |
| | <i>Acyclic Shortest Path Example</i> | 518 |
| | <i>Longest Path Problems and Acyclic Digraphs</i> | 519 |
| 9.7 | CPM Project Scheduling and Longest Paths | 520 |
| | <i>Project Management</i> | 520 |
| | <i>CPM Project Networks</i> | 521 |
| | <i>CPM Schedules and Longest Paths</i> | 523 |
| | <i>Critical Paths</i> | 523 |
| | <i>Computing an Early Start Schedule for the We Build Construction</i> | |
| | Application | 524 |
| | <i>Late Start Schedules and Schedule Slack</i> | 526 |
| | <i>Acyclic Character of Project Networks</i> | 527 |
| 9.8 | Discrete Dynamic Programming Models | 528 |
| | <i>Sequential Decision Problems</i> | 528 |
| | <i>States in Dynamic Programming</i> | 529 |
| | <i>Digraphs for Dynamic Programs</i> | 530 |
| | <i>Dynamic Programming Solutions as an Optimal Path</i> | 531 |
| | <i>Dynamic Programming Functional Equations</i> | 532 |
| | <i>Dynamic Programming Models with Both Stages and States</i> | 532 |
| | <i>Dynamic Programming Modeling of the President's Library</i> | |
| | Application | 534 |
| | <i>Backward Solution of Dynamic Programs</i> | 534 |
| | <i>Multiple Problem Solutions Obtained Simultaneously</i> | 537 |
| 9.9 | Solving Integer Programs with Dynamic Programming | 537 |
| | <i>Dynamic Programming Modeling of Electoral Vote</i> | |
| | Knapsack | 538 |
| 9.10 | Markov Decision Processes | 541 |
| | <i>Elements of MDP Models</i> | 541 |
| | <i>Solution of the Breast Cancer MDP</i> | 545 |
| | Exercises | 546 |
| | References | 556 |

CHAPTER 10 NETWORK FLOWS AND GRAPHS 557

- 10.1 Graphs, Networks, and Flows 557
 - Digraphs, Nodes, and Arcs* 557
 - OOI Application Network* 558
 - Minimum Cost Flow Models* 559
 - Sources, Sinks, and Transshipment Nodes* 560
 - OOI Application Model* 560
 - Total Supply = Total Demand* 562
 - Starting Feasible Solutions* 563
 - Artificial Network Flow Model* 563
 - Time-Expanded Flow Models and Networks* 565
 - Time-Expanded Modeling of Agrico Application* 567
 - Node–Arc Incidence Matrices and Matrix Standard Form* 568
- 10.2 Cycle Directions for Network Flow Search 570
 - Chains, Paths, Cycles, and Dicycles* 570
 - Cycle Directions* 571
 - Maintaining Flow Balance with Cycle Directions* 573
 - Feasible Cycle Directions* 574
 - Improving Cycle Directions* 576
 - Step Size with Cycle Directions* 577
 - Sufficiency of Cycle Directions* 578
 - Rudimentary Cycle Direction Search for Network Flows* 580
 - Rudimentary Cycle Direction Search of the OOI Application* 580
- 10.3 Cycle Cancelling Algorithms for Optimal Flows 582
 - Residual Digraphs* 582
 - Feasible Cycle Directions and Dicycles of Residual Digraphs* 584
 - Improving Feasible Cycle Directions and Negative Dicycles of Residual Digraphs* 585
 - Using Shortest Path Algorithms to Find Cycle Directions* 586
 - Cycle Cancelling Solution of the OOI Application* 586
 - Polynomial Computational Order of Cycle Cancelling* 589
- 10.4 Network Simplex Algorithm for Optimal Flows 591
 - Linear Dependence in Node–Arc Matrices and Cycles* 591
 - Spanning Trees of Networks* 594
 - Spanning Tree Bases for Network Flow Models* 595
 - Network Basic Solutions* 596
 - Simplex Cycle Directions* 597
 - Network Simplex Algorithm* 598
 - Network Simplex Solution of OOI Application* 598

| | | |
|-------|---|------------|
| 10.5 | Integrality of Optimal Network Flows | 601 |
| | <i>When Optimal Network Flows Must Be Integer</i> | 601 |
| | <i>Total Unimodularity of Node–Arc Incidence Matrices</i> | 603 |
| 10.6 | Transportation and Assignment Models | 604 |
| | <i>Transportation Problems</i> | 604 |
| | <i>Standard Form for Transportation Problems</i> | 605 |
| | <i>Assignment Problems</i> | 607 |
| | <i>Balancing Unequal Sets with Dummy Elements</i> | 610 |
| | <i>Integer Network Flow Solution of Assignment Problems</i> | 610 |
| | <i>CAM Assignment Application Model</i> | 610 |
| 10.7 | Hungarian Algorithm for Assignment Problems | 611 |
| | <i>Primal-Dual Strategy and Initial Dual Solution</i> | 611 |
| | <i>Equality Subgraph</i> | 613 |
| | <i>Labeling to Search for a Primal Solution in the Equality Subgraph</i> | 614 |
| | <i>Dual Update and Revised Equality Subgraph</i> | 616 |
| | <i>Solution Growth Along Alternating Paths</i> | 617 |
| | <i>Computational Order of the Hungarian Algorithm</i> | 617 |
| 10.8 | Maximum Flows and Minimum Cuts | 618 |
| | <i>Improving Feasible Cycle Directions and Flow Augmenting Paths</i> | 620 |
| | <i>The Max Flow Min Cut Algorithm</i> | 621 |
| | <i>Solution of Max Flow Application of Figure 10.25(a) with Algorithm 10E</i> | 621 |
| | <i>Equivalence of Max Flow and Min Cut Values</i> | 624 |
| | <i>Computational Order of Algorithm 10E Effort</i> | 625 |
| 10.9 | Multicommodity and Gain/Loss Flows | 625 |
| | <i>Multicommodity Flows</i> | 625 |
| | <i>Multicommodity Flow Models</i> | 627 |
| | <i>Tractability of Multicommodity Flow Models</i> | 629 |
| | <i>Flows with Gains and Losses</i> | 630 |
| | <i>Gain and Loss Network Flow Models</i> | 631 |
| | <i>Tractability of Network Flows with Gains and Losses</i> | 632 |
| 10.10 | Min/Max Spanning Trees | 633 |
| | <i>Minimum/Maximum Spanning Trees and the Greedy Algorithm</i> | 633 |
| | <i>Solution of the WE Application 10.8 by Greedy Algorithm 10F</i> | 633 |
| | <i>Representing Greedy Results in a Composition Tree</i> | 635 |
| | <i>ILP Formulation of the Spanning Tree Problem</i> | 635 |
| | <i>Computational Order of the Greedy Algorithm</i> | 638 |
| | Exercises | 639 |
| | References | 653 |

CHAPTER 11 DISCRETE OPTIMIZATION MODELS 655

- 11.1 **Lumpy Linear Programs and Fixed Charges 655**
 - Swedish Steel Application with All-or-Nothing Constraints 655*
 - ILP Modeling of All-or-Nothing Requirements 656*
 - Swedish Steel Model with All-or-Nothing Constraints 656*
 - ILP Modeling of Fixed Charges 658*
 - Swedish Steel Application with Fixed Charges 658*
- 11.2 **Knapsack and Capital Budgeting Models 661**
 - Knapsack Problems 661*
 - Capital Budgeting Models 662*
 - Budget Constraints 663*
 - Modeling Mutually Exclusive Choices 664*
 - Modeling Dependencies between Projects 665*
 - NASA Application Model 665*
- 11.3 **Set Packing, Covering, and Partitioning Models 666**
 - Set Packing, Covering, and Partitioning Constraints 667*
 - Minimum Cover EMS Model 669*
 - Maximum Coverage EMS Model 670*
 - Column Generation Models 672*
- 11.4 **Assignment and Matching Models 675**
 - Assignment Constraints 675*
 - CAM Linear Assignment Application Revisited 676*
 - Linear Assignment Models 676*
 - Quadratic Assignment Models 677*
 - Mall Layout Application Model 678*
 - Generalized Assignment Models 680*
 - CDOT Application Model 682*
 - Matching Models 683*
 - Superfi Application Model 684*
 - Tractability of Assignment and Matching Models 684*
- 11.5 **Traveling Salesman and Routing Models 685**
 - Traveling Salesman Problem 685*
 - Symmetric versus Asymmetric Cases of the TSP 686*
 - Formulating the Symmetric TSP 687*
 - Subtours 688*
 - ILP Model of the Symmetric TSP 690*
 - ILP Model of the Asymmetric TSP 690*
 - Quadratic Assignment Formulation of the TSP 692*
 - Problems Requiring Multiple Routes 693*
 - KI Truck Routing Application Model 694*
- 11.6 **Facility Location and Network Design Models 695**
 - Facility Location Models 695*
 - ILP Model of Facilities Location 696*

| | | |
|------|---|-----|
| | <i>Tmark Facilities Location Application Model</i> | 697 |
| | <i>Network Design Models</i> | 699 |
| | <i>Wastewater Network Design Application Model</i> | 701 |
| 11.7 | Processor Scheduling and Sequencing Models | 702 |
| | <i>Single-Processor Scheduling Problems</i> | 703 |
| | <i>Time Decision Variables</i> | 703 |
| | <i>Conflict Constraints and Disjunctive Variables</i> | 704 |
| | <i>Handling of Due Dates</i> | 706 |
| | <i>Processor Scheduling Objective Functions</i> | 706 |
| | <i>ILP Formulation of Minmax Scheduling Objectives</i> | 708 |
| | <i>Equivalences among Scheduling Objective Functions</i> | 710 |
| | <i>Job Shop Scheduling</i> | 710 |
| | <i>Custom Metalworking Application Decision Variables and Objective</i> | 711 |
| | <i>Precedence Constraints</i> | 711 |
| | <i>Conflict Constraints in Job Shops</i> | 712 |
| | <i>Custom Metalworking Application Model</i> | 713 |
| | Exercises | 715 |
| | References | 729 |

CHAPTER 12 EXACT DISCRETE OPTIMIZATION METHODS 731

| | | |
|------|---|-----|
| 12.1 | Solving by Total Enumeration | 731 |
| | <i>Total Enumeration</i> | 732 |
| | <i>Swedish Steel All-or-Nothing Application</i> | 732 |
| | <i>Exponential Growth of Cases to Enumerate</i> | 733 |
| 12.2 | Relaxations of Discrete Optimization Models and Their Uses | 734 |
| | <i>Constraint Relaxations</i> | 735 |
| | <i>Linear Programming Relaxations</i> | 737 |
| | <i>Relaxations Modifying Objective Functions</i> | 738 |
| | <i>Proving Infeasibility with Relaxations</i> | 738 |
| | <i>Solution Value Bounds from Relaxations</i> | 739 |
| | <i>Optimal Solutions from Relaxations</i> | 742 |
| | <i>Rounded Solutions from Relaxations</i> | 744 |
| | <i>Stronger LP Relaxations</i> | 747 |
| | <i>Choosing Big-M Constants</i> | 749 |
| 12.3 | Branch and Bound Search | 751 |
| | <i>Partial Solutions</i> | 752 |
| | <i>Completions of Partial Solutions</i> | 752 |
| | <i>Tree Search</i> | 753 |
| | <i>Incumbent Solutions</i> | 756 |
| | <i>Candidate Problems</i> | 757 |
| | <i>Terminating Partial Solutions with Relaxations</i> | 758 |

| | | |
|------|---|------------|
| | <i>LP-Based Branch and Bound</i> | 760 |
| | <i>Branching Rules for LP-Based Branch and Bound</i> | 761 |
| | <i>LP-Based Branch and Bound Solution of the River Power Application</i> | 762 |
| 12.4 | Refinements to Branch and Bound | 764 |
| | <i>Branch and Bound Solution of NASA Capital Budgeting Application</i> | 764 |
| | <i>Rounding for Incumbent Solutions</i> | 765 |
| | <i>Branch and Bound Family Tree Terminology</i> | 768 |
| | <i>Parent Bounds</i> | 769 |
| | <i>Terminating with Parent Bounds</i> | 769 |
| | <i>Stopping Early: Branch and Bound as a Heuristic</i> | 770 |
| | <i>Bounds on the Error of Stopping with the Incumbent Solution</i> | 771 |
| | <i>Depth First, Best First, and Depth Forward Best Back Sequences</i> | 772 |
| 12.5 | Branch and Cut | 777 |
| | <i>Valid Inequalities</i> | 777 |
| | <i>Branch and Cut Search</i> | 778 |
| | <i>Branch and Cut Solution of the River Power Application</i> | 779 |
| 12.6 | Families of Valid Inequalities | 782 |
| | <i>Gomory Cutting Planes (Pure Integer Case)</i> | 782 |
| | <i>Gomory Mixed-Integer Cutting Planes</i> | 785 |
| | <i>Families of Valid Inequalities from Specialized Models</i> | 787 |
| 12.7 | Cutting Plane Theory | 788 |
| | <i>The Convex Hull of Integer Feasible Solutions</i> | 789 |
| | <i>Linear Programs over Convex Hulls</i> | 791 |
| | <i>Faces, Facets, and Categories of Valid Inequalities</i> | 792 |
| | <i>Affinely Independent Characterization of Facet-Inducing Valid Inequalities</i> | 794 |
| | <i>Partial Dimensional Convex Hulls and Valid Equalities</i> | 795 |
| | Exercises | 797 |
| | References | 810 |

CHAPTER 13 LARGE-SCALE OPTIMIZATION METHODS 811

| | | |
|------|---|-----|
| 13.1 | Delayed Column Generation and Branch and Price | 811 |
| | <i>Models Attractive for Delayed Column Generation</i> | 813 |
| | <i>Partial Master Problems</i> | 815 |
| | <i>Generic Delayed Column Generation Algorithm</i> | 815 |
| | <i>Application of Algorithm 13A to Application 13.1</i> | 815 |
| | <i>Generating Eligible Columns to Enter</i> | 817 |
| | <i>Branch and Price Search</i> | 819 |

- 13.2 Lagrangian Relaxation 822
 - Lagrangian Relaxations* 822
 - Tractable Lagrangian Relaxations* 824
 - Lagrangian Relaxation Bounds and Optima* 825
 - Lagrangian Duals* 827
 - Lagrangian versus Linear Programming Relaxation Bounds* 830
 - Lagrangian Dual Objective Functions* 832
 - Subgradient Search for Lagrangian Bounds* 833
 - Application of Subgradient Search to Numerical Example* 835
- 13.3 Dantzig–Wolfe Decomposition 836
 - Reformulation in Terms of Extreme Points and Extreme Directions* 838
 - Reformulation from GB Application 13.4 Subproblems* 839
 - Delayed Generation of Subproblem Extreme-Point and Extreme-Direction Columns* 840
 - Dantzig–Wolfe Solution of GB Application 13.4* 841
- 13.4 Benders Decomposition 842
 - Benders Decomposition Strategy* 844
 - Optimality in Benders Algorithm 13E* 845
 - Solution of Heart Guardian Application 13.5 with Benders Algorithm 13E* 846
- Exercises 849**
- References 854**

CHAPTER 14 COMPUTATIONAL COMPLEXITY THEORY 855

- 14.1 Problems, Instances, and the Challenge 855
 - The Challenge* 856
- 14.2 Measuring Algorithms and Instances 857
 - Computational Orders* 857
 - Instance Size as the Length of an Encoding* 859
 - Expressions for Encoding Length of All a Problem's Instances* 860
- 14.3 The Polynomial-Time Standard for Well-Solved Problems 861
- 14.4 Polynomial and Nondeterministic-Polynomial Solvability 862
 - Decision versus Optimization Problems* 862
 - Class P - Polynomially Solvable Decision Problems* 863
 - Class NP - Nondeterministic-Polynomially Solvable Decision Problems* 864
 - Polynomial versus Nondeterministic Polynomial Problem Classes* 865

- 14.5 Polynomial-Time Reductions and NP-Hard Problems 866
Polynomial Reductions between Problems 866
NP-Complete and NP-Hard Problems 868
- 14.6 P versus NP 869
The $P \neq NP$ Conjecture 870
- 14.7 Dealing with NP-Hard Problems 871
Special Cases 871
Pseudo-Polynomial Algorithms 871
Average Case Performance 872
Stronger Relaxations and Cuts for B&B and B&C 872
Specialized Heuristics with Provable Worst-Case Performance 872
General Purpose Approximate/Heuristic Algorithms 874
Exercises 875
References 878

CHAPTER 15 HEURISTIC METHODS FOR APPROXIMATE DISCRETE OPTIMIZATION 879

- 15.1 Constructive Heuristics 879
Rudimentary Constructive Search Algorithm 880
Greedy Choices of Variables to Fix 880
Greedy Rule for NASA Application 881
Constructive Heuristic Solution of NASA Application 882
Need for Constructive Search 884
Constructive Search of KI Truck Routing Application 885
- 15.2 Improving Search Heuristics for Discrete Optimization INLPs 886
Rudimentary Improving Search Algorithm 886
Discrete Neighborhoods and Move Sets 887
NCB Application Revisited 888
Choosing a Move Set 889
Rudimentary Improving Search of the NCB Application 891
Multistart Search 892
- 15.3 Tabu and Simulated Annealing Metaheuristics 893
Difficulty with Allowing Nonimproving Moves 894
Tabu Search 894
Tabu Search of the NCB Application 895
Simulated Annealing Search 897
Simulated Annealing Search of NCB Application 899

- 15.4 Evolutionary Metaheuristics and Genetic Algorithms 902
 - Crossover Operations in Genetic Algorithms* 902
 - Managing Genetic Algorithms with Elites, Immigrants, Mutations, and Crossovers* 903
 - Solution Encoding for Genetic Algorithm Search* 904
 - Genetic Algorithm Search of NCB Application* 905
 - Exercises** 906
 - References** 911

CHAPTER 16 UNCONSTRAINED NONLINEAR PROGRAMMING 913

- 16.1 Unconstrained Nonlinear Programming Models 913
 - USPS Single-Variable Application Model* 915
 - Neglecting Constraints to Use Unconstrained Methods* 915
 - Curve Fitting and Regression Problems* 916
 - Linear versus Nonlinear Regression* 917
 - Regression Objective Functions* 918
 - Custom Computer Curve Fitting Application Model* 918
 - Maximum Likelihood Estimation Problems* 919
 - PERT Maximum Likelihood Application Model* 921
 - Smooth versus Nonsmooth Functions and Derivatives* 922
 - Usable Derivatives* 923
- 16.2 One-Dimensional Search 924
 - Unimodal Objective Functions* 924
 - Golden Section Search* 925
 - Golden Section Solution of USPS Application* 927
 - Bracketing and 3-Point Patterns* 929
 - Finding a 3-Point Pattern* 930
 - Quadratic Fit Search* 932
 - Quadratic Fit Solution of USPS Application* 933
- 16.3 Derivatives, Taylor Series, and Conditions for Local Optima in Multiple Dimensions 935
 - Improving Search Paradigm* 935
 - Local Information and Neighborhoods* 936
 - First Derivatives and Gradients* 936
 - Second Derivatives and Hessian Matrices* 937
 - Taylor Series Approximations with One Variable* 939
 - Taylor Series Approximations with Multiple Variables* 940
 - Stationary Points and Local Optima* 941
 - Saddle Points* 943
 - Hessian Matrices and Local Optima* 943
- 16.4 Convex/Concave Functions and Global Optimality 947
 - Convex and Concave Functions Defined* 948
 - Sufficient Conditions for Unconstrained Global Optima* 950
 - Convex/Concave Functions and Stationary Points* 951

- Tests for Convex and Concave Functions* 951
Unimodal versus Convex/Concave Objectives 954
- 16.5 Gradient Search 955
Gradient Search Algorithm 955
Gradient Search of Custom Computer Application 956
Steepest Ascent/Descent Property 958
Zigzagging and Poor Convergence of Gradient Search 959
- 16.6 Newton's Method 959
Newton Step 960
Newton's Method 961
Newton's Method on the Custom Computer Application 962
Rapid Convergence Rate of Newton's Method 963
Computational Trade-offs between Gradient and Newton Search 963
Starting Close with Newton's Method 964
- 16.7 Quasi-Newton Methods and BFGS Search 964
Deflection Matrices 965
Quasi-Newton Approach 965
Guaranteeing Directions Improve 966
BFGS Formula 966
BFGS Search of Custom Computer Application 967
Verifying Quasi-Newton Requirements 971
Approximating the Hessian Inverse with BFGS 972
- 16.8 Optimization without Derivatives and Nelder–Mead 973
Nelder–Mead Strategy 973
Nelder–Mead Direction 976
Nelder–Mead Limited Step Sizes 977
Nelder–Mead Shrinking 979
Nelder–Mead Search of PERT Application 980
- Exercises 981**
References 986

CHAPTER 17 **CONSTRAINED NONLINEAR PROGRAMMING 987**

- 17.1 Constrained Nonlinear Programming Models 987
Beer Belge Location-Allocation Model 988
Linearly Constrained Nonlinear Programs 989
Texaco Gasoline Blending Model 990
Engineering Design Models 992
Oxygen System Engineering Design Model 993
- 17.2 Convex, Separable, Quadratic, and Posynomial Geometric Programming Special NLP Forms 995
Pfizer Optimal Lot Sizing Model 996
Convex Programs 998

| | | |
|------|--|------|
| | <i>Special Tractability of Convex Programs</i> | 1000 |
| | <i>Separable Programs</i> | 1001 |
| | <i>Special Tractability of Separable Programs</i> | 1002 |
| | <i>Quadratic Portfolio Management Model</i> | 1004 |
| | <i>Quadratic Programs Defined</i> | 1005 |
| | <i>Special Tractability of Quadratic Programs</i> | 1006 |
| | <i>Cofferdam Application Model</i> | 1007 |
| | <i>Posynomial Geometric Programs</i> | 1008 |
| | <i>Special Tractability of Posynomial Geometric Programs</i> | 1010 |
| 17.3 | Lagrange Multiplier Methods | 1011 |
| | <i>Reducing to Equality Form</i> | 1011 |
| | <i>Lagrangian Function and Lagrange Multipliers</i> | 1012 |
| | <i>Stationary Points of the Lagrangian Function</i> | 1013 |
| | <i>Lagrangian Stationary Points and the Original Model</i> | 1014 |
| | <i>Lagrange Multiplier Procedure</i> | 1015 |
| | <i>Interpretation of Lagrange Multipliers</i> | 1017 |
| | <i>Limitations of the Lagrangian Approach</i> | 1018 |
| 17.4 | Karush–Kuhn–Tucker Optimality Conditions | 1019 |
| | <i>Fully Differentiable NLP Model</i> | 1019 |
| | <i>Complementary Slackness Conditions</i> | 1019 |
| | <i>Lagrange Multiplier Sign Restrictions</i> | 1020 |
| | <i>KKT Conditions and KKT Points</i> | 1020 |
| | <i>Improving Feasible Directions and Local Optima Revisited</i> | 1022 |
| | <i>KKT Conditions and Existence of Improving Feasible Directions</i> | 1024 |
| | <i>Sufficiency of KKT Conditions for Optimality</i> | 1027 |
| | <i>Necessity of KKT Conditions for Optimality</i> | 1027 |
| 17.5 | Penalty and Barrier Methods | 1028 |
| | <i>Penalty Methods</i> | 1028 |
| | <i>Penalty Treatment of the Service Desk Application</i> | 1030 |
| | <i>Concluding Constrained Optimality with Penalties</i> | 1031 |
| | <i>Differentiability of Penalty Functions</i> | 1031 |
| | <i>Exact Penalty Functions</i> | 1032 |
| | <i>Managing the Penalty Multiplier</i> | 1033 |
| | <i>Sequential Unconstrained Penalty Technique (SUMT)</i> | 1033 |
| | <i>Barrier Methods</i> | 1034 |
| | <i>Barrier Treatment of Service Desk Application</i> | 1035 |
| | <i>Converging to Optimality with Barrier Methods</i> | 1036 |
| | <i>Managing the Barrier Multiplier</i> | 1037 |
| | <i>Sequential Unconstrained Barrier Technique</i> | 1037 |
| 17.6 | Reduced Gradient Algorithms | 1038 |
| | <i>Standard Form for NLPs with Linear Constraints</i> | 1038 |
| | <i>Conditions for Feasible Directions with Linear Constraints</i> | 1040 |
| | <i>Bases of the Main Linear Equalities</i> | 1040 |

| | | |
|-------|---|------|
| | <i>Basic, Nonbasic, and Superbasic Variables</i> | 1041 |
| | <i>Maintaining Equalities by Solving Main Constraints for Basic Variables</i> | 1042 |
| | <i>Active Nonnegativities and Degeneracy</i> | 1042 |
| | <i>Reduced Gradients</i> | 1043 |
| | <i>Reduced Gradient Move Direction</i> | 1044 |
| | <i>Line Search in Reduced Gradient Methods</i> | 1046 |
| | <i>Basis Changes in Reduced Gradient Methods</i> | 1047 |
| | <i>Reduced Gradient Algorithm</i> | 1047 |
| | <i>Reduced Gradient Search of Filter Tuning Application</i> | 1048 |
| | <i>Major and Minor Iterations in Reduced Gradient</i> | 1049 |
| | <i>Second-Order Extensions of Reduced Gradient</i> | 1050 |
| | <i>Generalized Reduced Gradient Procedures for Nonlinear Constraints</i> | 1050 |
| 17.7 | Quadratic Programming Methods | 1051 |
| | <i>General Symmetric Form of Quadratic Programs</i> | 1051 |
| | <i>Quadratic Program Form of the Filter Tuning Application</i> | 1052 |
| | <i>Equality-Constrained Quadratic Programs and KKT Conditions</i> | 1053 |
| | <i>Direct Solution of KKT Conditions for Quadratic Programs</i> | 1054 |
| | <i>Active Set Strategies for Quadratic Programming</i> | 1055 |
| | <i>Step Size with Active Set Methods</i> | 1056 |
| | <i>Stopping at a KKT Point with Active Set Methods</i> | 1057 |
| | <i>Dropping a Constraint from the Active Set</i> | 1058 |
| | <i>Active Set Solution of the Filter Tuning Application</i> | 1059 |
| 17.8 | Sequential Quadratic Programming | 1061 |
| | <i>Lagrangian and Newton Background</i> | 1061 |
| | <i>Sequential Quadratic Programming Strategy</i> | 1062 |
| | <i>Application of Algorithm 17E to Modified Pfizer Application 17.9</i> | 1064 |
| | <i>Approximations to Reduce Computation</i> | 1065 |
| 17.9 | Separable Programming Methods | 1065 |
| | <i>Pfizer Application 17.4 Revisited</i> | 1066 |
| | <i>Piecewise Linear Approximation to Separable Functions</i> | 1067 |
| | <i>Linear Program Representation of Separable Programs</i> | 1069 |
| | <i>Correctness of the LP Approximation to Separable Programs</i> | 1070 |
| | <i>Convex Separable Programs</i> | 1071 |
| | <i>Difficulties with Nonconvex Separable Programs</i> | 1073 |
| 17.10 | Posynomial Geometric Programming Methods | 1073 |
| | <i>Posynomial Geometric Program Form</i> | 1073 |
| | <i>Cofferdam Application Revisited</i> | 1074 |
| | <i>Logarithmic Change of Variables in GPs</i> | 1075 |

| | |
|--|-------------|
| <i>Convex Transformed GP Model</i> | 1076 |
| <i>Direct Solution of the Transformed Primal GP</i> | 1077 |
| <i>Dual of a Geometric Program</i> | 1077 |
| <i>Degrees of Difficulty and Solving the GP Dual</i> | 1079 |
| <i>Recovering a Primal GP Solution</i> | 1080 |
| <i>Derivation of the GP Dual</i> | 1080 |
| <i>Signomial Extension of GPs</i> | 1082 |
| Exercises | 1082 |
| References | 1093 |

APPENDIX: GROUP PROJECTS 1095

SELECTED ANSWERS 1099

INDEX 1123

Preface

It is now nearly two decades since publication of the first edition of my textbook *Optimization in Operations Research*. Since that time thousands of students and hundreds of instructors, researchers, and practitioners have had the opportunity to benefit from its consistent content and accessible design. Of course, not all have seen benefit, but many have written kind reviews and letters expressing their high regard for the book. Also, the Institute of Industrial Engineers honored it with their Joint Publishers Book-of-the-Year Award in 1999.




In this second edition, I have tried to preserve what was best about the original while updating it with new and enhanced content. The goal remains the same—to make the tools of optimization modeling and analysis accessible to advanced undergraduate and beginning graduate students who follow the book in their studies, as well as researchers and working practitioners who use it as a reference for self-study. Emphasis is on the skills and intuitions they can carry away and apply in real settings or later coursework.

Although aimed at that same goal, much is new in the second edition:

- Stochastic optimization is covered for the first time with Stochastic Programming in Chapter 4, and Markov Decision Processes in Chapter 9.
- Coverage of linear programming techniques is expanded in Chapter 6 to encompass dual and primal-dual methods.
- New sections rigorously formalize optimality conditions for linear programming in Chapter 6, and cutting plane theory in Chapter 12.
- Treatment of the Hungarian Algorithm for assignment, and min/max spanning tree methods has been added to Chapter 10.
- A whole new Chapter 13 is devoted to large-scale optimization techniques including Delayed Column Generation, Lagrangian Relaxation, Dantzig–Wolfe Decomposition, and Benders’ Partitioning.
- A whole new Chapter 14 treats the theory of computational complexity to provide a rigorous foundation for comparing problems and algorithms.
- Nonlinear Chapter 17 now includes coverage of the popular Sequential Quadratic Programming method.
- More generally, additional mathematical rigor is added to justifications of methods throughout the book, including tracking computational orders for most.

New topics seek to cover even more completely the full breadth of optimization (or mathematical programming) that might be of interest to the book’s intended audience. Those span linear, integer, nonlinear, network, and dynamic programming models and algorithms, in both single and multi-objective context, and a rich sample of domains where they have been applied.

With content so inclusive, it is important to recognize that almost no reader or course will ever use it all, much less in the exact sequence presented in the book. For that reason, I have tried to make the organization of material as transparent and re-entrant as possible.

Dependencies between sections are minimized and clearly identified with explicit references. One- and two-page **Primers** concisely review prerequisite material where it is needed in the development to save diversions to other sources. To keep the focus on intuitions and strategies behind topics, **Definitions**, **Principles** and **Algorithms** are set out in easy-to-spot boxes, where high-level ideas can be located and absorbed quickly. When more detail is of interest, computations and discussions that may extend to several pages are recapped immediately in concise **Examples** (also marked for easy identification). For readers and instructors seeking more reinforcement with **Exercises** at the end of chapters, convenient icons clearly tag which of those require computer software () or advanced calculators (), and which have answers provided at the back of the book ()

The new edition also builds on my firm belief that making optimization materials accessible and exciting to readers of diverse backgrounds requires making the book a continuing discourse on optimization modeling. Every algorithm and analytic principle is developed in the context of a brief story set out as an **Application**. Also, computational exercises often begin with a formulation step. Many of those stories are derived from real OR applications footnoted in the development. Story settings—however contrived—provide a context for understanding both the needed decision variables, constraints and objectives of model forms, and steps in computation. For example, ideas like improving directions are more intuitive if some quantity in a story, not just a mathematical function, is clearly getting better as an algorithm is pursued. Likewise, binary decision variables become intuitive if the reader can see the either-or nature of some application decisions.

A related conviction is that students cannot really learn any mathematical topic without working with it in homework exercises. That is why the second edition continues the tradition of the first in providing a full range of exercises at the end of each chapter. Some continue from the first edition, but many are new or posed over modified parameter values. The range of exercises begins with verifications of algorithm details, definitions and properties, which are essential to building intuition about the methods. But a range of formulation exercises is also included extending from tiny examples subject to graphic or inspection solution to more complex applications drawn from real OR work that challenge formulation skills. In addition, a new Group Projects appendix details assignments I have used for years to engage student teams more deeply in published reports of actual optimization applications.

Early introductory books in optimization focused heavily on hand application of algorithms to compute solutions of tiny examples. With almost all real optimization now done with the help of large-scale computer software, more recent sources have sometimes limited attention to formulating data sets for submission to one of those algorithms—treating the computation largely as a black box.

I reject both these extremes. Graphic solution of small instances and hand implementation of algorithmic methods are essential if students are to internalize the principles on which the computation is based. The second edition continues my earlier pattern of moving quickly to such intuitive examples as each new concept is introduced. At the same time, no reader will ever grow excited about the power of optimization methods if he or she sees them applied only to tiny examples, much less abstract mathematical forms. That is why many of the examples and exercises in

both the first and second editions of the book ask students to apply available class software on models of greater size, where answers are not apparent until formal methods are shown to reveal them. Brief sections have also been added on coding models for software like AMPL.

Perhaps the greatest challenge in trying to bridge undergraduate and beginning graduate audiences in optimization is the question of mathematical rigor. Elementary treatments simply introduce algorithmic mechanics with little if any argument for their correctness. On the other hand, more advanced books on optimization methods often devolve quickly into rigorous mathematical propositions and formal proofs with almost no discussion of underlying strategies, intuitions, and tractability.

My effort in the first edition was to bridge that gap by focusing on the intuitions and strategies behind methods, and on their relative tractability, while offering only limited arguments for their correctness. In the interest of better serving the introductory graduate and self-study audiences, the second edition adds significantly more rigor to the arguments presented. They are still not stated in theorem or proof format, but most key elements of rationales are now justified.

I am proud of how the long overdue second edition has emerged, and I hope readers will agree that it is a significant advance over the first. I look forward to your comments as the new developments are absorbed.

I want to thank deeply the hundreds of students, friends, and colleagues at Georgia Tech, Purdue and the University of Arkansas for their advice and encouragement as the new edition has taken shape. This goes especially for a series of Graduate Assistants who have helped with exercises and solutions, and for the patience and support of department heads Marlin Thomas, Dennis Engi, John English, Kim Needy, and Ed Pohl. Finally, I need to thank my family—especially my wife Blanca and my son Rob—for their patience and encouragement in my long slog to finish the task.

About the Author



Dr. Ronald L. (Ron) Rardin retired as Distinguished Professor Emeritus in 2013 after a 40-year record of leadership as an educator and researcher in optimization methods and their application culminating after 2007 as John and Mary Lib White Distinguished Professor of Industrial Engineering on the faculty of the University of Arkansas-Fayetteville. He headed the University's Center on Innovation in Healthcare Logistics (CIHL) targeting supply chain and material flow aspects of healthcare operations in collaboration with a variety of healthcare industry organizations. He also took the lead with colleagues at Arkansas in founding the Health Systems Engineering Alliance (HSEA) of industrial engineering academic programs interested in healthcare.

Earlier, Professor Rardin retired in 2006 as Professor Emeritus of Industrial Engineering at Purdue University after completing 24 years there, including directing the Purdue Energy Modeling Research Groups, and playing a leading role in

Purdue's Regenstrief Center for Healthcare Engineering. Previously he had served on the Industrial and Systems Engineering faculty at the Georgia Institute of Technology for 9 years. He also served the profession in a rotation from 2000–2003 as Program Director for Operations Research and Service Enterprise Engineering at the National Science Foundation, including founding the latter program to foster research in service industries.

Dr. Rardin obtained his B.A. and M.P.A. degrees from the University of Kansas, and after working in city government, consulting and distribution for five years, a Ph.D. at Georgia Institute of Technology.

His teaching and research interests center on large-scale optimization modeling and algorithms, especially their application in healthcare and energy. He is an award winning teacher of those topics, and co-author of numerous research papers and two comprehensive textbooks: a graduate text *Discrete Optimization*, published in 1988, and a comprehensive undergraduate textbook on mathematical programming, *Optimization in Operations Research*, which was published in 1998 and received the Institute of Industrial Engineers (IIE) Book of the Year award. Among his many other honors, he is a Fellow of both IIE and the Institute for Operations Research and the Management Sciences (INFORMS), as well as 2012 winner of the IIE's David F. Baker award for career research achievement.

Optimization in Operations Research

This page intentionally left blank

Problem Solving with Mathematical Models

Any student with the most elementary scientific training has encountered the idea of solving problems by analyzing mathematical equations that approximate the physical realities of the universe we inhabit. Countless questions about objects falling, beams shearing, gases diffusing, currents flowing, and so on, are reduced to simple computations upon skillful application of one of the natural laws passed to us by Newton, Ohm, Einstein, and others.

The applicable laws may be less enduring, but “operations” problems such as planning work shifts for large organizations, choosing investments for available funds, or designing facilities for customer service can also be posed in mathematical form. A **mathematical model** is the collection of variables and relationships needed to describe pertinent features of such a problem.

Definition 1.1 | **Operations research (OR)** is the study of how to form mathematical models of complex engineering and management problems and how to analyze them to gain insight about possible solutions.

In this chapter some of the fundamental issues and vocabulary related to operations research are introduced.

1.1 OR APPLICATION STORIES

Operations research techniques have proved useful in an enormous variety of application settings. One of the goals of this book is to expose students to as broad a sample as possible. All application examples, many end-of-chapter exercises, several complete sections, and three full chapters present and analyze stories based on OR applications.

Whenever possible, these problems are drawn from reports of real operations research practice (identified in footnotes). Of course, they are necessarily reduced in size and complexity, and numerical details are almost always made up by the author.

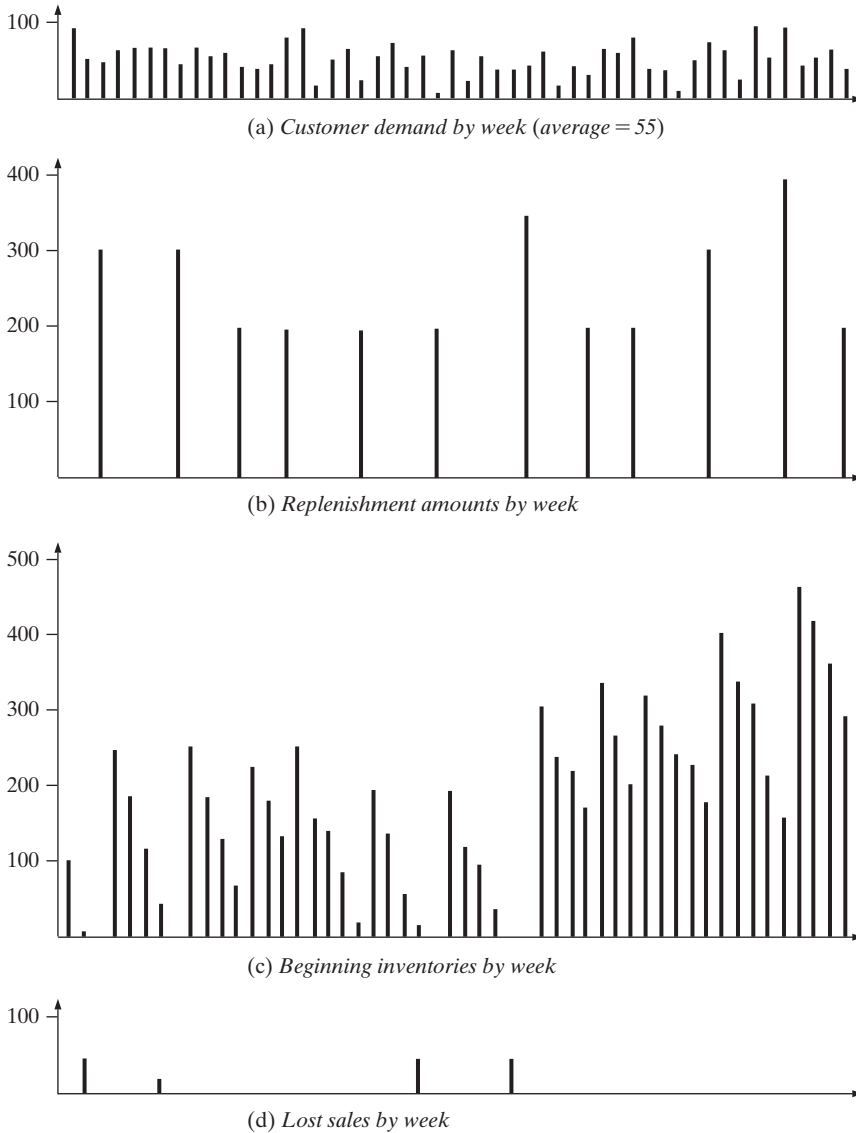


FIGURE 1.1 Mortimer Middleman Example History

Other stories illustrate key elements of standard applications but greatly oversimplify, to facilitate quick learning.

A handful of continuing examples are even smaller and more contrived. They still have a story, but convenience in illustrating methodological issues takes precedence over reality of application.

APPLICATION 1.1: MORTIMER MIDDLEMAN

Our first story is of the totally made-up variety. Mortimer Middleman—friends call him MM—operates a modest wholesale diamond business. Several times each year

MM travels to Antwerp, Belgium, to replenish his diamond supply on the international market. The wholesale price there averages approximately \$700 per carat, but Antwerp market rules require him to buy at least 100 carats each trip. Mortimer and his staff then resell the diamonds to jewelers at a profit of \$200 per carat. Each of the Antwerp trips requires 1 week, including the time for Mortimer to get ready, and costs approximately \$2000.

Customer demand values in Figure 1.1(a) show that business has been good. Over the past year, customers have come in to order an average of 55 carats per week.

Part (c) of Figure 1.1 illustrates Mortimer's problem. Weekly levels of on-hand diamond inventory have varied widely, depending on the ups and downs in sales and the pattern of MM's replenishment trips [Figure 1.1(b)].

Sometimes Mortimer believes that he is holding too much inventory. The hundreds of carats of diamonds on hand during some weeks add to his insurance costs and tie up capital that he could otherwise invest. MM has estimated that these holding costs total 0.5% of wholesale value per week (i.e., $0.005 \times \$700 = \3.50 per carat per week).

At other times, diamond sales—and Mortimer's \$200 per carat profit—have been lost because customer demand exceeded available stock [see Figure 1.1(d)]. When a customer calls, MM must either fill the order on the spot or lose the sale.

Adding this all up for the past year, MM estimates holding costs of \$38,409, unrealized profits from lost sales of \$31,600, and resupply travel costs of \$24,000, making the annual total \$94,009. Can he do better?

1.2 OPTIMIZATION AND THE OPERATIONS RESEARCH PROCESS

Operations research deals with **decision problems** like that of Mortimer Middleman by formulating and analyzing mathematical models—mathematical representations of pertinent problem features. Figure 1.2 illustrates this OR process.

The process begins with formulation or modeling. We define the variables and quantify the relationships needed to describe relevant system behavior.

Next comes analysis. We apply our mathematical skills and technology to see what conclusions the model suggests. Notice that these conclusions are drawn from

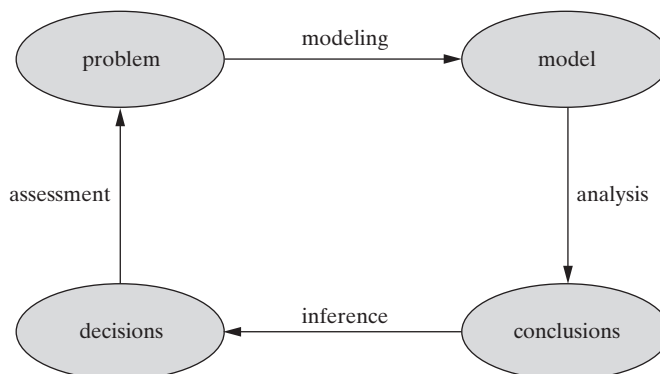


FIGURE 1.2 Operations Research Process

the model, not from the problem that it is intended to represent. To complete the process, we must engage in inference, that is, argue that conclusions drawn from the model are meaningful enough to infer decisions for the person or persons with the problem.

Often, an assessment of decisions inferred in this way shows them to be too inadequate or extreme for implementation. Further thought leads to revised modeling, and the loop continues.

Decisions, Constraints, and Objectives

We always begin modeling by focusing on three dimensions of the problem:

Definition 1.2 | The three fundamental concerns in forming operations research models are (a) the **decisions** open to decision makers, (b) the **constraints** limiting decision choices, and (c) the **objectives** making some decisions preferred to others.

In dealing with virtually any decision problem—engineering, management, or even personal—explicitly defining the decisions, constraints, and objectives helps to clarify the issues. Mortimer is obviously the decision maker in our diamond inventory management example. What decisions does he get to make?

Actually, MM makes hundreds of decisions each year about when to replenish his stock and how much to buy. However, it is common in inventory management circumstances such as Mortimer’s to reduce the question to two policy decisions: What **reorder point** level of inventory should trigger a decision to buy new stock, and what **order quantity** should be purchased each time? These two variables constitute our decisions. We presume that each time on-hand inventory falls below the reorder point, Mortimer will head to Antwerp to buy a standard reorder quantity.

The next issue is constraints. What restrictions limit MM’s decision choices? In this example there aren’t very many. It is only necessary that both decisions be non-negative numbers and that the order quantity conform to the 100 carat minimum of the Antwerp market.

The third element is objectives. What makes one decision better than another? In MM’s case the objective is clearly to minimize cost. More precisely, we want to minimize the sum of holding, replenishment, and lost-sales costs.

Summarizing in a verbal model or word description, our goal is *to choose a non-negative reorder point and a nonnegative reorder quantity to minimize the sum of holding, replenishment, and lost-sales costs subject to the reorder quantity being at least 100.*

Optimization and Mathematical Programming

Verbal models can help organize an analyst’s thinking, but in this book we address a higher standard. We deal exclusively with optimization (also called mathematical programming).

Definition 1.3 | **Optimization models** (also called **mathematical programs**) represent problem choices as decision variables and seek values that maximize or minimize objective functions of the decision variables subject to constraints on variable values expressing the limits on possible decision choices.

With our Mortimer Middleman example, the decision variables are

$q \triangleq$ reorder quantity purchased on each replenishment trip

$r \triangleq$ reorder point signaling the need for replenishment

(Here and throughout \triangleq means “is defined to be.”) Constraints require only that

$$q \geq 100$$

$$r \geq 0$$

The objective function,

$c(q, r) \triangleq$ total cost using a reorder quantity of q and a reorder point r

remains to be explicitly represented mathematically. We seek to minimize $c(q, r)$ over values of q and r satisfying all constraints.

Constant-Rate Demand Assumption

How we formulate constraints and objectives in terms of decision variables depends on what assumptions we are willing to make about the underlying system. We begin with a strong assumption regarding **constant-rate demand**: Assume that demand occurs at a constant rate of 55 carats per week. It is clear in Figure 1.1(a) that the demand rate is not exactly constant, but it does average 55 carats per week. Assuming that it is 55 carats in every week leads to some relatively simple analysis.

If the demand rate is constant, the pattern of on-hand inventory implied by a particular q and r will take one of the periodic “sawtooth” forms illustrated in Figure 1.3. Each time a shipment arrives, inventory will increase by order size q , then decline at the rate of 55 carats per week, producing regular cycles. Part (a) shows a case where inventory never runs out. A **safety stock** of (theoretically) untouched inventory protects against demand variability we have ignored. At the other extreme is part (c). Sales are lost because inventory runs out during the **lead time** between reaching the reorder point r and arrival of a new supply. Part (b) has neither safety stock nor lost sales. Stock runs out just as new supply arrives.

Back of Envelope Analysis

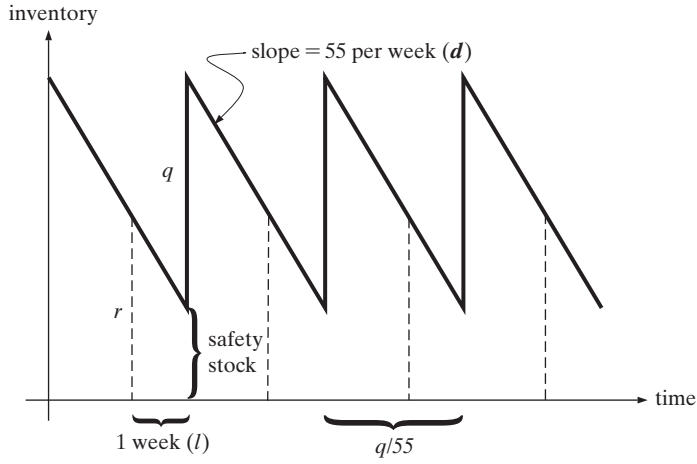
In cases where there are no lost sales [Figure 1.3(a) and (b)] it is easy to compute the length of each sawtooth cycle.

$$\frac{\text{order quantity}}{\text{demand rate}} = \frac{q}{55}$$

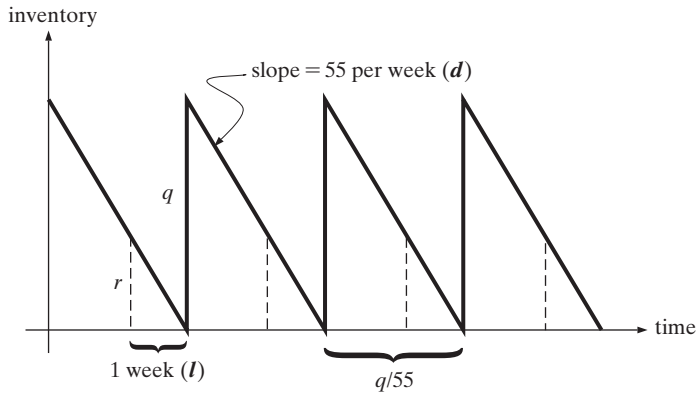
With lost sales [Figure 1.3(c)], each cycle is extended by a period when MM is out of stock that depends on both q and r .

Clearly, both modeling and analysis would be easier if we could ignore the lost-sales case. Can we afford to neglect lost sales? As in so many OR problems, a bit of crude “back of envelope” examination of the relevant costs will help us decide.

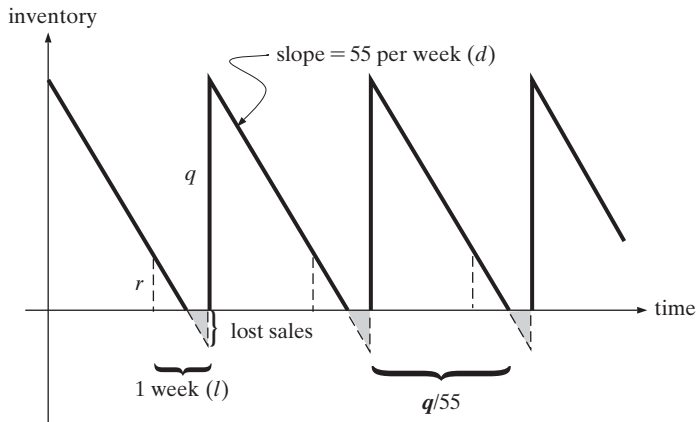
Lost sales may occur under the best of plans because of week-to-week variation in demand. Under our constant-rate demand assumption, however,



(a) With safety stock



(b) No safety stock or lost sales



(c) With lost sales

FIGURE 1.3 Inventories Under Constant-Rate Demand

there is no variation. Furthermore, MM can afford to add a unit to q and carry it for up to

$$\frac{\text{cost of lost sale}}{\text{weekly holding cost}} = \frac{\$200}{\$3.50} \approx 57.1 \text{ weeks}$$

rather than lose a carat of sales. Since the history in Figure 1.1 shows that inventory typically has been held no more than 4 to 6 weeks, it seems safe to make a second assumption regarding **no lost sales**: Assume that lost sales are not allowed.

Constant-Rate Demand Model

Since customers order a constant-rate 55 carats during the 1 week it takes Mortimer to carry out an Antwerp trip, both inventory at order arrival and lost sales can be computed by comparing 55 to r . If $r < 55$, we lose $(55 - r)$ carats of sales each cycle, something we have decided not to permit. Thus we may deduce the constraint

$$r \geq 55$$

With r restricted to be at least 55, $(r - 55)$ is the safety stock, and the cycle of rising and falling inventory repeats every $q/55$ weeks. Inventory on hand ranges from $(r - 55)$ at the low point of a cycle to $(r - 55) + q$ as a shipment arrives. The average will be the midpoint of these values, $(r - 55) + q/2$.

We are finally in a position to express all relevant costs. Holding cost per week is just the average inventory held times \$3.50. Replenishment cost per week is \$2000 divided by the cycle length or time between replenishments. Our first optimization model is

$$\begin{aligned} \text{minimize } c &= 3.50 \left[(r - 55) + \frac{q}{2} \right] + \frac{2000}{q/55} & (1.1) \\ \text{subject to } & q \geq 100, \quad r \geq 55 \end{aligned}$$

Feasible and Optimal Solutions

Remember that our goal is to help Mortimer make decisions. Since the decisions are the variables in our model, we would like to characterize good values for **decision variables** q and r .

Definition 1.4 | A **feasible solution** is a choice of values for the decision variables that satisfies all constraints. **Optimal solutions** are feasible solutions that achieve objective function value(s) as good as those of any other feasible solutions.

For example, $q = 200, r = 90$ is feasible in constant-rate demand model (1.1) because both constraints are satisfied: $200 \geq 100$ and $90 \geq 55$.

Here we can go farther and find an optimal solution. To begin, notice that if r deviates from demand 55, we incur extra holding cost and that no constraint prevents choosing r exactly 55. We conclude that

$$r^* = 55$$

will tell MM the perfect moment to start travel preparations. The asterisk (*) or **star** on a variable always denotes its optimal value.

Substituting this optimal choice of r of (1.1), the objective function reduces to

$$c(q, r) \triangleq 3.50 \left(\frac{q}{2} \right) + 2000 \left(\frac{55}{q} \right) \quad (1.2)$$

Elementary calculus will tell us how to finish (differentiate with respect to q and solve for a suitable point where the derivative is zero). To avoid being diverted by mathematical details in this introductory chapter, we leave the computation as an exercise for the reader.

The graphic presentation of cost function (1.2) in Figure 1.4 confirms the calculus result that the minimum average weekly cost occurs at

$$q^* = \pm \sqrt{\frac{2(2000)(55)}{3.50}} \approx 250.7$$

Since this value easily satisfies the $q \geq 100$ constraint, it is optimal.

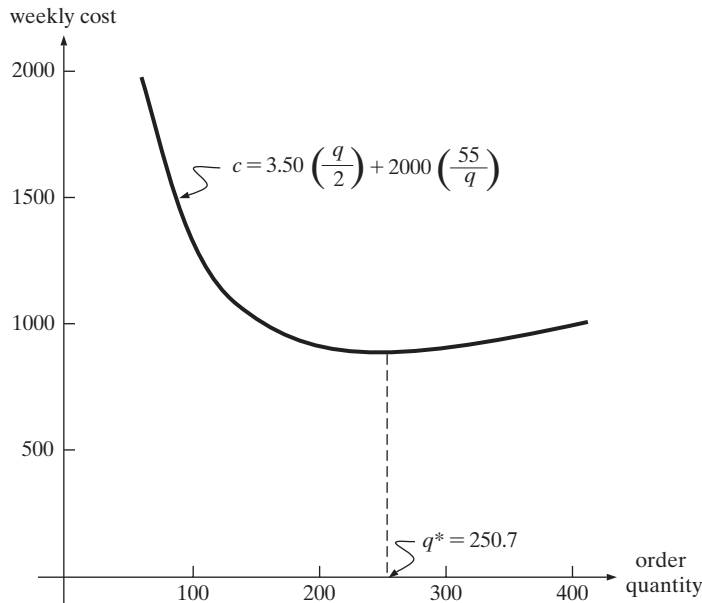


FIGURE 1.4 Optimal MM Order Quantity Under Constant-Rate Demand

To summarize, our assumptions of constant-rate demand and no lost sales have led us to advise Mortimer to go to Antwerp whenever inventory drops below $r^* = 55$ carats and to buy $q^* = 250.7$ carats of new diamonds each trip. Substituting these values in the objective function of (1.1), total cost should be about \$877.50 per week or \$45,630 per year—quite an improvement over Mortimer’s real experience of \$94,009.

1.3 SYSTEM BOUNDARIES, SENSITIVITY ANALYSIS, TRACTABILITY, AND VALIDITY

The modeling in Section 1.2 took as given many quantities, such as the demand per week and the cost per carat held, then computed optimal values for reorder point and reorder quantity. A line between those items taken as settled and those to be decided is called the **system boundary**. Figure 1.5 illustrates how **parameters**—quantities taken as given—define objective functions and constraints applicable to the decision model inside. Together, parameters and decision variables determine results measured as **output variables**.

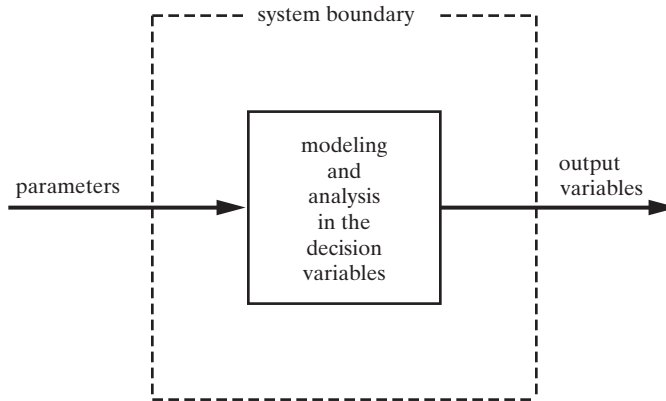


FIGURE 1.5 System Boundaries

EOQ Under Constant-Rate Demand

Only cost c is an output variable in our constant-rate demand model of Mortimer Middleman's problem. Enumerating the parameters, let

$d \triangleq$ weekly demand (55 carats)

$f \triangleq$ fixed cost of replenishment (\$2000)

$h \triangleq$ cost per carat per week for holding inventory (\$3.50)

$s \triangleq$ cost per carat of lost sales (\$200)

$\ell \triangleq$ lead time between reaching the reorder point and receiving a new supply (1 week)

$m \triangleq$ minimum order size (100 carats)

A great attraction of our constant-rate demand analysis is that it can be done just as well in terms of these symbols. If lost sales are not allowed, repetition of the analysis (calculus) in terms of symbolic parameters will cause us to conclude that

Principle 1.5

$$\begin{aligned} \text{optimal reorder quantity } q^* &= \sqrt{\frac{2fd}{h}} \\ \text{optimal reorder point } r^* &= \ell d \end{aligned}$$