

AIMMS

Optimization Modeling

AIMMS 3

April 3, 2011

AIMMS

Optimization Modeling

Paragon Decision Technology

Johannes Bisschop

Copyright © 1993–2010 by Paragon Decision Technology B.V. All rights reserved.

Paragon Decision Technology B.V.	Paragon Decision Technology Inc.	Paragon Decision Technology Pte.
Schipholweg 1	500 108th Avenue NE	Ltd.
2034 LS Haarlem	Ste. # 1085	80 Raffles Place
The Netherlands	Bellevue, WA 98004	UOB Plaza 1, Level 36-01
Tel.: +31 23 5511512	USA	Singapore 048624
Fax: +31 23 5511517	Tel.: +1 425 458 4024	Tel.: +65 9640 4182
	Fax: +1 425 458 4025	

Email: info@aimms.com
WWW: www.aimms.com

AIMMS is a registered trademark of Paragon Decision Technology B.V. IBM ILOG CPLEX and sc CPLEX is a registered trademark of IBM Corporation. GUROBI is a registered trademark of Gurobi Optimization, Inc. KNITRO is a registered trademark of Ziena Optimization, Inc. XPRESS-MP is a registered trademark of FICO Fair Isaac Corporation. MOSEK is a registered trademark of Mosek ApS. WINDOWS and EXCEL are registered trademarks of Microsoft Corporation. \TeX , \LaTeX , and $\AMS-\LaTeX$ are trademarks of the American Mathematical Society. LUCIDA is a registered trademark of Bigelow & Holmes Inc. ACROBAT is a registered trademark of Adobe Systems Inc. Other brands and their products are trademarks of their respective holders.

Information in this document is subject to change without notice and does not represent a commitment on the part of Paragon Decision Technology B.V. The software described in this document is furnished under a license agreement and may only be used and copied in accordance with the terms of the agreement. The documentation may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Paragon Decision Technology B.V.

Paragon Decision Technology B.V. makes no representation or warranty with respect to the adequacy of this documentation or the programs which it describes for any particular purpose or with respect to its adequacy to produce any particular result. In no event shall Paragon Decision Technology B.V., its employees, its contractors or the authors of this documentation be liable for special, direct, indirect or consequential damages, losses, costs, charges, claims, demands, or claims for lost profits, fees or expenses of any nature or kind.

In addition to the foregoing, users should recognize that all complex software systems and their documentation contain errors and omissions. The authors, Paragon Decision Technology B.V. and its employees, and its contractors shall not be responsible under any circumstances for providing information or corrections to errors and omissions discovered at any time in this book or the software it describes, whether or not they are aware of the errors or omissions. The authors, Paragon Decision Technology B.V. and its employees, and its contractors do not recommend the use of the software described in this book for applications in which errors or omissions could threaten life, injury or significant loss.

This documentation was typeset by Paragon Decision Technology B.V. using \LaTeX and the LUCIDA font family.

About AIMMS

AIMMS was introduced by Paragon Decision Technology as a new type of mathematical modeling tool in 1993—an integrated combination of a modeling language, a graphical user interface, and numerical solvers. AIMMS has proven to be one of the world's most advanced development environments for building optimization-based decision support applications and advanced planning systems. Today, it is used by leading companies in a wide range of industries in areas such as supply chain management, energy management, production planning, logistics, forestry planning, and risk-, revenue-, and asset- management. In addition, AIMMS is used by universities worldwide for courses in Operations Research and Optimization Modeling, as well as for research and graduation projects.

History

AIMMS is far more than just another mathematical modeling language. True, the modeling language is state of the art for sure, but alongside this, AIMMS offers a number of advanced modeling concepts not found in other languages, as well as a full graphical user interface both for developers and end-users. AIMMS includes world-class solvers (and solver links) for linear, mixed-integer, and nonlinear programming such as BARON, CPLEX, CONOPT, GUROBI, KNITRO, LGO, PATH, SNOPT and XA, and can be readily extended to incorporate other advanced commercial solvers available on the market today. In addition, concepts as stochastic programming and robust optimization are available to include data uncertainty in your models.

What is AIMMS?

Mastering AIMMS is straightforward since the language concepts will be intuitive to Operations Research (OR) professionals, and the point-and-click graphical interface is easy to use. AIMMS comes with comprehensive documentation, available electronically and in book form.

*Mastering
AIMMS*

AIMMS provides an ideal platform for creating advanced prototypes that are then easily transformed into operational end-user systems. Such systems can then be used either as

*Types of AIMMS
applications*

- stand-alone applications, or
- optimization components.

Application developers and operations research experts use AIMMS to build complex and large scale optimization models and to create a graphical end-user interface around the model. AIMMS-based applications place the power of the most advanced mathematical modeling techniques directly into the hands of end-users, enabling them to rapidly improve the quality, service, profitability, and responsiveness of their operations.

Stand-alone applications

Independent Software Vendors and OEMs use AIMMS to create complex and large scale optimization components that complement their applications and web services developed in languages such as C++, Java, .NET, or Excel. Applications built with AIMMS-based optimization components have a shorter time-to-market, are more robust and are richer in features than would be possible through direct programming alone.

Optimization components

Companies using AIMMS include

AIMMS users

- | | |
|-------------------|--------------------------|
| ■ ABN AMRO | ■ Merck |
| ■ Areva | ■ Owens Corning |
| ■ Bayer | ■ Perdigão |
| ■ Bluescope Steel | ■ Petrobras |
| ■ BP | ■ Philips |
| ■ CST | ■ PriceWaterhouseCoopers |
| ■ ExxonMobil | ■ Reliance |
| ■ Gaz de France | ■ Repsol |
| ■ Heineken | ■ Shell |
| ■ Innovene | ■ Statoil |
| ■ Lufthansa | ■ Unilever |

Universities using AIMMS include Budapest University of Technology, Carnegie Mellon University, George Mason University, Georgia Institute of Technology, Japan Advanced Institute of Science and Technology, London School of Economics, Nanyang Technological University, Rutgers University, Technical University of Eindhoven, Technische Universitt Berlin, UIC Bioengineering, Universidade Federal do Rio de Janeiro, University of Groningen, University of Pittsburgh, University of Warsaw, and University of the West of England.

A more detailed list of AIMMS users and reference cases can be found on our website www.aimms.com.

Contents

About AIMMS	v
Contents	vii
Preface	xiii
What is in the AIMMS documentation	xiii
What's in the Optimization Modeling guide	xvi
The authors	xviii
<hr/>	
Part I Introduction to Optimization Modeling	2
<hr/>	
1 Background	2
1.1 What is a model?	2
1.2 Why use models?	3
1.3 The role of mathematics	4
1.4 The modeling process	6
1.5 Application areas	7
1.5.1 Production	7
1.5.2 Production and inventory management	8
1.5.3 Finance	9
1.5.4 Manpower planning	10
1.5.5 Agricultural economics	11
1.6 Summary	12
2 Formulating Optimization Models	13
2.1 Formulating linear programming models	13
2.1.1 Introduction	13
2.1.2 Example of a linear programming model	14
2.1.3 Picturing the formulation and the solution	17
2.2 Formulating mixed integer programming models	23
2.3 Formulating nonlinear programming models	27
2.4 Summary	31

3	Algebraic Representation of Models	32
3.1	Explicit form	32
3.2	Symbolic form	33
3.3	Symbolic-indexed form	34
3.4	AIMMS form	37
3.5	Translating a verbal problem into a model	39
3.6	Summary	40
4	Sensitivity Analysis	41
4.1	Introduction	41
4.2	Shadow prices	42
4.3	Reduced costs	45
4.4	Sensitivity ranges with constant objective function value	48
4.5	Sensitivity ranges with constant basis	49
4.6	Summary	51
5	Network Flow Models	52
5.1	Introduction	52
5.2	Example of a network flow model	53
5.3	Network formulation	55
5.4	Solution and sensitivity analysis	57
5.5	Pure network flow models	58
5.6	Other network models	60
5.7	Summary	61
<hr/> Part II General Optimization Modeling Tricks		63
6	Linear Programming Tricks	63
6.1	Absolute values	63
6.2	A minimax objective	66
6.3	A fractional objective	67
6.4	A range constraint	68
6.5	A constraint with unknown-but-bounded coefficients	69
6.6	A probabilistic constraint	71
6.7	Summary	74
7	Integer Linear Programming Tricks	75
7.1	A variable taking discontinuous values	75
7.2	Fixed costs	76
7.3	Either-or constraints	77
7.4	Conditional constraints	79
7.5	Special Ordered Sets	80
7.6	Piecewise linear formulations	81
7.7	Elimination of products of variables	83

7.8	Summary	85
<hr/>		
Part III Basic Optimization Modeling Applications		87
<hr/>		
8	An Employee Training Problem	87
8.1	Hiring and training of flight attendants	87
8.2	Model formulation	88
8.3	Solutions from conventional solvers	90
8.4	Solutions from rounding heuristics	92
8.5	Introducing probabilistic constraints	93
8.6	Summary	95
	Exercises	95
9	A Media Selection Problem	96
9.1	The scheduling of advertising media	96
9.2	Model formulation	97
9.3	Adding logical conditions	99
9.4	Set covering and related models	102
9.5	Summary	103
	Exercises	104
10	A Diet Problem	105
10.1	Example of a diet problem	105
10.2	Model formulation	106
10.3	Quantities and units	108
10.4	Summary	112
	Exercises	112
11	A Farm Planning Problem	113
11.1	Problem description	113
11.2	Model formulation	116
11.3	Model results	120
11.4	Summary	122
	Exercises	122
12	A Pooling Problem	123
12.1	Problem description	123
12.2	Model description	126
12.3	A worked example	129
12.4	Summary	132
	Exercises	132

Part IV Intermediate Optimization Modeling Applications	134
13 A Performance Assessment Problem	134
13.1 Introduction and terminology	134
13.2 Relative efficiency optimization	135
13.3 A worked example	138
13.4 Computational issues	142
13.5 Summary	142
Exercises	143
14 A Two-Level Decision Problem	144
14.1 Problem description	144
14.2 Model formulation	146
14.3 Algorithmic approach	148
14.4 Optimal solution	150
14.5 Alternative solution approach	154
14.6 Summary	156
Exercises	157
15 A Bandwidth Allocation Problem	158
15.1 Problem description	158
15.2 Formulation I: enumerating bandwidth intervals	160
15.2.1 Preventing overlap using pairs of allocations	162
15.2.2 Preventing overlap using channel constraints	163
15.3 Formulation II: avoiding bandwidth interval construction	164
15.3.1 Improving sparsity in overlap constraints	166
15.4 Summary	167
Exercises	167
16 A Power System Expansion Problem	168
16.1 Introduction to methods for uncertainty	168
16.2 A power system expansion problem	170
16.3 A deterministic expansion model	173
16.4 What-if approach	174
16.5 Stochastic programming approach	176
16.6 Summary	179
Exercises	180
17 An Inventory Control Problem	181
17.1 Introduction to multi-stage concepts	181
17.2 An inventory control problem	184
17.3 A multi-stage programming model	185
17.4 Equivalent alternative objective function	189
17.5 A worked example	190
17.6 Summary	192

Exercises	193
---------------------	-----

Part V Advanced Optimization Modeling Applications	195
---	------------

18 A Portfolio Selection Problem	195
18.1 Introduction and background	195
18.2 A strategic investment model	198
18.3 Required mathematical concepts	200
18.4 Properties of the strategic investment model	203
18.5 Example of strategic investment model	206
18.6 A tactical investment model	208
18.7 Example of tactical investment model	211
18.8 One-sided variance as portfolio risk	213
18.9 Adding logical constraints	215
18.10 Piecewise linear approximation	216
18.11 Summary	219
Exercises	220
19 A File Merge Problem	221
19.1 Problem description	221
19.2 Mathematical formulation	224
19.3 Solving large instances	228
19.4 The simplex method	229
19.5 Algorithmic approach	230
19.6 Summary	232
Exercises	233
20 A Cutting Stock Problem	235
20.1 Problem description	235
20.2 The initial model formulation	236
20.3 Delayed cutting pattern generation	238
20.4 Extending the original cutting stock problem	242
20.5 Summary	244
Exercises	244
21 A Telecommunication Network Problem	245
21.1 Problem description	245
21.2 Bottleneck identification model	246
21.3 Path generation technique	250
21.4 A worked example	255
21.5 Summary	257
Exercises	257

22 A Facility Location Problem	258
22.1 Problem description	258
22.2 Mathematical formulation	260
22.3 Solve large instances through decomposition	261
22.4 Benders' decomposition with feasible subproblems	262
22.5 Convergence of Benders' decomposition	266
22.6 Formulating dual models	267
22.7 Application of Benders' decomposition	269
22.8 Computational considerations	272
22.9 A worked example	274
22.10 Summary	276
Exercises	277
<hr/> Part VI Appendices	<hr/> 279
Keyword Table	279
Index	280
Bibliography	285

Preface

The printed AIMMS documentation consists of three books

Three AIMMS books

- AIMMS—*The User's Guide*,
- AIMMS—*The Language Reference*, and
- AIMMS—*Optimization Modeling*.

The first two books emphasize different aspects in the use of the AIMMS system, while the third book is a general introduction to optimization modeling. All books can be used independently.

In addition to the printed versions, these books are also available on-line in the ADOBE Portable Document Format (PDF). Although new printed versions of the documentation will become available with every new functional AIMMS release, small additions to the system and small changes in its functionality in between functional releases are always directly reflected in the online documentation, but not necessarily in the printed material. Therefore, the online versions of the AIMMS books that come with a particular version of the system should be considered as the authoritative documentation describing the functionality regarding that particular AIMMS version.

Available online

Which changes and bug fixes are included in particular AIMMS releases are described in the associated release notes.

Release notes

What is in the AIMMS documentation

The AIMMS User's Guide provides a global overview of how to use the AIMMS system itself. It is aimed at application builders, and explores AIMMS' capabilities to help you create a model-based application in an easy and maintainable manner. The guide describes the various graphical tools that the AIMMS system offers for this task. It is divided into five parts.

The User's Guide

- Part I—*Introduction to AIMMS*—what is AIMMS and how to use it.
- Part II—*Creating and Managing a Model*—how to create a new model in AIMMS or manage an existing model.
- Part III—*Creating an End-User Interface*—how to create an intuitive and interactive end-user interface around a working model formulation.

- Part IV—*Data Management*—how to work with cases and datasets.
- Part V—*Miscellaneous*—various other aspects of AIMMS which may be relevant when creating a model-based end-user application.

The AIMMS Language Reference provides a complete description of the AIMMS modeling language, its underlying data structures and advanced language constructs. It is aimed at model builders only, and provides the ultimate reference to the model constructs that you can use to get the most out of your model formulations. The guide is divided into seven parts.

The Language Reference

- Part I—*Preliminaries*—provides an introduction to, and overview of, the basic language concepts.
- Part II—*Nonprocedural Language Components*—describes AIMMS' basic data types, expressions, and evaluation structures.
- Part III—*Procedural Language Components*—describes AIMMS' capabilities to implement customized algorithms using various execution and flow control statements, as well as internal and external procedures and functions.
- Part IV—*Sparse Execution*—describes the fine details of the sparse execution engine underlying the AIMMS system.
- Part V—*Optimization Modeling Components*—describes the concepts of variables, constraints and mathematical programs required to specify an optimization model.
- Part VI—*Data Communication Components*—how to import and export data from various data sources, and create customized reports.
- Part VII—*Advanced Language Components*—describes various advanced language features, such as the use of units, modeling of time and communicating with the end-user.

The book on optimization modeling provides not only an introduction to modeling but also a suite of worked examples. It is aimed at users who are new to modeling and those who have limited modeling experience. Both basic concepts and more advanced modeling techniques are discussed. The book is divided into five parts:

Optimization Modeling

- Part I—*Introduction to Optimization Modeling*—covers what models are, where they come from, and how they are used.
- Part II—*General Optimization Modeling Tricks*—includes mathematical concepts and general modeling techniques.
- Part III—*Basic Optimization Modeling Applications*—builds on an understanding of general modeling principles and provides introductory application-specific examples of models and the modeling process.
- Part IV—*Intermediate Optimization Modeling Applications*—is similar to part III, but with examples that require more effort and analysis to construct the corresponding models.

- Part V—*Advanced Optimization Modeling Applications*—provides applications where mathematical concepts are required for the formulation and solution of the underlying models.

In addition to the three major AIMMS books, there are several separate documents describing various deployment features of the AIMMS software. They are:

*Documentation
of deployment
features*

- AIMMS—*The Function Reference*,
- AIMMS—*The COM Object User's Guide and Reference*,
- AIMMS—*The Multi Agent and Web Services User's Guide*,
- AIMMS—*The Excel Add-In User's Guide*, and
- AIMMS—*The Open Solver Interface User's Guide and Reference*.

These documents are only available in PDF format.

The AIMMS documentation is complemented with a number of help files that discuss the finer details of particular aspects of the AIMMS system. Help files are available to describe:

Help files

- the execution and solver options which you can set to globally influence the behavior of the AIMMS' execution engine,
- the finer details of working with the graphical modeling tools, and
- a complete description of the properties of end-user screens and the graphical data objects which you can use to influence the behavior and appearance of an end-user interface built around your model.

The AIMMS help files are both available as Windows help files, as well as in PDF format.

Two tutorials on AIMMS in PDF format provide you with some initial working knowledge of the system and its language. One tutorial is intended for beginning users, while the other is aimed at professional users of AIMMS.

AIMMS tutorials

As the entire AIMMS documentation is available in PDF format, you can use the search functionality of Acrobat Reader to search through all AIMMS documentation for the information you are looking for. From within the **Help** menu of the AIMMS software you can access a pre-built search index to quicken the search process.

*Searching the
documentation*

AIMMS comes with an extensive model library, which contains a variety of examples to illustrate simple and advanced applications containing particular aspects of both the language and the graphical user interface. You can find the AIMMS model library in the Examples directory in the AIMMS installation directory. The Examples directory also contains an AIMMS project providing an index to all examples, which you can use to search for examples that illustrate specific aspects of AIMMS.

*AIMMS model
library*

What's in the Optimization Modeling guide

Part I—Introduction to Optimization Modeling—covers what models are, where they come from, and how they are used. This part is for anyone who is new to the area of optimization modeling.

Introduction to optimization modeling

- Chapter 1. “Background,” gives background information on optimization modeling, and highlights the process from a real life problem to a well-posed problem statement.
- Chapter 2. “Formulating Optimization Models,” lays the foundation for all chapters that follow. It gives an overview of linearly constrained optimization models and their characteristics. These models are then extended to include integer and nonlinear constraints.
- Chapter 3. “Algebraic Representation of Models,” compares different formulations of the same model, and introduces the fundamental concept of *index notation*.
- Chapter 4. “Sensitivity Analysis,” gives an extensive introduction to the use of *marginal values* for sensitivity analysis.
- Chapter 5. “Network Flow Models,” describes classes of network flow models as well as a example network formulation in AIMMS.

Part II—General Optimization Modeling Tricks—includes mathematical concepts and modeling tricks for linear and mixed-integer linear programming.

General optimization modeling tricks

- Chapter 6. “Linear Programming Tricks,” provides some standard formulation tricks for linear programming.
- Chapter 7. “Integer Linear Programming Tricks,” describes some standard formulation tricks for mixed-integer linear programming, and introduces the concept of Special Ordered Sets.

Part III—Basic Optimization Modeling Applications—builds on an understanding of general modeling principles and gives introductory application-specific examples of models and the modeling process.

Basic optimization modeling applications

- Chapter 8. “An Employee Training Problem,” comes from an application for hiring and training new flight attendants and incorporates a *heuristic* for rounding a continuous solution into an integer-valued solution.
- Chapter 9. “A Media Selection Problem,” comes from an application for deciding how best to market a product based on demographic data and contains examples of modeling *logical conditions* and *set covering, packing, and partitioning*.
- Chapter 10. “A Diet Problem,” comes from an application for planning a healthy and inexpensive diet. The model shows how to use *units* and to *change the formulation* of a linear program into a mixed-integer program.
- Chapter 11. “A Farm Planning Problem,” comes from an application for farm management and illustrates the use of *units*.

- Chapter 12. “A Pooling Problem,” comes from an application in which final products are blended from intermediate products to meet various quality specifications. Both *linear and nonlinear blending rules* are introduced.

Part IV—Intermediate Optimization Modeling Applications—similar to part III, but with examples that require extra effort and analysis to construct the corresponding models.

*Intermediate
optimization
modeling
applications*

- Chapter 13. “A Performance Assessment Problem,” comes from an application for *evaluating the performance* of several comparable organizations. The chapter illustrates a step-wise approach to determine efficient decision making units.
- Chapter 14. “A Two-Level Decision Problem,” comes from an application in which waste water *regulations* are *enforced through taxes and subsidies*. Both an iterative approach and a single-step approach using marginal values are used to solve the problem.
- Chapter 15. “A Frequency Allocation Problem,” concerns the *assignment* of frequency intervals to links in a communication system while *minimizing total interference*. Two formulations are discussed in detail.
- Chapter 16. “A Power System Expansion Problem,” comes from the electric power industry and provides an overview of several approaches to uncertainty, namely *what-if analysis, two-stage stochastic programming* and *robust optimization*.
- Chapter 17. “An Inventory Control Problem,” comes from an application for storing enough beer to meet uncertain demand. The demand is time dependent, and the model is formulated as a *multi-stage stochastic model*.

Part V—Advanced Optimization Modeling Applications—provides applications where mathematical concepts are required for the formulation and solution of the underlying models.

*Advanced
optimization
modeling
applications*

- Chapter 18. “A Portfolio Selection Problem,” comes from a financial application in which both *strategic* and *tactical investment* decisions are made. One-side variance is used as a measure of *portfolio risk*.
- Chapter 19. “A File Merge Problem,” comes from a database application in which two large *statistical data files* are *merged* into one single file. The corresponding model is formulated as a network model for which *columns are evaluated* as needed.
- Chapter 20. “A Cutting Stock Problem,” comes from an application in which large raws of paper and textile are sliced into patterns. *Patterns are generated* by an auxiliary model during the solution process, which makes this chapter a first introduction to *column generation*.
- Chapter 21. “A Telecommunication Network Problem,” comes from an application in telecommunication network design with a focus on *bot-*

tleneck capacity identification. Again, a *column generation* technique is used to generate paths through the network.

- Chapter 22. “A Facility Location Problem,” comes from a multi-commodity transportation application and implements a *Benders’ decomposition algorithm* to solve the problem.

Before you begin, you should be familiar with mathematical notation. This should be sufficient to read Parts I and III. Basic linear algebra and probability analysis is required for Parts II and IV. An introductory course in mathematical programming is recommended before reading the advanced chapters in Part V.

Preliminaries

The authors

Johannes Bisschop received his Ph.D. in Mathematical Sciences from the Johns Hopkins University in Baltimore USA in 1974. From 1975 to 1980 he worked as a Researcher in the Development Research Center of the World Bank in Washington DC, USA. In 1980 he returned to The Netherlands and accepted a position as a Research Mathematician at Shell Research in Amsterdam. After some years he also accepted a second part-time position as a full professor in the Applied Mathematics Department at the Technical University of Twente. From 1989 to 2003 he combined his part-time position at the University with managing Paragon Decision Technology B.V. and the continuing development of AIMMS. From 2003 to 2005 he held the position of president of Paragon Decision Technology B.V. His main interests are in the areas of computational optimization and modeling.

*Johannes
Bisschop*

In addition to the main authors, various current and former employees of Paragon Decision Technology B.V. and external consultants have made a contribution to the AIMMS documentation. They are (in alphabetical order):

*Other contribu-
tors to AIMMS*

- | | |
|-----------------------------|-----------------------|
| ■ Pim Beers | ■ Gertjan Kloosterman |
| ■ John Boers | ■ Joris Koster |
| ■ Peter Bonsma | ■ Chris Kuip |
| ■ Mischa Bronstring | ■ Gertjan de Lange |
| ■ Ximena Cerda Salzmann | ■ Ovidiu Listes |
| ■ Michelle Chamalaun | ■ Bianca Makkink |
| ■ Robert Entriiken | ■ Peter Nieuwesteeg |
| ■ Thorsten Gragert | ■ Giles Stacey |
| ■ Koos Heerink | ■ Richard Stegeman |
| ■ Nico van den Hijligenberg | ■ Selvy Suwanto |
| ■ Marcel Hunting | ■ Jacques de Swart |
| ■ Roel Janssen | ■ Martine Uytterlinde |

Part I

Introduction to Optimization Modeling

Chapter 1

Background

This chapter gives a basic introduction to various aspects of modeling. Different types of models are distinguished, some reasons for using models are listed, some typical application areas of modeling are mentioned, and the roles of mathematics and AIMMS are described. Finally, an overview is given of the “model formulation process.”

This chapter

1.1 What is a model?

Since this guide describes different models and modeling techniques, it will start by defining what a model is. In general: *a model is a prototype of something that is real*. Such a prototype can be concrete or abstract.

A model

An example of a concrete model is a teddy bear. A teddy bear is concrete—you can touch it and play with it—and it is also a scaled-down (and somewhat friendlier) version of a real bear.

Concrete model

To illustrate an abstract model, the Teddy Bear Company is introduced. This company produces black and brown teddy bears in three sizes, and its owners consider the teddy bear in terms of an *abstract* model. That is, they describe everything they need to know about producing it:

Abstract model

- materials: fur cloth in black and brown, thread, buttons, different qualities of foam to stuff the bears,
- information on prices and suppliers of materials,
- three different sized patterns for cutting the fur, and
- assembly instructions for the three sizes of bears.

In this abstract model of the teddy bear, there is just the basic information about the bear. A *mathematical model* can be used to derive further information.

Suppose you want to know the cost of a small black teddy bear. Then you sum the material costs (material prices by the amounts used) and the production costs (assembly time by labor cost). The relationship between all these components can be expressed in general terms. Once formulated, you have developed a mathematical model. A mathematical model is an abstract model. It is a description of some part of the real world expressed in the language of mathematics. There may be some variables in the description—values that are unknown. If the model is formulated so that you can calculate these values (such as the cost of a small black teddy bear) then you can *solve* the model.

Mathematical models

One class of mathematical models is referred to as *optimization models*. These are the main subject of this guide. For now, a small example will be given. Recall that the teddy bears are filled with foam which comes in small pieces and is available in different qualities. By choosing a certain mix you can achieve a specified “softness” for each size of teddy bear. The company wants to decide how much to buy of each quality. A constrained optimization model could be used to determine the cheapest mix of foams to yield a certain softness. Note that the *optimization* is in the determination of the *cheapest* combination (cost minimization), and that the constraint(s) are in terms of the softness requirement.

Optimization models

1.2 Why use models?

In Section 1.1 two types of models were introduced and some typical modeling applications given. Advantages of using models are now listed.

This section

A model serves as a learning tool. In the process of constructing a model you are forced to concentrate on all the elements required to realize the model. In this way, you learn about the relationships between the elements that make up the model. For instance, during the design of a teddy bear you must consider the outside—what is the shape of the ears?—as well as the inside—the foam to choose. During the process, you may discover an unexpected relationship between the softness of the bear and its assembly.

Learning while constructing

Depending on the purpose of the model you then need to set priorities since mathematical models can grow in complexity with the quantity of detail. If the objective of the model is to determine the cheapest mix of foams, you may wish to ignore the softness of the ears. Such filtering and structuring of information is an important contribution to the modeling process.

Filtering information

Models can also be a valuable tool of expression during discussions between concerned parties. If the formulation and data of a model are agreed, it can be used to make predictions of the consequences of different actions, which can then be compared.

Medium of discussion

Some decisions are irreversible or expensive. In these cases, it is beneficial to be able to predict the consequences and to experiment with different options. Again models can help. The Teddy Bear Company may use the optimization model to calculate the cost of different foam mixes while varying the softness requirements. Perhaps a slightly softer bear turns out to be considerably cheaper.

Making predictions

For the Teddy Bear Company it might be possible to cut the foam cost further by buying it from several suppliers and monitoring price variations. The optimization model could easily be extended to incorporate this information. It will enable the Teddy Bear Company to react quickly to changes in foam prices.

Cutting costs to stay competitive

The combination of models and computers enhances the speed of decision making. The computer enables you to handle far more information than could be done manually.

With computers

Another benefit possible from modeling is called *streamlining*. That is, in the process of constructing a model and structuring its information you may discover inefficiencies which can then be addressed. For example, consider a consulting company, while making a model of the Teddy Bear Company as a whole, discovers that there are two business units performing virtually the same task. Suppose that for some historical reason, there are separate units producing black and brown teddy bears. Since the only difference is in the color of the cloth, integrating these business units would make the company more efficient.

Streamlining

1.3 The role of mathematics

Even though this guide concerns the specification of *mathematical* programs (optimization models), you do not have to be a mathematician to become a good modeler. The degree of mathematics required depends upon the sort of model you are building. In this book there is a distinction between basic, intermediate and advanced models such that each type requires an increase in knowledge of mathematical theory and concepts. When considering the role of mathematics, one can distinguish three aspects, namely, language, theory and algorithms. These aspects are discussed in the following paragraphs.

Mathematical requirements

Mathematics provides a language in which abstract models can be formulated using the following elements.

Mathematical language

- Mathematical concepts such as variables (unknowns) and parameters (symbols representing known data);
- Operators such as:
 - unary operators (+, −, NOT),
 - comparison operators (equal to, not equal to, etc.),
 - algebraic operators (addition, subtraction, multiplication, division, power, etc.),
 - logical operators (AND, OR, etc.),
 - differential operators and integral operators;
- Data: which links a model to a real-world situation.

Mathematics provides a theoretical framework for the formulation and solution of those models for which the translation from problem to model is not immediate. Mathematical concepts are then needed to capture aspects of tangible and non-tangible real-life situations. A non-tangible example is the risk minimization in a financial portfolio selection model. There is no hard definition of *risk* but there are plenty of interpretations. When building a model one or more of these interpretations must be made concrete in the form of a mathematical formula for risk. This can only be achieved by using mathematical concepts and their associated theory.

Mathematical theory

Mathematical algorithms are needed to obtain solutions of mathematical models. Fortunately, there are algorithms that have wide spread application and give solutions for whole classes of mathematical models. As a model builder you can choose not to acquire an in-depth knowledge of these algorithms as they are commercially available and directly accessible within a modeling system. Nevertheless, there are instances of large-scale models for which standard solution algorithms no longer suffice, and special algorithms need to be developed. Examples of these can be found in Part 21 containing advanced optimization modeling applications.

Mathematical algorithms

The AIMMS modeling system is especially designed to minimize the mathematical complexity.

AIMMS minimizes need for mathematics

- The AIMMS modeling language incorporates indexing facilities to guide you in expressing and verifying complex mathematical models.
- Since there is a separation between the formulation and the *solving* of a model, you do not need to know the details of the solution method. If you do not specify a solution method, AIMMS will determine the best method for you.

1.4 The modeling process

The process of developing a model usually involves several different activities. Although these activities are listed sequentially below, several will be repeated as new information becomes available during the execution of other steps, thus an *iterative* approach will often be necessary.

Iteration of activities

- Define the goal.
- Consult literature and other people.
- Formulate the model, and collect the data.
- Initial testing.
- Validation.

In a complex problem, the structure and goal of the model may not be obvious. Therefore, the first step should be to analyze the general problem conceptually and to determine which aspects of the real-world situation must be included. At this stage the emphasis is on problem definition, rather than on mathematics. It is likely there are different views on the problem and its causes, and these must be resolved so that all concerned parties agree on the goal of the model. In one sense, you can say that the goal of a model is to support a decision. In a wider sense, a model can be used to *predict* the consequences of particular decisions, and so to compare alternatives.

Investigating contribution of a model

Once the goal of the model is agreed, the next step is to investigate if a similar model has already been developed. There are benefits to review the work of others. It may help you to find the *data* you need and it may give you some hints on how to formulate your model. However, in general you can expect to do some customization.

Investigating sources

In most cases, by far the harder task is collecting the data. This task is both time consuming and prone to errors. Most models, built to analyze real-world problems, require lots of data. In some instances data is available through databases. In other instances data may not be readily available, and once it is found, it is seldom in a form directly suitable for your purposes. In this case data must be copied, checked for errors, and manipulated. During this *data adjustment phase*, errors of different types may be introduced. Examples are approximation errors, typographical errors, and consistency errors. Many typographical errors and inconsistencies in data can be identified by AIMMS, and will result in error messages when compiled.

Data collection

It is not usually self-evident which particular type of model formulation is the most appropriate. There are no clear rules for making a choice. As mentioned in the previous paragraph, the choices of other modelers in similar situations can give suggestions, and your own experience and knowledge will always influence your eventual choice. In any case, when you want to stay within the framework of the available modeling tools, you may have to compromise.

Choosing a formulation

It is recommended that you start with a small model containing only the basic relationships. You should verify that these are formulated correctly before adding increased complexity, and continue in this gradual fashion.

Initial testing

Validation is the process of checking whether initial model results agree with known situations. It is an important final step before the model results are used in support of a real-world decision. Two situations may occur:

Validation

- A methodology already exists with the same purpose as the new model.

In order to be credible, you will have to prove that the results of the new model are *at least* as good as the results produced with an existing method. Since the “old method” is probably known to be accurate (or its weaknesses are known), you can compare the old and new methods side by side. Your model should at a minimum reproduce the old results, and you should aim for better results.

- There is no existing methodology.

In this case, you should use historical data, and try to reproduce the past. In essence, you try to predict what you already know.

1.5 Application areas

This section gives an indication of the applicability of optimization models in practice. Obviously, these few pages do not pretend to give a complete overview. Rather the goal is to combine a limited overview with a description of a typical example taken from each application area mentioned. These descriptions are qualitative in that specific data and formulas are not given. The focus is on the general structure of models.

This section

1.5.1 Production

In the petroleum industry *refinery models* are used for planning, scheduling and process control, see for instance [Ma56]. Following is a brief account of a refinery planning model.

The petroleum industry

Planning models support long-term decisions, concerning raw material purchases, and utilization of production processes, in order to meet future demands for final products with specific qualities. These quality specifications require *blending models* to be embedded in refinery models, see for instance [Ke81]. A refinery planning model describes the process from crude oils to final products such as different blends of gasoline, fuel oil, and distillate. The production process makes use of various production units including crude distillers, catalytic crackers, and vacuum distillers.

*Refinery
planning model*

This qualitative model description is based on a model as described in [Ke81].

*Refinery model
description*

Maximize: $\text{profit} = \text{revenue} - \text{material cost} - \text{operating cost}$,

Subject to:

- for all crude oils: a constraint stating that the amount of crudes used in processes must not exceed the amount of crudes purchased,
- for all intermediates: a constraint stating that the net production and purchases of an intermediate commodity must exceed its use in blending,
- for each production unit: a capacity constraint,
- for all crude oils: limits on purchases,
- for all final products: a constraint stating that final sales must equal the output of the blending process, and
- for all final products, and all quality attributes: upper and lower bounds on the quality of final products.

The application area that studies the efficient production and distribution of energy is known as *energy economics*. This area is of strategic importance to modern economies because of the world's dependence on energy, and in particular oil. The conversion of scarce energy resources to electricity and heat can be modeled, see for instance [Ma76]. From such a model, the least-cost mix including extraction of energy resources, and investment in production technologies, can be determined. Moreover, constraints on emission levels for certain pollutants and the use of abatement technologies can be included. Another application is *scheduling power plants* to meet the varying load at different times of the day, and in different seasons. An example is given in Chapter 16.

*Other
applications*

1.5.2 Production and inventory management

The scheduling of production and inventory can be formulated as a linear optimization model. Typically, multiple time periods and material balances are incorporated in such a model. Suppose factory production must satisfy a fluctuating demand. There are many strategies for reacting to these fluctuations:

*Typical model
components*

extra workers can be hired and fired, workers can work overtime, a stock inventory can be kept to cover future shortages, etc. Obviously, the hiring and firing of workers or overtime working adds extra costs. Storage however is also costly.

A verbal formulation of the production and inventory problem, which can be found in [Ch83], follows.

An example

Minimize: costs of storage, hiring, firing, and overtime work,

Subject to:

- for each time period: upper bounds on overtime work and changes in work force level, and
- for each time period: a material balance involving production, inventory, and demand.

Other useful references on inventory planning are [Ch55] and [Ha60].

Linear optimization models are quite successful for *resource allocation* problems. The potato chips problem in Chapter 2 is a simplified example. Different products compete for a finite set of resources, and decisions may possibly be made about the selection of a production process for a particular product. Another widespread application of mathematical optimization models is in developing *production schedules*. An overview of techniques is given in [Bu89].

Other applications

1.5.3 Finance

A widely-used application in the finance field is the selection of *investment portfolios*. A portfolio is a collection of securities held by an investor, such as a pension fund. Diversification is the spreading of investments over different securities with the aim to reduce the overall risk content of a portfolio while still maintaining a respectable return. One of the original models is due to Markowitz ([Ma52]). An investor wants the expected return of the portfolio to be high, and the risk to be low. The risk can be measured using the statistical concept *variance*—the deviation of realized return from the expected return. Furthermore, the *covariance* between two securities is a measure of correlation. A covariance matrix of returns summarizes variance and covariance figures, and can be derived from historical data. It can be used to compute the risk attached to the objective.

Portfolio selection

Minimize: risk,

Subject to:

- a constraint on minimum expected return,
- a limitation on the budget that can be spent, and
- additional constraints concerning the proportions invested in certain securities.

An example

The resulting model has a quadratic objective and linear constraints, and is known as a *quadratic optimization model*. In Chapter 18, a more elaborate treatment of portfolio models is given. In real situations, the size of the covariance matrix can be overwhelming. For example, 100 possible securities give rise to a covariance matrix of $100 \times 100 = 10,000$ entries. Alternative methods have been developed to compress this information.

Optimization models can be used to plan the cash receipts and disbursements of an individual firm. These *cash management models* can sometimes be formulated as networks in which the nodes represent different time periods. The arcs then represent flows of money over time. Optimization models can also be used for the management of large funds. Such a fund requires careful planning since commitments made today have long term financial consequences. A multi-period model considers, for each period, loans, repayments, and investments, subject to financial and political constraints. In *accountancy*, linear optimization models are often used because of the useful economic information that can be derived from shadow prices and reduced costs. A thorough account of the use of optimization models in finance is given by in [Ko87].

*Other
applications*

1.5.4 Manpower planning

Linear and integer optimization models can be used to schedule work shifts when the demand for workers varies over the hours of the day, or the days of the week. The decision to be made is how many people should work during each shift, see for instance [Sc91] and [Ch68b]. Examples are the scheduling of bus drivers, nurses, and airline crews. A general framework for the solution process is as follows:

*Scheduling
work shifts*

1. Identify or make a forecast of the personnel requirements for the period to be scheduled.
2. Evaluate all possible shift patterns. All these patterns have costs associated with them, depending on efficiency, wasted time etc.
3. Determine the least cost combination of shifts that satisfies the personnel requirements.

An optimization model is used for the final step. Obviously, the second step can also be computerized. In large problems this will be essential. Advanced methods exist in which promising shift patterns are generated using shadow prices produced by the simplex method. An iterative process between the Steps 2 and 3 evolves. Such a technique is known as *delayed column generation*, see for instance [An91] and [Ch83].

A slightly different approach is taken in the following problem. Suppose a bus company employs drivers who are entitled to two consecutive days off per week. For each day of the week the number of drivers required is known. The problem is: “How many drivers should start their five-day working week on each day of the week?” The qualitative model formulation can now be given:

A model for scheduling bus drivers

Minimize: *total number of drivers,*

Subject to:

for each day: the driver requirements must be satisfied.

The formulation of the constraint is a special task, because for each day the number of drivers that are part way through their shift must also be calculated. Complicating factors might be higher costs for working during weekends, or the availability of part-time workers.

Longer time-scale models exist which consider personnel as *human capital*, whose value increases as experience grows, training followed, etc., see for instance [Th92]. Such a model can be formulated as a *network-with-gains* model. The nodes represent moments in time, and the flow between these nodes represents the flow of employees whose value varies in time due to training, resignations, and new hiring.

Other applications

1.5.5 Agricultural economics

Models of both a country’s agricultural sector and of individual farms have been used extensively by governments of developing countries for such purposes as assessing new technologies for farmers, deciding on irrigation investments, and determining efficient planting schedules to produce export-goods. A short account of an agricultural *subsector model*, see for instance [Ku88], is given here, Chapter 11 has a more extensive account.

Individual farm models

The subsector model combines the activities, constraints, and objective functions of a group of farms. The main activity of a farm is growing different crops and perhaps keeping livestock. Typical constraints restrict the availability of land and labor. Agricultural models become multi-period models as different growing seasons are incorporated.

A subsector model

The simplest type of agricultural subsector model is one comprised of a number of similar farms which compete for a common resource or share a common market. By “similar” is meant that they share the same technology in terms of crop possibilities, labor requirements, yields, prices, etc. Such subsector models can be used to verify if common resources are really a limiting factor at a regional level. Hired labor, for instance, might well be a constraint at the regional level.

Similar farms

A different situation arises when there are fundamental differences between some of the farms in the subsector to be modeled. One common difference is in size, which means that resource availabilities are no longer similar. In the following example, it is assumed that the types of farms are distinguished mostly by size.

Farm types

Maximize:

regional income = income aggregated over different farm types,

An example

Subject to:

- *for each type of farm, for each time period: a restriction on the availability of resources (land, labor, water) needed for cropping activities, and*
- *for each time period: a regional labor restriction on the availability of hired workers (temporary plus permanent).*

Other farming applications include blending models for blending cattle feed or fertilizers at minimum cost. Distribution problems may arise from the distribution of crops or milk. Large models can be used for the management of water for irrigation, or to examine the role of livestock in an economy. Furthermore, these principles can be applied with equal success to other economic subsectors.

Other applications

1.6 Summary

This chapter discussed the basic concepts of modeling. Some of the main questions addressed in this chapter are: *What is a mathematic model, why use mathematical models, and what is the role of mathematics during the construction of models.* The modeling process has been introduced as an iterative process with the steps to be executed described in detail. The latter portion of this chapter introduced several application areas in which optimization models can be used.

Chapter 2

Formulating Optimization Models

This chapter explains the general structure of optimization models, and some characteristics of their solution. In Chapter 1, an introduction to optimization models was given. In this chapter, optimization models are introduced at a more technical level.

This chapter

The three main classes of constrained optimization models are known as *linear*, *integer*, and *nonlinear programming models*. These types have much in common. They share the same general structure of optimization with restrictions. Linear programming is the simplest of the three. As the name indicates, a linear programming model only consists of linear expressions. Initially, linear programming will be explained, followed by integer and nonlinear programming.

Three classes of constrained optimization models

The term *programming*, as used here, does not denote a particular type of computer programming, but is synonymous with the word *planning*. The three classes of programming models mentioned above all come under the heading of *mathematical programming models*.

The term programming

2.1 Formulating linear programming models

Linear programming was developed at the beginning of the mathematical programming era, and is still the most widely used type of constrained optimization model. This is due to the existence of extensive theory, the availability of efficient solution methods, and the applicability of linear programming to many practical problems.

Wide applicability

2.1.1 Introduction

Basic building blocks of linear programming models are *linear equations* and *linear inequalities* in one or more unknowns. These are used in linear programming models to describe a great number of practical applications. An example of a linear equation is:

Linear equations and inequalities

$$2x + 3y = 8$$

By changing the “=” sign to a “≥” or “≤”, this equation becomes a linear inequality, for instance:

$$2x + 3y \geq 8$$

The signs “<” and “>”, denoting strict inequalities, are not used in linear programming models. The *linearity* of these equations and inequalities is characterized by the restriction of employing only “+” and “−” operations on the terms (where a term is defined as a coefficient times a variable) and no power terms.

The unknowns are referred to as *variables*. In the example above the variables are x and y . A solution of a linear programming model consists of a set of values for the variables, consistent with the linear inequalities and/or equations. Possible solutions for the linear equation above are, among others: $(x, y) = (1, 2)$ and $(4, 0)$.

Variables

2.1.2 Example of a linear programming model

To illustrate a linear programming model, the production of chips by a small company will be studied. The company produces plain and Mexican chips which have different shapes. Both kinds of potato chips must go through three main processes, namely slicing, frying, and packing. These processes have the following time characteristics:

*Production of
potato chips*

- Mexican chips are sliced with a serrate knife, which takes more time than slicing plain chips.
- Frying Mexican chips also takes more time than frying plain chips because of their shape.
- The packing process is faster for Mexican chips because these are only sold in one kind of bag, while plain chips are sold in both family-bags and smaller ones.

There is a limit on the amount of time available for each process because the necessary equipment is also used for other purposes. The chips also have different contributions to net profit.

The data is specified in Table 2.1. In this simplified example it is assumed that the market can absorb all the chips at the fixed price.

Data

The planner of the company now has to determine a production plan that yields maximum net profit, while not violating the constraints described above.

time [min/kg] required for:	plain chips	Mexican chips	availability [min]
slicing	2	4	345
frying	4	5	480
packing	4	2	330
net profit contribution [\$ /kg]	2	1.5	

Table 2.1: Data in the potato chips problem

The planner's decision problem can be formulated in terms of a mathematical notation using linear inequalities. The variables in the inequalities must reflect what is unknown to the planner, namely his decisions. In this example, the decision variables concern a production plan. The quantity of plain and Mexican chips to be produced are unknown to the planner. Therefore, the variables are the *amounts of both types of chips to be produced*.

*Decision
variables*

In order to obtain a concise mathematical description it is convenient to choose short names for the variables. Let X_p therefore denote the unknown amount of plain chips to be produced, and let X_m denote the unknown quantity of Mexican chips to be produced. X_p and X_m are both measured in kilograms. Inequalities that reflect the availability of the production processes can now be stated.

Variable names

The following inequality, measured in minutes, can be written to describe the limited availability of the fryer:

*The constraint
on frying*

$$4X_p + 5X_m \leq 480 \quad [\text{min}]$$

In words this inequality states:

*The four minutes required to fry a kilogram of plain chips
multiplied by the planned number of kilograms of plain chips
plus
the five minutes required to fry a kilogram of Mexican chips
multiplied by the planned number of kilograms of Mexican chips
must be less than or equal to
the 480 minutes the fryer is available.*

Or, a bit shorter:

*The time required to fry the plain chips
plus
the time required to fry the Mexican chips
must be less than or equal to
the time the fryer is available.*

So now there is an inequality that describes the limited availability of the fryer.

An easy check to see whether the meaning of an inequality makes sense is to write it in terms of units of measure. This yields:

*Units of
measure*

$$4[\text{min/kg}]X_p[\text{kg}] + 5[\text{min/kg}]X_m[\text{kg}] \leq 480[\text{min}]$$

The resulting units for each term should be identical, which they are in this case (minutes).

Similar inequalities can also be written for the availabilities of the slicer and the packer:

*Other
constraints*

$$2X_p + 4X_m \leq 345 \quad [\text{min}]$$

$$4X_p + 2X_m \leq 330 \quad [\text{min}]$$

Together these inequalities almost give a complete description of the situation. One set of inequalities is still missing. Obviously, it is not possible to produce a negative amount of chips. So, the following lower bounds on the variables must be added for a complete description of the problem:

$$X_p \geq 0, \quad X_m \geq 0 \quad [\text{kg}]$$

These last inequalities are referred to as *nonnegativity constraints*.

The company's planner has to make a choice. From these possible production options, he wants to choose the plan that yields the maximum net profit. By maximizing profit, the number of plans is reduced to those that are preferred. The following linear equation gives the net profit:

*Optimal
decisions*

$$P = 2X_p + 1.5X_m \quad [\$]$$

The quantity P can be regarded as an additional variable, for which the maximum value is to be found. The value of P depends on the value of the other variables.

The decision problem has just been posed as a mathematical problem instead of a verbal problem. In order to show similarities and differences between them, both a verbal description of the problem and the mathematical model are given. The *verbal* description of the decision problem is:

*Verbal
summary*

Maximize: *Net profit,*

Subject to:

- *a time restriction on slicing,*
- *a time restriction on frying,*
- *a time restriction on packing, and*
- *negative amounts cannot be produced.*

The *mathematical* model is formulated as follows:

*Mathematical
summary*

$$\begin{array}{ll}
 \text{Maximize:} & P = 2X_p + 1.5X_m \\
 \text{Subject to:} & 2X_p + 4X_m \leq 345 \quad (\text{slicing}) \\
 & 4X_p + 5X_m \leq 480 \quad (\text{frying}) \\
 & 4X_p + 2X_m \leq 330 \quad (\text{packing}) \\
 & X_p, X_m \geq 0
 \end{array}$$

2.1.3 Picturing the formulation and the solution

In this small problem the inequalities can be visualized in a two-dimensional drawing. Where the x -axis and the y -axis represent X_p and X_m respectively, the inequalities and their implications can be plotted. The easiest way to plot the slicer availability inequality is as follows.

*Picturing
the decision
problem*

- First change the “ \leq ” sign to an “ $=$ ” and plot the border.

Setting the value of X_p to 0, then the value of X_m can be calculated: $X_m = 345/4 = 86.25$. In the same way the value of X_p is calculated as 172.5 when X_m is set to zero.

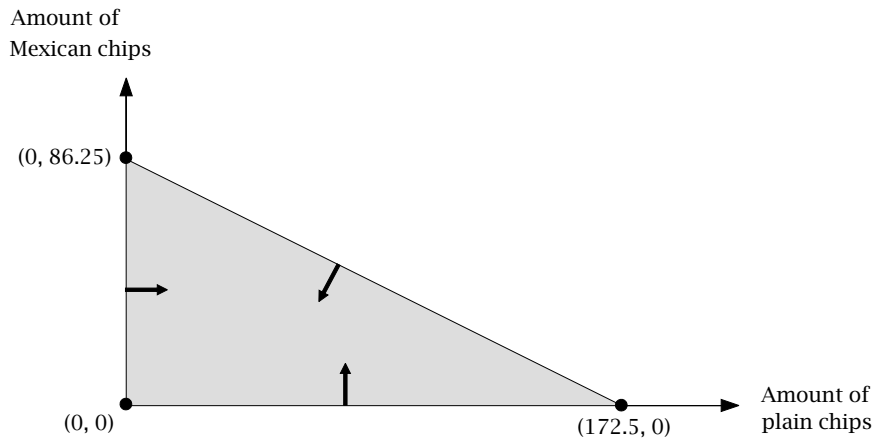


Figure 2.1: The constraint on slicing visualized

So far, two points have been found in the X_p - X_m plane, namely $(X_p, X_m) = (0, 86.25)$ and $(X_p, X_m) = (172.5, 0)$. The line that connects these points is the line

$$2X_p + 4X_m = 345$$

which is plotted.

- Second, determine whether a single point at one side of the line, such as the origin, satisfies the constraint. If it does, shade that side of the line.

The shaded region in Figure 2.1 contains all (X_p, X_m) that satisfy the constraints:

$$2X_p + 4X_m \leq 345, \quad X_p \geq 0 \text{ and } X_m \geq 0$$

In other words, the shaded region contains all combinations of the quantities of the two types of chips that can be sliced in one day.

Other inequalities can also be represented graphically, as shown in Figure 2.2. From this figure, it is clear that there are many combinations of production levels for plain and Mexican chips that satisfy all these inequalities. The shaded region bounded by the lines corresponding to the inequalities represents all the allowed production levels, and is called the *feasible region*.

*Picturing
the feasible
region*

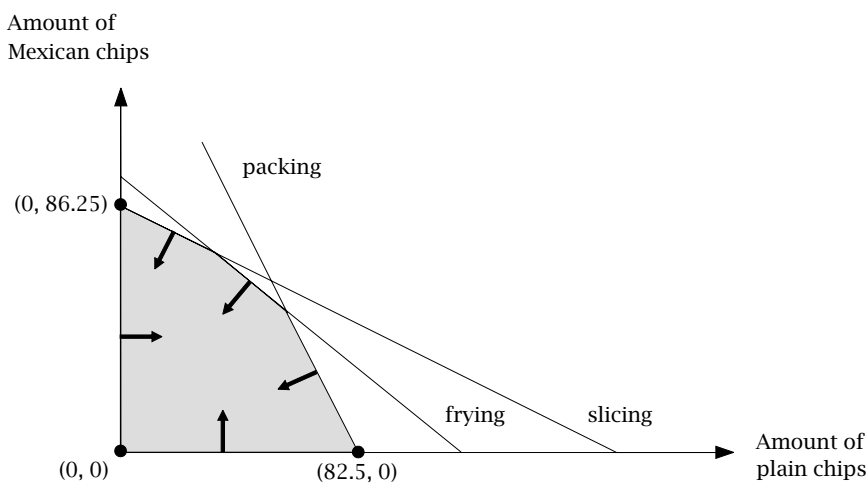


Figure 2.2: The feasible region

When a value, of say \$150, is chosen for the variable P , a line can be drawn that represents all combinations of production levels for Mexican and plain chips that yield a profit of \$150. Such a line is called a *contour* of the profit function, and is drawn in Figure 2.3. The arrow indicates the direction of increasing profit. Since the profit function is linear, the contours are straight lines.

*Picturing
the profit
function*

But how can one determine which combination yields the *maximum* net profit? Observe that a higher value for P yields a contour parallel to the previous one. Moreover, increasing the value of P causes the line to shift to the right. This is also illustrated in Figure 2.3. However, the profit cannot increase indefinitely, because the profit line will fall outside the feasible region if it shifts too far to the right. In fact, it is clear from the application that the profit cannot increase indefinitely because of the limitations imposed by the availability of the slicer, fryer, and packer.

*Picturing
the optimal
decision*

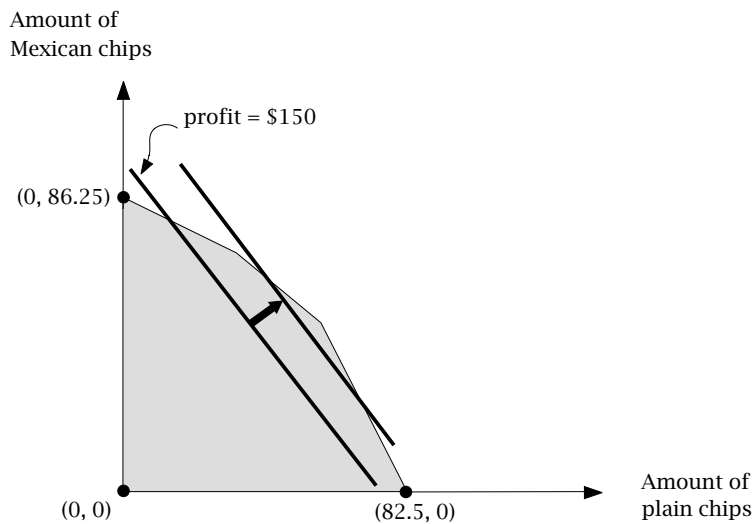


Figure 2.3: Different profit lines

The best solution attainable is when the profit line is shifted *as far to the right as possible* while still touching the feasible region. *Best solution*

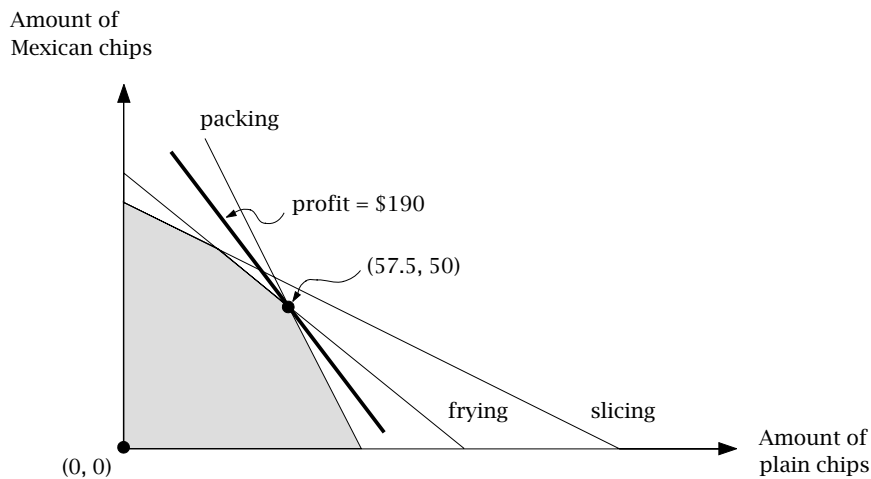


Figure 2.4: The optimal solution

From Figure 2.4, it can be seen that this point is the intersection of the lines corresponding to the frying and packing restrictions. The coordinates of this point can now be calculated by solving a system of two equations in two unknowns—the frying and packing restrictions as equalities:

$$\begin{aligned} 4X_p + 5X_m &= 480 && \text{(frying)} \\ 4X_p + 2X_m &= 330 && \text{(packing)} \end{aligned}$$

The above system yields $(X_p, X_m) = (57.5, 50)$ and the corresponding profit is \$190. This combination of values for the decision variables is called the *optimal solution*.

Considering Figure 2.4 once again, it can be observed that only two constraints really restrict the optimal solution. Only the constraints on frying and packing are *binding*. The constraint on slicing can be omitted without changing the optimal solution. Such a constraint is known as a *non-binding* constraint. Although non-binding constraints can be removed from the model without consequences, it is often sensible to include them anyway. A non-binding constraint could become binding as data change, or when experiments are carried out using the model. Moreover, when you build a model, you probably will not know in advance which constraints are non-binding. It is therefore better to include all known constraints.

Non-binding constraints

Considering Figure 2.4 once again, one can see that the optimal solution is on a corner of the feasible region, namely, the intersection of two lines. This implies that the exact value can be calculated by solving a system of two equations and two unknowns. In general, it can be stated that if a linear programming model has an optimal solution, then there is always an optimal corner solution. This is illustrated in Figure 2.5. Depending on the slope of the objective function, the solution is either at A, B, or C.

Corner solutions

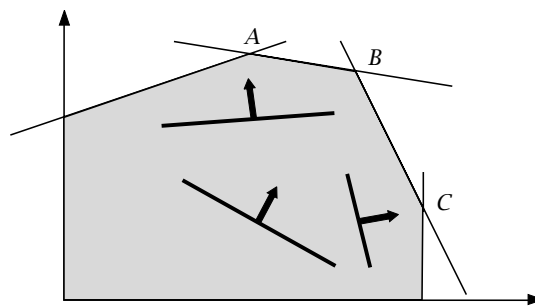


Figure 2.5: There is always an optimal corner solution

A special case occurs when the slope of the objective is parallel to the slope of one of the binding constraints, as in Figure 2.6. Then there are two optimal corner solutions and an infinite number of optimal solutions along the line segment connecting these two corner solutions. This is a case of so-called *multiple* or *alternative optima*.

Multiple optima

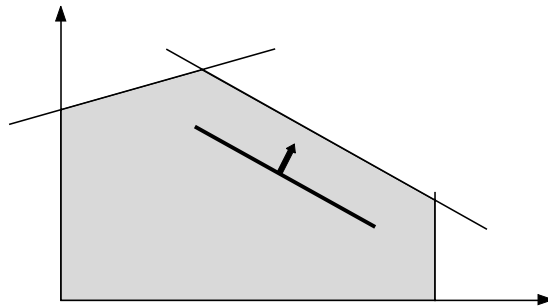


Figure 2.6: Multiple optima

The graphical solution method so far used in this section is only suitable for problems with two decision variables. When there are three decision variables, a similar three-dimensional figure evolves, but this is a lot harder to draw and interpret. Figures with more than three dimensions, corresponding to the number of decision variables, cannot be drawn. The optimal solution must then be expressed algebraically, and solved numerically. The graphical solution method is only used for purposes of illustration.

Limitations of pictures

The method most often used to calculate an optimal solution is the so-called *simplex method*, developed by George Dantzig ([Da63]). This method examines the corners of the feasible region in a structured sequence, and stops when the optimal solution has been found. For very small models, these calculations could be done manually, but this is both time consuming and prone to errors. In general, these calculations are best done by a computer, using a sophisticated implementation of the simplex method. Almost without exception, such an algorithm finds an optimal solution, or concludes that no such solution exists.

Computer solutions

When an optimal solution is found by a solver, caution is still needed. Since a model is a simplified picture of the real problem, there may be aspects that are neglected by the model but still influence the *practical* optimality of the solution. Moreover, for most situations there is no single, excellent from all aspects, optimal solution, but a few different ones, each one optimal in its own way. Therefore, the practical interpretation of the model results should always be considered carefully. Experiments can be done using different objectives.

Optimality

In Chapter 4, an introduction is given to *sensitivity analysis*—solution changes due to changes in the data.

When a problem is declared to be *infeasible* by a solver, it means that the feasible region is empty. In other words, there is no solution that satisfies all the constraints simultaneously. This is illustrated in Figure 2.7. *Infeasibility* can be caused by having too many or conflicting requirements, or by errors in data specification, or by errors in the construction of model equations. Such a result is an incentive to check the correctness of the model. Sometimes an infeasible solution is practically acceptable because the constraints that are violated are not so critical.

Infeasibility

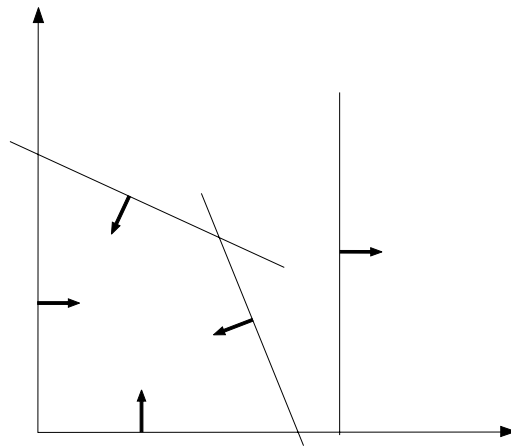


Figure 2.7: Infeasibility illustrated

Unboundedness is just what the word implies; the constraints fail to bound the feasible region in the direction in which the objective is optimized. As a result, the value of the objective function increases or decreases indefinitely, which might look attractive, but is certainly not realistic. In Figure 2.8, a graphical illustration is given. Unboundedness is not a problem occurring in reality but a formulation error. Common errors causing unboundedness include the following. It could be that a free variable should be nonnegative, or that the direction of optimization has been inadvertently reversed. Alternatively, a constraint may have been omitted. Note that whether an unbounded constraint set causes difficulties depends on the objective function. In Figure 2.9 an example is given in which an optimal solution does exist, although the feasible region is unbounded.

Unboundedness

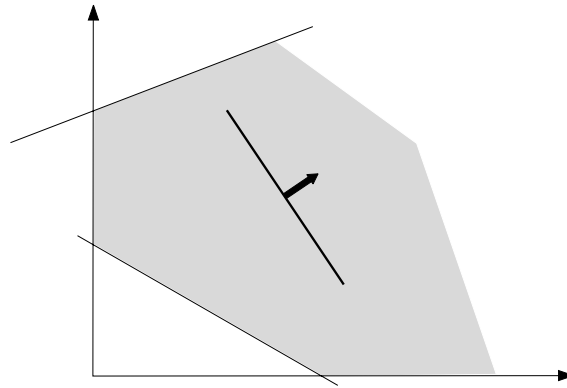


Figure 2.8: Unbounded objective, unbounded feasible region

Readers who want to know more about the theoretical aspects of linear programming, or the simplex method, can refer to [Ch83] or [Lu73], and there are many other excellent works on the subject.

References

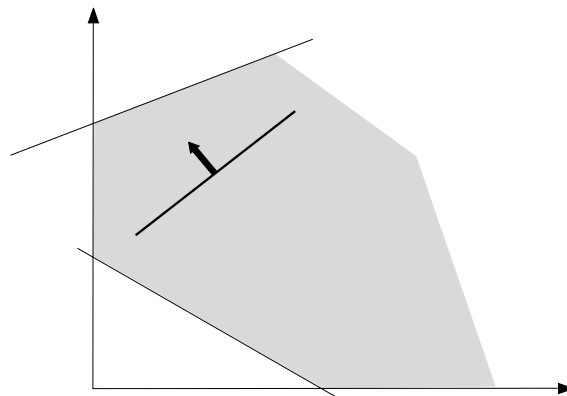


Figure 2.9: Bounded objective, unbounded feasible region

2.2 Formulating mixed integer programming models

In this section some basic characteristics of mixed integer programming models and their solution will be discussed.

This section

Recall the example of the chips-producing company in the previous section where the optimal solution was $(X_p, X_m) = (57.5, 50)$. If chips are packed in bags of 1 kilogram, this optimal solution cannot be translated into a practical

Need for integer solutions ...

solution. One approach is to round down the solution. In this example, rounding yields the feasible solution $(X_p, X_m) = (57, 50)$, but the profit is reduced from \$190 to \$189.

There are many real-world problems that require their solutions to have integer values. Examples are problems that involve equipment utilization, setup costs, batch sizes, and “yes-no” decisions. Fractional solutions of these problems do not make real-world sense; just imagine constructing half a building.

... illustrated

Rounding is a straightforward way to obtain *integer values*. There might be doubt, however, whether such a solution is optimal or even feasible. In practice, rounding is a satisfactory approach when:

Pros and cons of rounding ...

- the figures to be rounded are so large that the error is negligible,
- the input data are not known with certainty, so that a fractional solution is not that accurate anyway,
- the rounding procedure is certain to give a feasible solution, and
- algorithms developed especially to search for an integer solution are too expensive in terms of computer resources and time.

The alternative to rounding has already been mentioned—making use of an *integer programming algorithm*. Next, the potato chips example will be used to present some aspects of integer programming. When an integer programming algorithm is used in the example above, the solution $(X_p, X_m) = (49, 58)$ yields a higher profit, \$189.5.

... an alternative

Consider again the chips-producing company. Their special chips are becoming so popular that a large supermarket chain wants to sell these potato chips in their stores. The additional restriction is that the chips must be delivered in batches of 30 kg, for efficiency reasons. As a result, the optimal production plan, determined above, is no longer adequate. The optimal amounts of 57.5 kg of plain and 50 kg of Mexican chips would make up, respectively, 1.92 and 1.67 batches of potato chips. This solution does not satisfy the additional restriction imposed by the supermarket chain. An integer programming model could now be used to determine a new (and acceptable) production plan.

Extending the potato chips example

The linear programming model of Section 2 can be reformulated as an integer programming model by measuring the amounts of chips in batches, and by adding the requirement that the solution takes on integer values. When measuring the amounts of chips in batches of 30 kg, the production processes require the following amounts of time. Each batch of plain chips takes 60 minutes to slice, 120 minutes to fry, and 120 minutes to pack. Each batch of Mexican chips takes 120 minutes to slice, 150 minutes to fry, and 60 minutes to pack. Furthermore, the net profit on a batch of plain chips is \$60, while the net profit on a batch of Mexican chips is \$45.

Description

Let X_p^B denote the number of batches of plain chips produced, and let X_m^B denote the number of batches of Mexican chips produced. Note that the restriction is added that X_p^B and X_m^B must have integer values (fractions of batches are not allowed). Then the integer programming model becomes:

Formulation

$$\begin{aligned}
 &\textbf{Maximize:} && 60X_p^B + 45X_m^B = P \\
 &\textbf{Subject to:} && \\
 &&& 60X_p^B + 120X_m^B \leq 345 && \text{(slicing)} \\
 &&& 120X_p^B + 150X_m^B \leq 480 && \text{(frying)} \\
 &&& 120X_p^B + 60X_m^B \leq 330 && \text{(packing)} \\
 &&& X_p^B, X_m^B \geq 0 \\
 &&& X_p^B, X_m^B \text{ integers}
 \end{aligned}$$

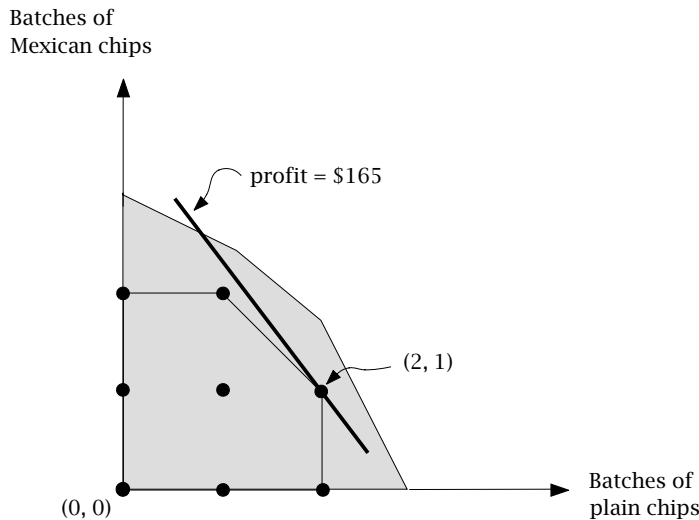


Figure 2.10: The feasible region in the integer program

Figure 2.10 is almost the same as Figure 2.4, except that the feasible region is now limited to the grid points within the region bounded by the constraints. It is clear that the grid point that maximizes the profit, subject to the constraints, is $(X_p^B, X_m^B) = (2, 1)$. This implies a production plan with 60 kilograms of plain chips, 30 kilograms of Mexican chips, and a profit of \$165. Note that this solution is not a corner or on the boundary of the region determined by the frying, slicing, and packing constraints. These constraints do limit the feasible region, however, because otherwise a higher profit would be possible. Notice also that the profit has dropped from \$190 to \$165. The company might consider making a weekly production plan, and delivering the batches once a week. In this way the production of a part of a batch in a day would be allowed, and this would increase the average daily profit. From this last comment, it is

Picturing the solution

clear that formulating a model involves more than merely formulating a set of constraints and an objective function.

Constrained optimization problems in which all variables must take on integer values are referred to as *pure integer programming* problems. Constrained optimization problems, in which only some of the variables must take on integer values, are referred to as *mixed integer programming* problems. If both the objective and all constraints are linear, then the term *(mixed) integer linear programming* applies.

*Pure and mixed
integer
programming*

A *zero-one programming* problem is a pure integer programming problem with the additional restraint that all variables are either zero or one. Such zero-one variables are referred to as *binary variables*. Binary variables provide a variety of new possibilities for formulating logical conditions and yes/no decisions in integer programming models. The use of binary variables is illustrated in Chapters 7 and 9.

*Zero-one
programming*

For a model builder it is a straightforward matter to add the requirement of integrality to a mathematical problem description. However, solving an integer programming problem is almost always harder than solving the underlying linear program. Most of the solution algorithms available do, in fact, *test* all promising integer solutions until the best (or a good) one is found. A sequence of linear programs is solved in order to obtain a solution of an integer program. The most widely used solution algorithms work on the so-called *branch and bound method*, in which a tree structure is used to conduct the search systematically. The branch and bound method, as well as some alternative algorithms, is explained in many textbooks on integer programming, for example [Ne88].

Solvability

In integer programming, the way a model is formulated can influence its solvability. If you have trouble finding an integer solution, a good approach is to first drop the integrality requirement and solve the model as an LP (this LP model is referred to as the *relaxed (LP) model* of the integer programming model). Then reformulate the model until the optimal LP solution value is close to the integer one. You can then apply an integer programming solution algorithm (see [Ga72] for examples). One method of speeding up the solution process is to bound all variables (upper and lower bounds) as tight as possible, so that the number of possible integer solutions is reduced.

*Relaxed LP
models*

In the previous paragraph, the remark was made that in *integer programming*, searching for an optimal solution might take too much time, because a sequence of linear programs has to be solved in order to obtain an integer solution. A widespread strategy is to specify a measure of the maximum (absolute or relative) deviation between the objective value of the integer solution and

*Optimality
tolerance*

the optimal objective value of the relaxed (LP) problem. As soon as a solution is within this measure, the algorithm terminates. The solution thus found can be regarded as a *compromise* between optimality and practical feasibility.

Notice that a feasible linear program may become infeasible as soon as the requirement of integrality is added. For example, picture a feasible region which is not empty, but excludes all grid points.

Infeasibility

To be able to solve a mixed integer models, integer variables should only be able to take a finite number of values. For this reason, pure integer models (i.e. models without any continuous variable) are never unbounded.

Unboundedness

Besides difficult integer programming problems, there exist also easy ones. These are problems that can be formulated as *network flow problems*. These problems can be solved as linear programs, and the optimal solution is guaranteed to be integer. This is due to a special mathematical property of network flow problems (i.e. *totally unimodularity* of the constraint matrix). It is unfortunate that this property can easily disappear as a result of small model extensions, so the number of easy-to-solve integer programming problems encountered in practice is relatively small. Network flow problems are discussed in more detail in Chapter 5.

Integer solutions from linear programs

2.3 Formulating nonlinear programming models

In this section some basic characteristics of the third type of constrained models, namely nonlinear programming models, will be discussed.

This section

Besides adding integrality constraints to a linear programming model, another major extension can be made by relaxing the convention that all expressions must be linear. Since many physical and economic phenomena in the world surrounding us are highly nonlinear, *nonlinear programming* models are sometimes required rather than linear ones.

Nonlinear expressions

Competition is growing. The chips-producing company decides to change its selling prices, which seem to be too high. The company decides to maximize the production of potato chips (which forms the supply), by setting the prices so that supply and demand are equal. Models in which supply and demand are equated are widely used in economics, and are known as *equilibrium models*. [Va84] gives an introduction. The company's marketing manager has determined the relationship between demand (D) and price (P) to be the following identities.

Extending the potato chips example

- Demand for plain chips: $D_p = -66.7P_p + 300$
- Demand for Mexican chips: $D_m = -83.4P_m + 416.6$

Furthermore, the fixed production costs are \$2 per kg for both chip types.

The constraints on the limited availability of the slicer, fryer, and packer have not changed. The objective, however, has. The net profit now depends on the prices that the company sets. By definition, the profit equals revenue minus costs, which can be written as

Formulation

$$P = (P_p - 2)X_p + (P_m - 2)X_m$$

Since the company wants to set the price in such a way that supply equals demand, the demand curves can be used to determine that price.

■ *Supply equals demand for plain chips:*

$$X_p = D_p = -66.7P_p + 300$$

■ *Supply equals demand for Mexican chips:*

$$X_m = D_m = -83.4P_m + 416.6$$

Which can also be written as

$$P_p = -0.015X_p + 4.5$$

$$P_m = -0.012X_m + 5.0$$

where D_p and D_m have been eliminated, and P_p and P_m have been solved for. These prices can now be used in the objective function, which becomes a nonlinear function depending only on X_p and X_m :

$$P = (-0.015X_p + 4.5 - 2)X_p + (-0.012X_m + 5 - 2)X_m$$

or

$$P = -0.015X_p^2 + 2.5X_p - 0.012X_m^2 + 3X_m$$

A new (nonlinear) objective function has been derived, and the model can now be stated completely.

Algebraic description

Maximize: $-0.015X_p^2 + 2.5X_p - 0.012X_m^2 + 3X_m = P$

Subject to:

$$2X_p + 4X_m \leq 345 \quad (\text{slicing})$$

$$4X_p + 5X_m \leq 480 \quad (\text{frying})$$

$$4X_p + 2X_m \leq 330 \quad (\text{packing})$$

$$X_p, X_m \geq 0$$

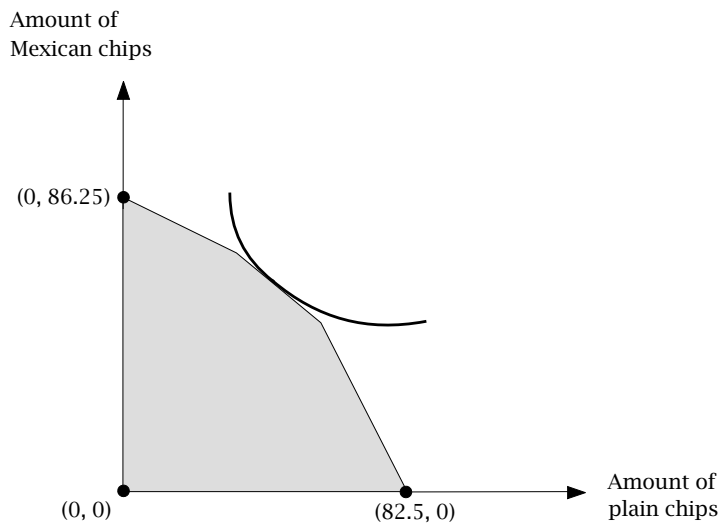


Figure 2.11: The feasible region in the nonlinear programming model

When the contour of the objective and the area of the feasible region are plotted, the same pictorial solution procedure can be followed as in the previous examples. Figure 2.11 shows this plot. One difference from previous examples is that the contour of the profit function is no longer linear. This contour can still be shifted to the right as long as it touches the feasible region. From the figure the optimal point can roughly be determined: about 40 kg of plain chips, and about 60 kg of Mexican chips. Note that this optimal point is not on a corner. Using AIMMS to determine the exact solution gives 42.84 kg of plain chips, and 61.73 kg of Mexican chips, which yields a profit of \$219.03.

Picturing the solution

A nonlinear programming problem consists of an algebraic objective function subject to a set of algebraic constraints and simple bounds. The term *algebraic* is used here to indicate that algebraic operators for addition, subtraction, division, multiplication, exponentiation, etc. are applied to the variables. Differential and integral operators are not considered. The algebraic constraints consist of equations and/or inequalities. The *simple bounds* are lower and/or upper bounds on the variables. The variables themselves can take on any real value between these bounds. If no simple bounds are stated for a particular variable, then its value may vary between minus infinity and plus infinity. Nonlinear programming models are often referred to as NLP models, and their objective contours and constraint boundaries need no longer be straight lines. The combinations of curved contours and boundaries can make them very difficult to solve.

Terminology

Local and global optima

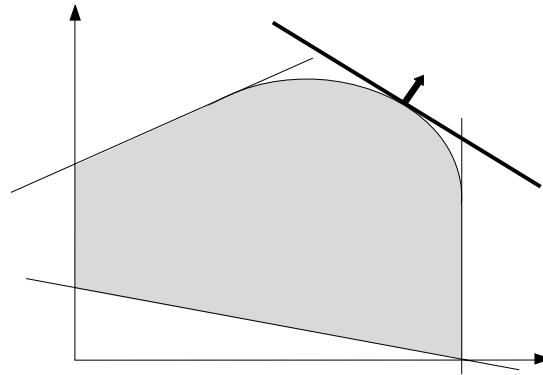


Figure 2.12: A non-corner solution of a nonlinear program

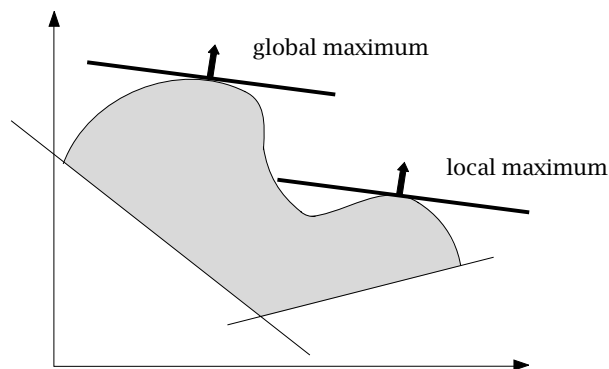


Figure 2.13: Local and global optima illustrated

Figures 2.11 and 2.12 illustrate that the optimal solution of a nonlinear programming model need not be on a corner. Furthermore if a solution is found, it may only be optimal with respect to the points in a small neighborhood, while a better optimum exists further away. An example of this is given in Figure 2.13. The terms *globally* and *locally optimal* are used to make the distinction. Only in the specific situation where the nonlinear programming problem has only one optimal solution can one ignore this distinction. The theoretical conditions under which a problem only has one optimal solution are not easy to verify for most real problems. As a result caution on the part of a model builder is needed. For a theoretical treatment of nonlinear programming, see [Ba79].

Even though it is not too difficult to construct a large variety of nonlinear programming problems, solving them is a different matter. The reasons for this are that the theory of nonlinear programming is much less developed than the theory of linear programming, and because the solution algorithms are not always capable of solving the nonlinear programming problems presented to them. It should be noted, however, that problems in which the nonlinear terms appear only in the objective function are generally easier to solve than those in which nonlinear terms occur in the constraints. Nonlinear programming models with integer variables are even more difficult to solve and will not be addressed in this book.

Solvability

2.4 Summary

In this chapter, the three most common types of constrained have been introduced: linear programming models, integer linear programming models and nonlinear programming models. *Linear programming (LP) models* must satisfy the restriction that both the objective function and the constraints are linear. Despite this restriction, linear programming models are still the most common due to the availability of powerful solution algorithms. *Integer linear programming (IP) models* are like linear ones, except that one or more of the variables must take on integer values. Except for a small group, namely network flow models, integer programming models are harder to solve than linear programs. *Nonlinear programming (NLP) models* can have a nonlinear objective and/or nonlinear constraints. An optimal solution is not guaranteed to be globally optimal. When an optimization model is solved, one of the following situations will occur: an optimal solution is found, which is the desired situation; the problem is infeasible, which is a result of inconsistencies in the constraint set; the problem is unbounded, which is often caused by a modeling error.

Chapter 3

Algebraic Representation of Models

In this chapter, the method of translating an explicit formulation to an AIMMS formulation is explained. A sequence of different representations of the same model demonstrates the use of *symbols* to represent data, the use of *index notation*, and the AIMMS *modeling language*.

This chapter

The notation in this chapter is derived from standard mathematical notation. For the representation of models, you are referred to [Sc91] and [Wi90].

References

3.1 Explicit form

In this section, the potato chips example from the previous chapter is revisited. The formulation below is usually referred to as the *explicit* form in standard algebraic notation. *Algebraic notation* is a mathematical notation, as are other notations such as matrix notation, or the AIMMS notation in Section 3.4. With the help of this example, the differences between several representations of the same model are illustrated.

This section

Variables:

X_p	amount of plain chips produced [kg]
X_m	amount of Mexican chips produced [kg]

Potato chips model

Maximize:

$$2X_p + 1.5X_m \quad (\text{net profit})$$

Subject to:

$$2X_p + 4X_m \leq 345 \quad (\text{slicing})$$

$$4X_p + 5X_m \leq 480 \quad (\text{frying})$$

$$4X_p + 2X_m \leq 330 \quad (\text{packing})$$

$$X_p, X_m \geq 0$$

The above formulation is a correct representation of the problem description in mathematical form. However, it is *not a practical* mathematical description of the problem.

The most obvious shortfall of the explicit form is that the numbers in the model are given without comment. While examining the model one must either look up or recall the meaning of each number. This is annoying and does not promote a quick understanding of the model. In larger models, it can cause errors to go unnoticed.

Unexplained numbers

It is better to attach a descriptive symbol to each number or group of numbers, plus a brief description for even further clarification. Entering these symbols into the model formulation instead of the individual numbers will lead to a model statement that is easier to understand. In addition, it paves the way for a more structured approach to model building. Specifically, if the values associated with a symbol change at a later stage, then the changes only need to be made at one place in the model. This leads to a considerable improvement in efficiency. These remarks give the motivation for *symbolic* model formulation.

Possible improvements

3.2 Symbolic form

In the symbolic form, there is a separation between the symbols and their values. A model in symbolic form consists of the following building blocks:

Separation between symbols and values

- *symbols (parameters)*, representing data in symbolic form,
- *variables*, representing the unknowns, and
- *objective and constraints*, defining the relationships between symbols and variables.

The *data* is not a part of a symbolic model formulation. Values are assigned to the symbols when the model is solved. The data for the potato chips model can be found in Chapter 2.

Parameters:

Potato chips model

A_S	<i>available slicing time [min]</i>
A_F	<i>available frying time [min]</i>
A_P	<i>available packing time [min]</i>
N_p	<i>net profit of plain chips [\$/kg]</i>
N_m	<i>net profit of Mexican chips [\$/kg]</i>
S_p	<i>time required for slicing plain chips [min/kg]</i>
S_m	<i>time required for slicing Mexican chips [min/kg]</i>
F_p	<i>time required for frying plain chips [min/kg]</i>
F_m	<i>time required for frying Mexican chips [min/kg]</i>
P_p	<i>time required for packing plain chips [min/kg]</i>
P_m	<i>time required for packing Mexican chips [min/kg]</i>

Nonnegative variables:

X_p	<i>quantity of plain chips produced [kg]</i>
X_m	<i>quantity of Mexican chips produced [kg]</i>

Maximize:

$$N_p X_p + N_m X_m \quad (\text{net profit})$$

Subject to:

$$S_p X_p + S_m X_m \leq A_S \quad (\text{slicing time})$$

$$F_p X_p + F_m X_m \leq A_F \quad (\text{frying time})$$

$$P_p X_p + P_m X_m \leq A_P \quad (\text{packing time})$$

$$X_p, X_m \geq 0$$

This representation is evaluated and discussed below.

In this small example, eleven parameters and two variables are needed to generate a symbolic description of the model. Imagine a situation in which the number of production processes and the number of chip types are both in double figures. The number of constraints will be in the tens but the number of parameters will be in the hundreds. This is clearly unacceptable in practice. The way to compact the formulation is to use *index notation*, as explained in Section 3.3.

Too many symbols

It is worthwhile to note that the names of the symbols have not been chosen arbitrarily. Although they are short, they give more meaning than a number. For instance, the S which indicates the slicer in A_S (available slicing time) also indicates the slicer in S_p (time required for slicing plain chips). Furthermore, the A in A_S obviously denotes availability. It is important to choose the names of the symbols in a sensible way because it improves the clarity of the model. However, as observed earlier, there are quite a lot of symbols in the model statement above. The larger the model, the more inventiveness one requires to think of meaningful, unique names for all the identifiers. Again, index notation provides a way out, and thus, the naming of symbols will be reconsidered in the next section.

Meaningful names for symbols

When the data is kept separate from the symbolic model statement, the model statement can describe a whole range of situations, rather than one particular situation. In addition, if changes occur in the data, these changes only have to be made in one place. So the separation of model and data provides greater flexibility and prevents errors when updating values.

Separation of model and data

3.3 Symbolic-indexed form

Index notation is a technique for reducing the number of symbols and facilitating the naming of parameters. Consider the potato chip example using this new, compressed formulation.

This section

According to Webster's dictionary [We67], one of the meanings of the word *index* is pointer. It points to, or indicates an element of a set. The terms, *set* and *index*, are elaborated further using the potato chips example.

Indicating an element of a set

Recall the notation in the previous example, for instance: X_p “amount of plain chips produced.” It is clear that the “ p ” indicates plain chips. So the “ p ” is used as an index, but it only points to a set with one element. The difficulty encountered in the previous section, where there were too many symbols, was caused by having all indices pointing only to single-element sets. When combining these sets with similar entities, the number of symbols can be reduced. The first set that seems logical to specify is a set of chip types:

Set of chip types

$$I = \{\text{plain}, \text{Mexican}\}$$

Then one can state:

$$x_i \quad \text{amount of chips produced of type } i \text{ [kg]}$$

So the index i indicates an element of the set I , and the two decision variables are now summarized in one statement. It is customary to use *subscripts* for indices. Moreover, the mathematical shorthand for “ i taken from the set I ” is $i \in I$. The index i for every symbol referring to chip types in the model can be introduced to obtain four new parameter declarations.

Index notation

Parameters:

$$\begin{array}{ll} n_i & \text{net profit of chips of type } i \text{ [$/kg]} \\ S_i & \text{time required for slicing chips of type } i \text{ [min/kg]} \\ F_i & \text{time required for frying chips of type } i \text{ [min/kg]} \\ P_i & \text{time required for packing chips of type } i \text{ [min/kg]} \end{array}$$

The number of parameters has been reduced from eleven to seven by adding one set. Note that indices do not need units of measurement. They just indicate certain entities—elements of a set.

What is striking in the above list is the similarity of S_i , F_i , and P_i . All three symbols are for time requirements of different production processes. In a way, S , F , and P serve as indices pointing to single element sets of production processes. Because the processes all play a similar role in the model, one more general set can be introduced.

Set of production processes

$$J = \{\text{slicing}, \text{frying}, \text{packing}\}$$

An index j , pointing to members of J , can take over the role of S , F , and P . Now one symbol can summarize the six symbols S_p , S_m , F_p , F_m , P_p , P_m that were previously needed to describe the time required by the production processes.

$$r_{ij} \quad \text{time required by process } j \text{ for chips of type } i \text{ [min/kg]}$$

The index j can also be used to express the availabilities of the machines that carry out the processes.

a_j *available processing time for process j [min]*

At this point two sets (I and J) and three parameters (a_j , n_i , r_{ij}) remain. The notation for the constraint specifications can also be compacted using indexing.

When looking at the first constraint, and trying to write it down with the notation just developed, the following expression can be obtained.

Summation operator

$$r_{\text{mexican}, \text{ slicing}} x_{\text{mexican}} + r_{\text{plain}, \text{ slicing}} x_{\text{plain}} \leq a_{\text{slicing}}$$

Obviously there is room for improvement. This is possible using the well-known summation operator; now used to indicate a summation over different elements of the set of chip types,

$$\sum_i r_{ij} x_i \leq a_j \quad \forall j$$

where $\forall j$ is shorthand notation meaning *for all elements j (in the set J)*.

The symbols defined above are used in the following *indexed* formulation of the potato chips problem with the actual numeric data omitted.

Symbolic-indexed formulation

Indices:

i *chip types*
 j *production processes*

Parameters:

a_j *available processing time of process j [min]*
 n_i *net profit of chips of type i [\$/kg]*
 r_{ij} *time requirements of type i and of process j [min/kg]*

Variables:

x_i *amount of chips produced of type i [kg]*

Maximize:

$$\sum_i n_i x_i \quad \text{(net profit)}$$

Subject to:

$$\begin{aligned} \sum_i r_{ij} x_i &\leq a_j & \forall j & \quad \text{(time limits)} \\ x_i &\geq 0 & \forall i & \end{aligned}$$

In previous statements of the potato chips model, there were always three constraints describing the limited availability of different production processes. In the symbolic indexed formulation, the use of the index j for production processes enables one to state just one *grouped* constraint, and add the remark “ $\forall j$ ” (for all j). Thus, index notation provides not only a way to summarize many similar identifiers, but also to summarize similar equations. The latter are referred to as *constraint declarations*

Reducing the number of statements

In the previous section, it was noted that index notation would also be helpful in reducing the number of identifiers. Using indices of group parameters and variables has reduced the number of identifier descriptors from thirteen to four.

Reducing the number of identifiers

As a result of reducing the number of identifiers, it is easier to choose unique and meaningful names for them. A name should indicate the common feature of the group. For algebraic notation, the convention is to choose single letter names, but this marginally improves the readability of a model. At most, it contributes to its compactness. In practical applications longer and more meaningful names are used for the description of identifiers. The AIMMS language permits the names of identifiers to be as long as you find convenient.

More meaningful names

Note that the size of a set can be increased without increasing the length of the model statement. This is possible because the list of set elements is part of the data and not part of the model formulation. The advantages are obvious. Specifically, the number of indexed identifiers and the number of indexed equations are not impacted by the number of set elements. In addition, as with the rest of the data, changes can be made easily, so index notation also contributes to the generality of a model statement. When symbolic notation is introduced there is separation between the model statement and the data. This separation is complete when index notation is used.

Expanding the model with set elements

3.4 AIMMS form

The last step is to represent the model using the AIMMS modeling language. This yields the advantages that error checks can be carried out, and that the software can activate a solver to search for a solution.

This section

By using the AIMMS Model Explorer, a model created in AIMMS is essentially a graphical representation. At the highest level there is a tree to structure your model in sections and subsections. At the leaves of the tree you specify your declarations and procedures. For each identifier declaration there is a form by which you enter all relevant attributes such as index domain, range, text, unit, definition, etc.

Models in AIMMS

Figure 3.1 gives you an idea on how the symbolic-indexed representation of the potato chips problem can be structured in the Model Editor. Note that in AIMMS, the full length descriptor of $\text{ProcessTimeRequired}(p, c)$ replaces the r_{ij} which was used in the earlier mathematical formulation. Clearly, this improves the readability of the model. In AIMMS, symbols are still typically used for set indexing. The set of chips is given the index c and the set of processes, the index p . In the earlier mathematical representation, i and j were used for these sets respectively.

Example



Figure 3.1: AIMMS Model representation of the potato chips model

The graphical tree representation of models inside the Model Explorer is just one way to view a model. In large-scale applications it is quite natural to want to view selected portions of your model. AIMMS allows you to create your own identifier selections and your own view windows. By dragging a particular identifier selection to a particular view window you obtain your own customized view. You may also edit your model components from within such a customized view.

Multiple views

Figure 3.2 gives an example of a view in which the variables and constraints of the potato chips problem are listed, together with their index domain, definition and unit. Note that the AIMMS notation in the definition attributes resembles the standard algebraic index notation introduced in the previous section.

Example

View Window: Domain - Definition - Unit				
	Identifier	Index domain	Definition	Unit
<input checked="" type="checkbox"/>	Production	(c)		kg
<input checked="" type="checkbox"/>	TotalProfit		$\text{sum}[c, \text{NetProfit}(c) * \text{Production}(c)]$	\$
<input checked="" type="checkbox"/>	ProcessRequirements	(p)	$\text{sum}[c, \text{ProcessTimeRequired}(p, c) * \text{Production}(c)]$ \leq $\text{AvailableTime}(p)$	min

Figure 3.2: An AIMMS view for the potato chips model

Data must be initialized and added to an AIMMS model formulation because the computer needs this data to solve the model. More than one such data set can be associated with a model, allowing for different versions of the same model. The data set for the potato chips problem is presented in the form of an ASCII file. In most real-world applications such data would be read directly by AIMMS from a database.

*Data
initialization*

3.5 Translating a verbal problem into a model

Throughout this book, the same sequence of steps will be used when translating a verbal problem description into an optimization model. It is assumed that a verbal problem description, posed in such a way that a model can be used to support the decision, is available. Of course, the translation from a real-life problem into a well-posed verbal problem statement is far from trivial, but this exercise is outside the scope of this book.

*Getting verbal
problem
description*

The framework for analyzing a verbal problem is presented below. Such a framework has the advantage that it facilitates a structured approach.

*A general
framework*

When analyzing a problem in order to develop a model formulation the following questions need to be answered.

- Which sets can be specified for indexing data and variables?

Such sets have just been explained. The advantages mentioned in Section 3.3 justify the use of index notation throughout the remainder of this manual. Sets often appear in verbal problem descriptions as lists of similar entities, or as labels in tables, such as the production processes in Table 2.1.

- What are the decision variables in the problem?

Decision variables reflect a choice, or a trade-off, to be made. They are the unknowns to be determined by the model. In fact, the decision reflected in the decision variables is often the very reason for building the model.

- What entity is to be optimized?

In small examples, the *objective* is often stated explicitly in the problem description. In real-world problems, however, there may be many, possibly conflicting, objectives. In these cases, it is worthwhile experimenting with different objectives.

- What constraints are there?

Constraints can also include procedural checks on solutions to see if they are usable. A potential solution that does not satisfy all constraints is not usable. The two questions about the objective and constraints can often be answered simultaneously. It is strongly recommended that you specify the measurement

units of the variables, the objective and the constraints. Since this is a way of checking the consistency of the model statement and can prevent you from making many mistakes.

The answers to these questions for the potato chips problem have already been given implicitly. They are summarized here once more. The *sets* in the potato chips problem are given by the sets of production processes and types of chips. The *decision variables* are the amounts of both types of chips to be produced, measured in kilograms. The *objective* is net profit maximization, measured in dollars. The *constraints* are formed by the limited availability of the production processes, measured in minutes.

Potato chips
model

3.6 Summary

This chapter has shown a sequence of different representations of the same model in order to introduce the use of symbols, index notation and the AIMMS language. While using an *explicit* (with numeric values) form of standard algebraic notation may initially be the intuitive way to write down a model, this form is only suitable for the simplest of models. A superior representation is to replace numbers with symbols, thereby obtaining a *symbolic* model representation. Advantages of this representation are that you do not have to remember the meanings of the numbers and that the data, which does not influence the model structure, is separated from the model statement. Another refinement to model formulation is to specify index sets and to use indices to group identifiers and constraints. This yields the *symbolic-indexed* form. This form is recommended because it combines the advantages of the symbolic form with the compactness of index notation. Finally, the sequence of steps to translate a verbal description of a problem to a mathematical programming model was given.

Chapter 4

Sensitivity Analysis

The subject of this chapter is the introduction of *marginal values* (shadow prices and reduced costs) and *sensitivity ranges* which are tools used when conducting a sensitivity analysis of a linear programming model. A sensitivity analysis investigates the changes to the optimal solution of a model as the result of changes in input data.

This chapter

Sensitivity analysis is discussed in a variety of text books. A basic treatment can be found in, for instance, [Ch83], [Ep87] and [Ko87].

References

4.1 Introduction

In a linear program, slack variables may be introduced to transform an inequality constraint into an equality constraint. When the simplex method is used to solve a linear program, it calculates an *optimal solution* (i.e. optimal values for the decision and/or slack variables), an *optimal objective function value*, and partitions the variables into *basic variables* and *nonbasic variables*. Nonbasic variables are always at one of their bounds (upper or lower), while basic variables are between their bounds. The set of basic variables is usually referred to as the *optimal basis* and the corresponding solution is referred to as the *basic solution*. Whenever one or more of the basic variables (decision and/or slack variables) happen to be at one of their bounds, the corresponding basic solution is said to be *degenerate*.

Terminology

The simplex algorithm gives extra information in addition to the optimal solution. The algorithm provides *marginal values* which give information on the variability of the optimal solution to changes in the data. The marginal values are divided into two groups:

Marginal values

- *shadow prices* which are associated with constraints and their right-hand side, and
- *reduced costs* which are associated with the decision variables and their bounds.

These are discussed in the next two sections.

In addition to marginal values, the simplex algorithm can also provide *sensitivity range* information. These ranges are defined in terms of two of the characteristics of the optimal solution, namely the optimal objective value and the optimal basis. By considering the objective function as fixed at its optimal value, sensitivity ranges can be calculated for both the decision variables and the shadow prices. Similarly, it is possible to fix the optimal basis, and to calculate the sensitivity ranges for both the coefficients in the objective function and the right-hand sides of the constraints. All four types will be illustrated in this chapter.

*Sensitivity
ranges*

Although algorithms for integer programming also provide marginal values, the applicability of these figures is very limited, and therefore they will not be used when examining the solution of an integer program.

*Linear
programming
only*

4.2 Shadow prices

In this section all constraints are assumed to be in standard form. This means that all variable terms are on the left-hand side of the (in)equality operator and the right-hand side consists of a single constant. The following definition then applies.

*Constant
right-hand side*

The marginal value of a constraint, referred to as its shadow price, is defined as the rate of change of the objective function from a one unit increase in its right-hand side. Therefore, a positive shadow price indicates that the objective will increase with a unit increase in the right-hand side of the constraint while a negative shadow price indicates that the objective will decrease. For a nonbinding constraint, the shadow price will be zero since its right-hand side is not constraining the optimal solution.

Definition

To improve the objective function (that is, decreases for a minimization problem and increases for a maximization problem), it is necessary to weaken a binding constraint. This is intuitive because relaxing is equivalent to enlarging the feasible region. A " \leq " constraint is weakened by *increasing* the right-hand side and a " \geq " constraint is weakened by *decreasing* the right-hand side. It therefore follows that the signs of the shadow prices for binding inequality constraints of the form " \leq " and " \geq " are opposite.

*Constraint
weakening*

When your model includes equality constraints, such a constraint could be incorporated into the LP by converting it into two separate inequality constraints. In this case, at most one of these will have a nonzero price. As discussed above, the nature of the binding constraint can be inferred from the sign of its shadow price. For example, consider a minimization problem with a negative shadow price for an equality constraint. This indicates that the objective will decrease

*Equality
constraints*

(i.e. improve) with an increase in the right-hand side of the equality constraint. Therefore, it is possible to conclude that it is the “ \leq ” constraint (and not the “ \geq ” constraint) that is binding since it is relaxed by increasing the right-hand side.

Table 4.1 presents the shadow prices associated with the constraints in the potato chips example from Chapter 2.

Potato chips model

process	constraint	optimal time [min]	upper bound [min]	shadow price [\$ / min]
slicing	$2X_p + 4X_m \leq 345$	315	345	0.00
frying	$4X_p + 5X_m \leq 480$	480	480	0.17
packing	$4X_p + 2X_m \leq 330$	330	330	0.33

Table 4.1: Shadow prices

The objective in the potato chips model is profit maximization with less than or equal process requirement constraints. The above shadow prices can be used to estimate the effects of changing the binding constraints. Specifically, as discussed below, it is possible to deduce from the positive values of the frying and packing constraints that there will be an increase in the overall profit if *these* process times are increased.

It is important to note that the slicing inequality constraint is nonbinding at the optimal solution and hence its associated shadow price is zero. This predicts that there will be no improvement in the objective function value if the constraint is relaxed. This is expected because a sufficiently small change in the right-hand side of such a constraint has no effect on the (optimal) solution. In contrast, a change in the objective function is expected for each sufficiently small change in the right-hand side of a *binding* constraint.

Nonbinding constraint

The benefit of relaxing a binding constraint can be investigated by resolving the LP with the upper bound on the availability of the packer increased to 331 minutes. Solving gives a new profit of \$190.33, which is exactly the amount predicted by the shadow price (\$0.33). Similarly an upper bound of 332 minutes gives rise to a profit of \$190.67. This shows that the shadow price gives the revenue of an extra minute of packing time, which can then be compared with the cost of installing additional packing equipment. The shadow price can therefore be considered to represent the benefit of relaxing a constraint. From comparing the shadow prices of frying and packing, it is fair to conclude that upgrading packing equipment is probably more attractive than frying equipment.

Relaxing binding constraints

If a binding constraint is tightened, the value of the objective will deteriorate by the amount approximated by the shadow price. For the case of lowering the upper bound on the availability of the packer to 329 minutes, profit decreases by \$0.33 to \$189.67. This shows that the shadow price gives the amount of change in both directions.

Tightening constraint

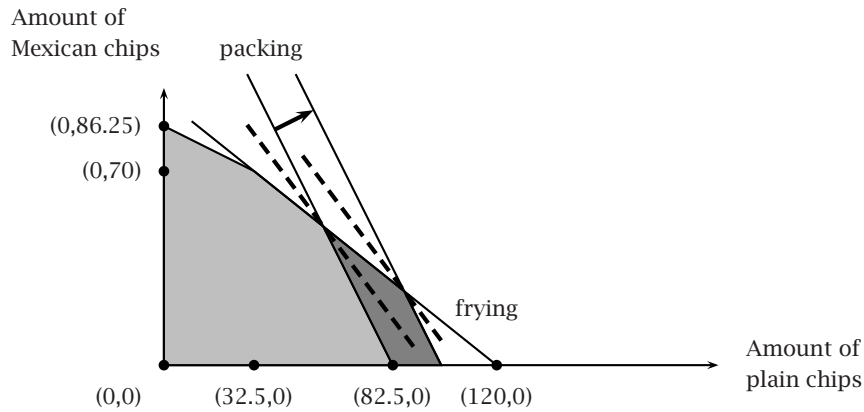


Figure 4.1: Weakening the constraint on packing

In Figure 4.1, there is a graphical illustration of what happens when the packing constraint is weakened. The corresponding line shifts to the right, thus enlarging the feasible region. Consequently, the dashed line segment representing the linear objective function can also shift to the right, yielding a more profitable optimal solution. Notice that if the constraint is weakened much further, it will no longer be binding. The optimal solution will be on the corner where the frying line and the plain chips axis intersect (120,0). This demonstrates that the predictive power of shadow prices in some instances only applies to limited changes in the data.

Picturing the process

In general, the conclusions drawn by analyzing shadow prices are only true for small changes in data. In addition, they are only valid if one change is made at a time. The effects of changing more data at once cannot be predicted.

Shadow price limitations

In the two decision variable example to date, there have only been two binding constraints. However, if there were three or more constraints binding at the optimal solution, weakening one requirement may not have the effect suggested by the shadow prices. Figure 4.2 depicts this situation, where the bold lines can be interpreted as three constraints that intersect at the optimal solution. This condition is referred to as *degeneracy* and can be detected when one or more of the variables (decision and/or slack variables) in the basis are at one of their bounds. In this example, one of the three slack variables will be in the basis at their bound of zero. In the presence of degeneracy, shadow

Degeneracy

prices are no longer unique, and their interpretation is therefore ambiguous. Under these conditions, it is useful to analyse sensitivity ranges as discussed in Section 4.4.

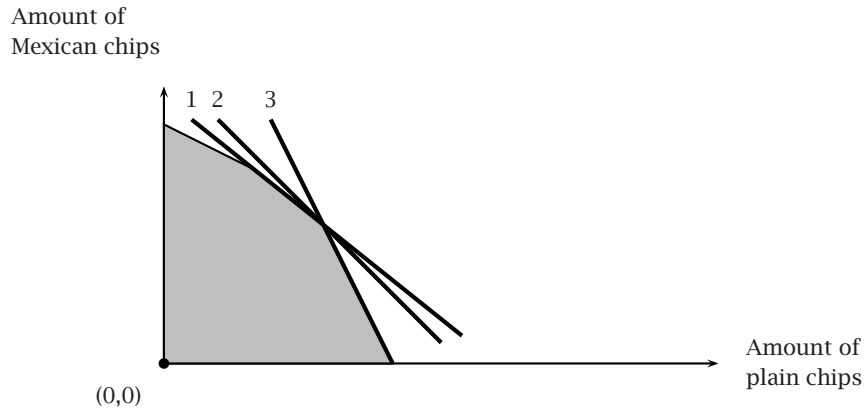


Figure 4.2: Non-unique solutions illustrated

In general, the information provided by shadow prices should only be used as an indication of the potential for improvement to the optimal objective function value. However, there are some circumstances where a slightly stronger conclusion can be drawn. Specifically, if there are shadow prices that are large relative to others, then it is possible that the optimal solution is overly sensitive to changes in the corresponding data. Particular care should be taken when this data is not known exactly. Under these conditions, it might be wise to use methods specifically designed for handling uncertainty in data, or to run a set of experiments investigating the exact effect on the (optimal) solution with particular data modifications.

Conclusion

4.3 Reduced costs

A decision variable has a marginal value, referred to as its reduced cost, which is defined as the rate of change of the objective function for a one unit increase in the bound of this variable. If a nonbasic variable has a positive reduced cost, the objective function will increase with a one unit increase in the binding bound. The objective function will decrease if a nonbasic variable has a negative reduced cost. The reduced cost of a basic variable is zero since its bounds are nonbinding and therefore do not constrain the optimal solution.

Definition

By definition, a nonbasic variable is at one of its bounds. Moving it off the bound when the solution is optimal, is detrimental to the objective function value. A nonbasic variable will improve the objective function value when its binding bound is relaxed. Alternatively, the incentive to include it in the basis can be increased by adjusting its cost coefficient. The next two paragraphs explain how reduced cost information can be used to modify the problem to change the basis.

Improving the objective

A nonbasic variable is at either its upper or lower bound. The reduced cost gives the possible improvement in the objective function if its bound is relaxed. Relax means decreasing the bound of a variable at its lower bound or increasing the bound of a variable at its upper bound. In both cases the size of the feasible region is increased.

Bound relaxation

The objective function value is the summation of the product of each variable by its objective cost coefficient. Therefore, by adjusting the objective cost coefficient of a nonbasic variable it is possible to make it active in the optimal solution. The reduced cost represents the amount by which the cost coefficient of the variable must be *lowered*. A variable with a positive reduced cost will become active if its cost coefficient is lowered, while the cost coefficient of a variable with a negative reduced cost must be increased.

Objective coefficient

Table 4.2 gives the reduced costs associated with the optimal solution of the potato chips model. In this problem the decision variables are the quantity of both types of chips to be included in the optimal production plan. The reduced costs of both chip types are zero. This is expected since neither chip type is at a bound (upper or lower) in the optimal solution. It is possible to make one variable nonbasic (at a bound) by modifying the data.

Potato chips model

chip type	optimal value [kg]	reduced costs [\$ /kg]
plain	57.5	0.0
Mexican	50.0	0.0

Table 4.2: Reduced costs in the potato chips model

A modification to the potato chips model which results in a nonzero reduced cost, is to lower the net profit contribution of Mexican chips from 1.50 to 0.50 \$/kg. Solving the model gives the optimal production plan in Table 4.3, where Mexican chips production is now nonbasic and at its lower bound of zero. As a result, there is a reduced cost associated with the production of Mexican chips. The profit has dropped from \$190 to \$165, which is the best achievable with these profit contributions.

The modified potato chips model

chip type	optimal value [kg]	reduced costs [\$ /kg]
plain	82.5	0.0
Mexican	0.0	−0.5

Table 4.3: Reduced costs in the modified potato chips model

In Table 4.3, the zero reduced cost of plain chips production reflects that it is basic. The nonzero reduced cost for Mexican chips indicates that it is at a bound (lower). Its negative reduced cost indicates that the objective function value will decrease should the quantity of Mexican chips be increased by one unit. Given that it is a maximization problem, this is unattractive and hence is consistent with the decision to place the variable at its lower bound.

Interpretation

The optimal Mexican chips production is at its lower bound because it is more attractive to produce plain chips. However, by adjusting its objective cost coefficient it is possible for Mexican chips to become active in the optimal solution. From Table 4.3, and using the fact that the potato chips model is a maximization model, it can be concluded that if the profit contribution from Mexican chips is increased by at least 0.5 \$/kg, then Mexican chips will become basic.

*Adjusting
objective
coefficient*

Changing coefficients in the objective can be regarded as changing the slope of the objective function. In Figure 4.3, profit lines corresponding to different profit contributions from Mexican chips are given. It can easily be seen that the slope of the objective determines which corner solution is optimal. Reduced costs give the minimal change in a coefficient of the objective such that the optimal solution shifts from a corner on one of the axes to another corner of the feasible region (possibly on one or more of the other axes). Note that the slope of line (2) is parallel to the slope of the constraint on packing, thus yielding multiple optimal solutions.

*Picturing the
process*

One might conclude that a model never needs to be solved on the computer more than once since all variations can be derived from the reduced costs and shadow prices. However, often it is useful to conduct some further sensitivity analysis. In general, shadow prices and reduced costs are only valid in a limited sense, but that they are useful when their values are large relative to others. Their exact range of validity is not known a priori and their values need not be unique. It is a result of this limitation that the study of sensitivity ranges becomes useful.

Conclusion

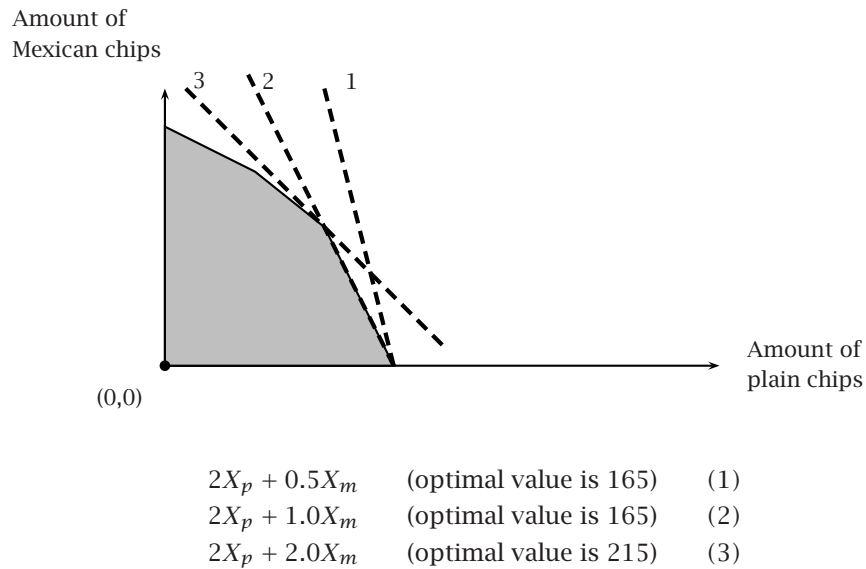


Figure 4.3: Varying the slope of the objective function

4.4 Sensitivity ranges with constant objective function value

Optimal decision variables and shadow prices are not always unique. In this section the range of values of optimal decision variables and optimal shadow prices for the potato chips model is examined. In AIMMS there are in-built facilities to request such range information.

This section

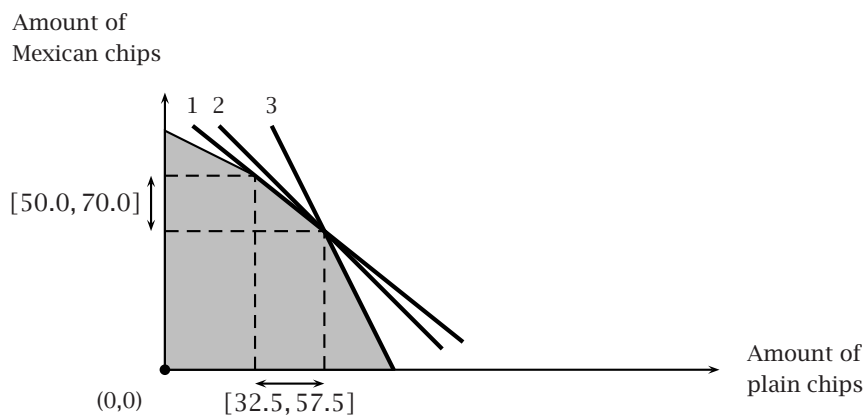


Figure 4.4: Decision variable ranges illustrated

Figure 4.4 illustrates the sensitivity ranges for decision variables if the three bold lines are interpreted as different objective contours. It is clear that for contour (1) there is a *range* of values for the amount of plain chips ([32.5, 57.5] kg) and, a corresponding range for the amount of Mexican chips ([50.0, 70.0] kg) that can yield the same objective value. Contour (3) also exhibits this behavior but the ranges are different. For objective contour (2), there is a unique optimal decision.

*Decision
variable ranges*

The bold lines in Figure 4.4 were initially interpreted as constraints that intersect at the optimal solution. In this case, the shadow prices are not unique and the situation is referred to as a case of degeneracy. The potato chip problem to date does not have a constraint corresponding to line (2) but a new constraint can easily be added for illustrative purposes only. This constraint limits the objective value to be less than its optimal value. Thus, the contours in Figure 4.4 can also be interpreted as follows:

*Shadow price
ranges*

1. frying constraint,
2. new constraint limiting the optimal value, and
3. packing constraint.

Examine the shadow prices for values of the bounds in a very small neighborhood about their nominal values. This helps to see that there are multiple solutions for the shadow prices. If constraint (2) in Figure 4.4 is binding with shadow price equal to 1.0 \$/min, then the shadow prices on constraints (1) and (3) will necessarily be zero. By relaxing constraint (2) a very small amount, it becomes non-binding. Its shadow price will go to zero, and as this happens, constraints (1) and (3) become binding with positive prices equal to the optimal values from Table 4.1. This means that in this special case there is a range of shadow prices for all three constraints where the optimal objective value remains constant.

*Examining their
values*

1. frying constraint has shadow price range [0.0, 0.17]
2. new constraint has shadow price range [0.0, 1.0], and
3. packing constraint has shadow price range [0.0, 0.33].

4.5 Sensitivity ranges with constant basis

The optimal basis does not always remain constant with changes in input data. In this section the ranges of values of objective function coefficients and right-hand sides of the original potato chips model are examined with the requirement that the optimal basis does not change. In AIMMS there are in-built facilities to request such range information.

This section

Changing one or more coefficients in the objective has the effect of changing the slope of the objective contours. This can be illustrated by interpreting the bold lines in Figure 4.4 as the result of

Ranges of objective coefficients

1. decreased plain chip profits (1.2 \$/kg)
2. nominal plain chip profits (2.0 \$/kg), and
3. increased plain chip profits (3.0 \$/kg),

Note that the optimal basis for the nominal profits is still optimal for the other two objectives. Therefore, the range of objective coefficient values defined by contours (1) and (3) represent the amount of plain chips for which the optimal basis remains constant. Outside this range, there would be a change in the optimal basis (movement to a different extreme point).

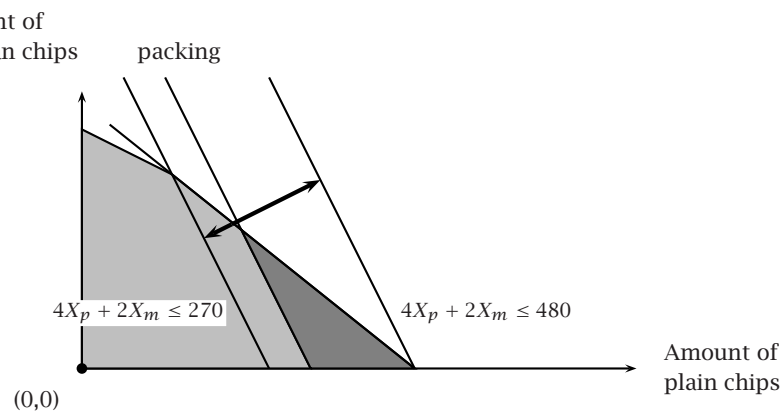


Figure 4.5: Right-hand side ranges illustrated

The potato chip model uses less than or equal constraints, but the following analysis also holds for greater than or equal constraints. The nominal solution of the potato chip problem has the packing and frying constraints binding. These binding constraints represent a basis for the shadow prices. By changing the right-hand side on the packing constraint, it will shift as can be seen in Figure 4.5.

Ranges of right-hand sides

The right-hand side can shift *up* to 480.0 minutes, where it would become redundant with the lower bound of zero on the amount of Mexican chips. The solution is then degenerate, and there are multiple shadow price solutions. This can also be interpreted as a change in the basis for the shadow prices. The right-hand side can shift *down* to 270.0 minutes, where it becomes redundant with the slicing constraint, and another change in the shadow price basis can occur. Through this exercise, it has been shown that the right-hand side on

Examining their values

the packing constraint has a range of $[270.0, 480.0]$ minutes over which the shadow price basis does not change. Any extension of this range will force a change in the binding constraints at the optimal solution. Changing the right-hand side of non-binding constraints can make them become binding. The non-binding constraints in the potato chip problem are the slicing constraint and the two non-negativity constraints on the decision variables.

4.6 Summary

In this chapter, the concepts of marginal values and ranges have been explained using the optimal solution of the potato chips model. The use of both shadow prices and reduced costs in sensitivity analysis has been demonstrated. Sensitivity ranges have been introduced to provide validity ranges for the optimal objective function value and optimal basis. Although there is some benefit in predicting the effect of changes in data, it has been shown that these indicators do have their limits. Repeated solving of the model provides the best method of sensitivity analysis, and the AIMMS modeling system has some powerful facilities to support this type of sensitivity analysis.

Chapter 5

Network Flow Models

Here, network flow models are introduced as a special class of linear programming models. The AIMMS network formulation is also introduced, and some sensitivity analysis is performed. Furthermore, several classes of network flow models are described.

This chapter

Overviews of network algorithms can be found in [Go77], [Ke80] and [Or93]. An overview of applications of network flow models can be found in [Gl92] and [Or93].

References

5.1 Introduction

A *network* is a schematic diagram, consisting of points which are connected by lines or arrows. An example is given in Figure 5.1. The points are referred to as *nodes* and the lines are called *arcs*. A *flow* may occur between two nodes, via an arc. When the flow is restricted to one direction, then the arcs are pointed and the network is referred to as a *directed network*.

What is a network?

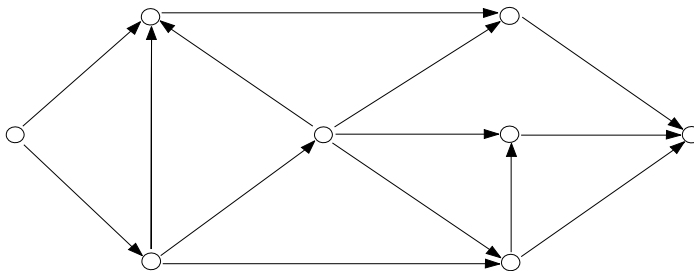


Figure 5.1: A directed network

Network flow models form a class by themselves. They are linear programming models, and can be formulated and solved as such. In practice, however, network flow models are modeled more naturally in terms of nodes and arcs, and are solved quicker by special network algorithms. Therefore, a special type of AIMMS formulation is available for network problems.

What is a network flow model?

In the following section, an example of a network flow model is given. This example concerns the shipment of goods from factories to customers. The nodes of the network are the factories and the customers, while the arcs represent the possible routes over which the goods can be shipped. The amounts of goods actually shipped form the flows along the various arcs.

The next section

5.2 Example of a network flow model

A Dutch company has two factories, one located at Arnhem and one located at Gouda. The company sells its products to six customers, located in London, Berlin, Maastricht, Amsterdam, Utrecht and The Hague. For reasons of efficiency, deliveries abroad are only made by one factory: Arnhem delivers to Berlin, while Gouda ships goods to London. Figure 5.2 illustrates the situation.

*Verbal
description*



Figure 5.2: Factories and customers

Each factory has a limited supply of goods: Arnhem has 550 tons, and Gouda has 650 tons available. Customer demands and transportation costs from factory to customer are specified in Table 5.1. The goal is to satisfy the customers' demand while minimizing transportation costs.

Model data

from to	Arnhem [1000 \$/ton]	Gouda [1000 \$/ton]	Demand [tons]
London		2.5	125
Berlin	2.5		175
Maastricht	1.6	2.0	225
Amsterdam	1.4	1.0	250
Utrecht	0.8	1.0	225
The Hague	1.4	0.8	200

Table 5.1: Transportation costs and customer demands

The index sets are the sets of factories and customers. These two quantities determine the size of the underlying optimization model.

Index sets

Since the goal of the model is to answer the question, “How many tons of goods should be shipped from the various factories to the various customers?”, the decision variables are the numbers of items to be shipped from each factory to each customer, and are measured in tons. Notice that there are ten decision variables, one for each factory-customer pair drawn in Figure 5.2. Models like this one illustrate the usefulness of index notation, since the number of variables increases rapidly when the number of factories or customers grows.

Decision variables

The objective is to minimize the transportation costs of transporting goods from factories to customers. The costs are measured in thousands of dollars per ton.

Objective

The constraints follow logically from the problem statement. First, the amount of goods shipped from a factory must be less than or equal to the supply at that factory. Second, the amount of goods shipped to customers must meet their demand. So there are two groups of constraints: supply and demand constraints, both measured in tons.

Constraints

Minimize: The total transportation costs,

Subject to:

- for all factories: Total shipment from a factory can at most be the supply, and
- for all customers: Total shipment to a customer must at least be the demand.

The verbal formulation

In the verbal formulation, no attention has been paid to the fact that not all combinations of factories and customers are permitted. There are several ways to model this. The first one is to incorporate non-permitted combinations with high cost, so that they will never be used. This option is not recommended for efficiency reasons, since the model contains unwanted variables that unnecessarily increase the size of the problem. The second, and recommended, alternative is to restrict the domain over which the variables are defined, thereby eliminating unwanted variables. For example, one could replace the first constraint as follows:

- *for all factories: Total shipment from a factory to permitted customers can at most be the supply.*

In this example, the nonzero transportation costs from Table 5.1 can be used as a basis to restrict the index domain of the variables. These costs give an unambiguous indication of which combinations are permitted. When no cost figure is supplied, then a particular combination factory-customer is to be ignored.

*Modeling
non-permitted
combinations ...*

*... using
nonzero costs*

5.3 Network formulation

There are two possible formulations for network problems. One is the version of the linear programming model stated above. Another option is to take advantage of the special structure of the network. Network flow models can be formulated in a natural way in terms of nodes and arcs. In addition, AIMMS provides a special network flow algorithm, that solves these problems faster than would a standard linear programming algorithm.

*Two
formulation
options*

Before formulating the example as a network flow model, some comments are made on the network interpretation of this problem. The basic concepts are *supply* and *demand*. The factories are considered supply nodes. The flow (of goods) out of the various supply nodes must not exceed the amount of goods available. This is again the supply constraint. The customers are considered demand nodes. The flow into the demand nodes must at least match the amount of goods required. Again, the demand constraint appears. Finally, the flows must be such that the costs of transportation are minimized.

*Network
interpretation*

In AIMMS it is possible to specify a problem by creating a model using arcs and nodes. ARC and NODE declarations have taken the place of VARIABLES and CONSTRAINTS, respectively. Furthermore, the keyword `NetInflow` indicates the flow into a node minus the flow out of it, whereas the keyword `NetOutflow` has the opposite interpretation. These keywords enable one to specify the *balance constraints* on each node. For each arc, the associated pair of nodes is specified, as well as costs attached to it, using the attributes FROM, TO, and COST. Capacities on arcs are specified using its RANGE attribute.

*Network
formulation and
AIMMS*

The following symbols are used in the mathematical description of the network example of the previous section.

*Model
declarations*

Indices:

f	<i>factories</i>
c	<i>customers</i>

Parameters:

S_f	<i>supply at factory f</i>
D_c	<i>demand by customer c</i>
T_{fc}	<i>unit transport cost between f and c</i>

Nodes:

FN_f	<i>factory supply node for f</i>
CN_c	<i>customer demand node for c</i>

Arcs:

$Flow_{fc}$	<i>transport between f and c</i>
-------------	--

The following declarations mimic the declarations typically found in AIMMS network models. They take the place of the usual algebraic notation to describe constraints in constraint-based models.

*Nodes and arcs
in AIMMS*

NODES:

```

identifier : FN
index domain : f
definition : NetOutflow <= S(f)
text       : factory supply node for f ;

identifier : CN
index domain : c
definition : NetInflow >= D(c)
text       : customer demand node for c ;

```

ARC:

```

identifier : Flow
index domain : (f,c) | T(f,c)
range       : nonnegative
from        : FN(f)
to          : CN(c)
cost        : T(f,c) ;

```

Network models form a special class of mathematical programs for which there is no generally accepted notation other than the standard flow balances. This is an instance in which modeling languages such as AIMMS have introduced their own keywords to facilitate the description of large-scale symbolic network models.

*No standard
notation*

5.4 Solution and sensitivity analysis

The optimal solution of the model and data described in the previous sections could be given in a table, but it is more clearly presented as a picture. The optimal deliveries are given in Figure 5.3. The optimal total transportation cost is \$1,715,000.

The optimal solution

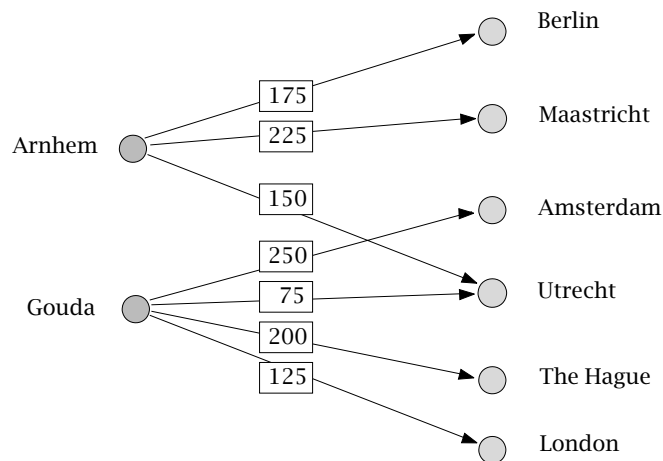


Figure 5.3: The optimal network solution

In Table 5.2, the reduced costs are given for those routes that were not included in the optimal solution.

Reduced costs

from factory	to customer	reduced costs [1000 \$/ton]
Arnhem	Amsterdam	0.6
	The Hague	0.8
Gouda	Maastricht	0.2

Table 5.2: Reduced costs

From this table, it is likely that shipments from Gouda to Maastricht would be included in the optimal program if the transportation costs were reduced by approximately \$200/ton (from \$2000/ton to \$1800/ton). Solving the modified model confirms this prediction. Another optimal solution exists and is given in Figure 5.4. The total costs are still \$1,715,000.

The modified network model

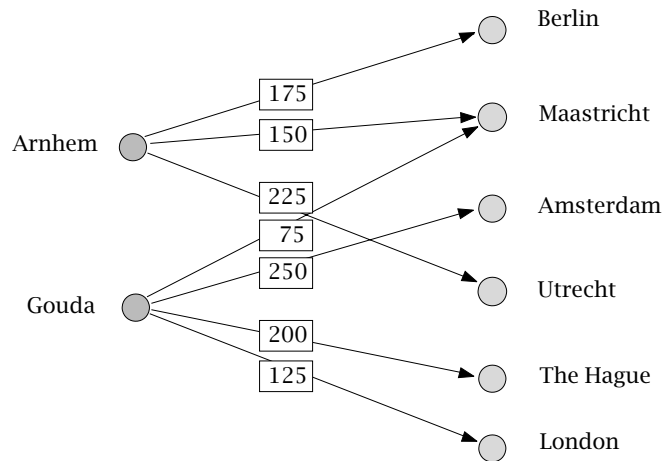


Figure 5.4: An optimal solution of the modified network model

In Table 5.3, the shadow prices corresponding to the demand constraints for the original network model are given. Adjusting the demand in Berlin downwards reduces the objective the most. There is a reduction of \$2,500 per unit transport, but there is the extra benefit that Arnhem is then free to supply Utrecht at a reduction of \$200 per unit over Gouda. This gives an overall reduction of \$2,700 per unit.

Shadow prices

	shadow price [\$1000/ton]
London	2.5
Berlin	2.7
Maastricht	1.8
Amsterdam	1.0
Utrecht	1.0
The Hague	0.8

Table 5.3: The shadow prices for the demand constraint

5.5 Pure network flow models

In this section several generic examples of a pure network flow model are presented.

This section

The example from the previous sections is generally referred to as a *transportation problem*. Transportation problems are characterized by the fact that the nodes are divided into two distinct sets of supply nodes and demand nodes. Supply nodes are often referred to as *sources*, and demand nodes are known as *sinks*. All the arcs in the network go from a source to a sink.

The transportation problem

The *assignment problem* is a special case of the transportation problem. It has the same *bipartite* structure as the transportation problem, but the supply or demand of each node is exactly one. Several practical problems can be modeled as an assignment problem. Examples are the assignment of personnel to tasks and the assignment of jobs to machines.

The assignment problem

A general pure network flow model may also contain intermediate (or transshipment) nodes. These nodes can have both a flow into the node and a flow out of the node. This type of problem is often referred to as the *transshipment problem*. For instance, adding nodes for distribution centers, with arcs from the factories and arcs to the customers, turns the transportation problem into a transshipment problem. Transshipment models are used in a wide range of practical problems, such as distribution problems, scheduling inventory and production problems, and land allocation problems.

The transshipment problem

In most practical situations, the flow along an arc is not unlimited, but restricted to some finite capacity. Upper bounds on the flow along arcs are easily handled by the network algorithm. Similarly, lower bounds on the flow along arcs can also be included.

Adding capacities on the arcs

Assuming that the objective is to minimize the total costs associated with the flow along the arcs, the general pure network flow model can be summarized as follows.

The general pure network flow model

Minimize: Total costs,

Subject to:

- for each supply node: the net outflow must be (less than or) equal to the available supply,
- for each demand node: the net inflow must be (greater than or) equal to the required demand,
- for each transshipment node: the net inflow must be equal to the net outflow, and
- for each arc: the flow must be within its bounds.

Pure network flow problems have a surprising feature. When all demands are integer-valued, and all lower and upper bounds on the flows along the arcs are integer-valued, then the following is true:

Integer solutions

If the network flow model has at least one feasible solution, it has an integer-valued feasible solution, furthermore if it has an optimal solution, it has an integer-valued optimal solution.

In this situation, the network simplex algorithm is guaranteed to find such a solution, and you do not have to resort to an integer programming algorithm. The gains in computation time are considerable.

5.6 Other network models

This section describes two types of problems that can be formulated as pure network flow models, and also some types of network problems that cannot be formulated as pure network models. All of these models can be represented within the AIMMS modeling language.

This section

The *shortest path problem* is a problem that can be formulated as a transshipment model. As the name suggests, the goal is to find the shortest path between a single origin and a single destination. All one has to do is to place a supply of one at the origin node and a demand of one at the destination node. All the intermediate nodes have zero demand and supply. The lengths of the arcs are used as costs. The shortest path from the origin to the destination is then determined by the arcs that carry a nonzero flow in the optimal solution of the transshipment model.

The shortest path problem

The objective in a *maximum flow problem* is to maximize the flow through the network from a single source to a single sink, while the arcs can only carry a limited flow. This problem can be stated as a capacitated transshipment problem by introducing one additional arc from the sink to the source with infinite capacity. The cost attached to this new arc is -1.0 , while the cost attached to all the original arcs is zero. All nodes in the network (including the source and sink) have zero demand and supply. By minimizing the total cost, the maximum flow through the network is found. An example of the maximum flow problem can be found in a traffic network, where the arcs, representing roads, have limited traffic capacity. The traffic flows are measured in, for example, number of cars per hour. There are other examples in which the flows represent either messages in a telephone network, or cash in a currency trading network, or water in a pipe transport network.

The maximum flow problem

In a pure network flow model, the flow along an arc is conserved, while in *generalized network problems* gains or losses can be specified for each arc. Gains and losses can be due to conversion of units, waste, loss in quality, etc. Generalized network models cannot be solved with a pure network flow algorithm. There are special codes, however, that solve generalized network models, but

Generalized network problems

these codes, just as with linear programming solvers, do not necessarily terminate with an integer-valued optimal solution.

Multi-commodity network flow problems are just like capacitated transshipment or transportation problems, except that there is more than one commodity to be shipped. In most applications, the arc capacity restrictions do not apply to just a single commodity, but to several commodities together. In this case, the multi-commodity network model cannot be solved with a pure network flow algorithm. The comments made for generalized network models apply here as well. There are specialized solvers, but they too do not necessarily terminate with an integer-valued optimal solution. In practice, linear programming solvers and constraint generation techniques are frequently used for the solution of large-scale multi-commodity network models.

Multi-commodity network problems

5.7 Summary

In this chapter, a transportation problem has been formulated as an optimization model. Transportation problems belong to a special class of network flow problems. Although these problems can be formulated as linear programming models, it is much more natural to formulate them in terms of nodes and arcs, taking advantage of the special structure of the problem. Moreover, solution algorithms exist that take advantage of the network structure of the problem. These algorithms often reach an optimal solution much faster than would linear programming solvers. AIMMS provides facilities for both formulating and solving network models. A special property of many network flow models is that the optimal solution is integer-valued as long as the supplies and demands attached to the sources and sinks are integers. Some examples of different well-known classes of network flow problems were given

Part II

**General Optimization Modeling
Tricks**

Chapter 6

Linear Programming Tricks

This chapter explains several *tricks* that help to transform some models with special, for instance nonlinear, features into conventional linear programming models. Since the fastest and most powerful solution methods are those for linear programming models, it is often advisable to use this format instead of solving a nonlinear or integer programming model where possible.

This chapter

The linear programming tricks in this chapter are not discussed in any particular reference, but are scattered throughout the literature. Several tricks can be found in [Wi90]. Other tricks are referenced directly.

References

Throughout this chapter the following general statement of a linear programming model is used:

Statement of a linear program

$$\begin{array}{ll}\text{Minimize:} & \sum_{j \in J} c_j x_j \\ \text{Subject to:} & \sum_{j \in J} a_{ij} x_j \geq b_i \quad \forall i \in I \\ & x_j \geq 0 \quad \forall j \in J\end{array}$$

In this statement, the c_j 's are referred to as *cost coefficients*, the a_{ij} 's are referred to as *constraint coefficients*, and the b_i 's are referred to as *requirements*. The symbol " \geq " denotes any of " \leq ", " $=$ ", or " \geq " constraints. A maximization model can always be written as a minimization model by multiplying the objective by (-1) and minimizing it.

6.1 Absolute values

Consider the following model statement:

The model

$$\begin{array}{ll}\text{Minimize:} & \sum_{j \in J} c_j |x_j| \quad c_j > 0 \\ \text{Subject to:} & \sum_{j \in J} a_{ij} x_j \geq b_i \quad \forall i \in I \\ & x_j \text{ free}\end{array}$$

Instead of the standard cost function, a weighted sum of the absolute values of the variables is to be minimized. To begin with, a method to remove these absolute values is explained, and then an application of such a model is given.

The presence of absolute values in the objective function means it is not possible to directly apply linear programming. The absolute values can be avoided by replacing each x_j and $|x_j|$ as follows.

*Handling
absolute
values ...*

$$\begin{aligned} x_j &= x_j^+ - x_j^- \\ |x_j| &= x_j^+ + x_j^- \\ x_j^+, x_j^- &\geq 0 \end{aligned}$$

The linear program of the previous paragraph can then be rewritten as follows.

$$\begin{aligned} \text{Minimize:} \quad & \sum_{j \in J} c_j (x_j^+ + x_j^-) & c_j > 0 \\ \text{Subject to:} \quad & \sum_{j \in J} a_{ij} (x_j^+ - x_j^-) \geq b_i & \forall i \in I \\ & x_j^+, x_j^- \geq 0 & \forall j \in J \end{aligned}$$

The optimal solutions of both linear programs are the same if, for each j , at least one of the values x_j^+ and x_j^- is zero. In that case, $x_j = x_j^+$ when $x_j \geq 0$, and $x_j = -x_j^-$ when $x_j \leq 0$. Assume for a moment that the optimal values of x_j^+ and x_j^- are both positive for a particular j , and let $\delta = \min\{x_j^+, x_j^-\}$. Subtracting $\delta > 0$ from both x_j^+ and x_j^- leaves the value of $x_j = x_j^+ - x_j^-$ unchanged, but reduces the value of $|x_j| = x_j^+ + x_j^-$ by 2δ . This contradicts the optimality assumption, because the objective function value can be reduced by $2\delta c_j$.

... correctly

Sometimes x_j represents a *deviation* between the left- and the right-hand side of a constraint, such as in *regression*. Regression is a well-known statistical method of fitting a curve through observed data. One speaks of *linear regression* when a straight line is fitted.

*Application:
curve fitting*

Consider fitting a straight line through the points (v_j, w_j) in Figure 6.1. The coefficients a and b of the straight line $w = av + b$ are to be determined. The coefficient a is the *slope* of the line, and b is the *intercept* with the w -axis. In general, these coefficients can be determined using a model of the following form:

Example

$$\begin{aligned} \text{Minimize:} \quad & f(z) \\ \text{Subject to:} \quad & w_j = av_j + b - z_j \quad \forall j \in J \end{aligned}$$

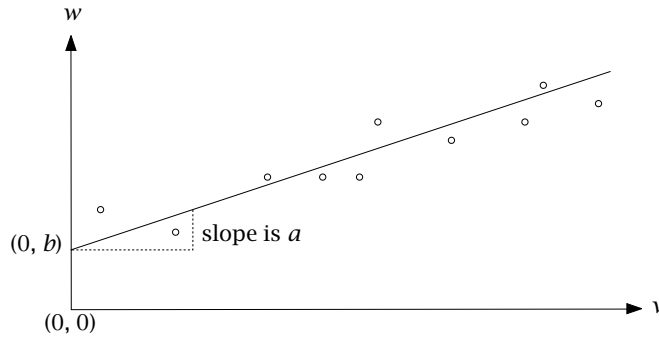


Figure 6.1: Linear regression

In this model z_j denotes the difference between the value of $av_j + b$ proposed by the linear expression and the observed value, w_j . In other words, z_j is the *error* or deviation in the w direction. Note that in this case a , b , and z_j are the decision variables, whereas v_j and w_j are data. A function $f(z)$ of the error variables must be minimized. There are different options for the objective function $f(z)$.

Least-squares estimation is an often used technique that fits a line such that the sum of the squared errors is minimized. The formula for the objective function is:

$$f(z) = \sum_{j \in J} z_j^2$$

It is apparent that quadratic programming must be used for least squares estimation since the objective is quadratic.

Least absolute deviations estimation is an alternative technique that minimizes the sum of the absolute errors. The objective function takes the form:

$$f(z) = \sum_{j \in J} |z_j|$$

When the data contains a few extreme observations, w_j , this objective is appropriate, because it is less influenced by extreme outliers than is least-squares estimation.

Least maximum deviation estimation is a third technique that minimizes the maximum error. This has an objective of the form:

$$f(z) = \max_{j \in J} |z_j|$$

This form can also be translated into a linear programming model, as explained in the next section.

Different objectives in curve fitting

6.2 A minimax objective

Consider the model

The model

$$\begin{aligned}
 \text{Minimize:} \quad & \max_{k \in K} \sum_{j \in J} c_{kj} x_j \\
 \text{Subject to:} \quad & \sum_{j \in J} a_{ij} x_j \geq b_i \quad \forall i \in I \\
 & x_j \geq 0 \quad \forall j \in J
 \end{aligned}$$

Such an objective, which requires a maximum to be minimized, is known as a *minimax objective*. For example, when $K = \{1, 2, 3\}$ and $J = \{1, 2\}$, then the objective is:

$$\text{Minimize:} \quad \max\{c_{11}x_1 + c_{12}x_2, c_{21}x_1 + c_{22}x_2, c_{31}x_1 + c_{32}x_2\}$$

An example of such a problem is in least maximum deviation regression, explained in the previous section.

The minimax objective can be transformed by including an additional decision variable z , which represents the maximum costs:

Transforming a minimax objective

$$z = \max_{k \in K} \sum_{j \in J} c_{kj} x_j$$

In order to establish this relationship, the following extra constraints must be imposed:

$$\sum_{j \in J} c_{kj} x_j \leq z \quad \forall k \in K$$

Now when z is minimized, these constraints ensure that z will be greater than, or equal to, $\sum_{j \in J} c_{kj} x_j$ for all k . At the same time, the optimal value of z will be *no greater than* the maximum of all $\sum_{j \in J} c_{kj} x_j$ because z has been minimized. Therefore the optimal value of z will be both as small as possible and exactly equal to the maximum cost over K .

$$\begin{aligned}
 \text{Minimize:} \quad & z \\
 \text{Subject to:} \quad & \sum_{j \in J} a_{ij} x_j \geq b_i \quad \forall i \in I \\
 & \sum_{j \in J} c_{kj} x_j \leq z \quad \forall k \in K \\
 & x_j \geq 0 \quad \forall j \in J
 \end{aligned}$$

The equivalent linear program

The problem of maximizing a minimum (a *maximin* objective) can be transformed in a similar fashion.

6.3 A fractional objective

Consider the following model:

The model

$$\begin{aligned}
 &\textbf{Minimize:} && \left(\sum_{j \in J} c_j x_j + \alpha \right) / \left(\sum_{j \in J} d_j x_j + \beta \right) \\
 &\textbf{Subject to:} && \sum_{j \in J} a_{ij} x_j \geq b_i \quad \forall i \in I \\
 &&& x_j \geq 0 \quad \forall j \in J
 \end{aligned}$$

In this problem the objective is the ratio of two linear terms. It is assumed that the denominator (the expression $\sum_{j \in J} d_j x_j + \beta$) is either positive or negative over the entire feasible set of x_j . The constraints are linear, so that a linear program will be obtained if the objective can be transformed to a linear function. Such problems typically arise in financial planning models. Possible objectives include the rate of return, turnover ratios, accounting ratios and productivity ratios.

The following method for transforming the above model into a regular linear programming model is from Charnes and Cooper ([Ch62]). The main trick is to introduce variables y_j and t which satisfy: $y_j = tx_j$. In the explanation below, it is assumed that the value of the denominator is positive. If it is negative, the directions in the inequalities must be reversed.

Transforming a fractional objective

1. Rewrite the objective function in terms of t , where

$$t = 1 / \left(\sum_{j \in J} d_j x_j + \beta \right)$$

and add this equality and the constraint $t > 0$ to the model. This gives:

$$\begin{aligned}
 &\textbf{Minimize:} && \sum_{j \in J} c_j x_j t + \alpha t \\
 &\textbf{Subject to:} && \sum_{j \in J} a_{ij} x_j \geq b_i \quad \forall i \in I \\
 &&& \sum_{j \in J} d_j x_j t + \beta t = 1 \\
 &&& t > 0 \\
 &&& x_j \geq 0 \quad \forall j \in J
 \end{aligned}$$

2. Multiply both sides of the original constraints by t , ($t > 0$), and rewrite the model in terms of y_j and t , where $y_j = x_j t$. This yields the model:

$$\begin{aligned}
&\text{Minimize:} && \sum_{j \in J} c_j y_j + \alpha t \\
&\text{Subject to:} && \sum_{j \in J} a_{ij} y_j \geq b_i t \quad \forall i \in I \\
&&& \sum_{j \in J} d_j y_j + \beta t = 1 \\
&&& t > 0 \\
&&& y_j \geq 0 \quad \forall j \in J
\end{aligned}$$

3. Finally, temporarily allow t to be ≥ 0 instead of $t > 0$ in order to get a linear programming model. This linear programming model is equivalent to the fractional objective model stated above, provided $t > 0$ at the optimal solution. The values of the variables x_j in the optimal solution of the fractional objective model are obtained by dividing the optimal y_j by the optimal t .

6.4 A range constraint

Consider the following model:

The model

$$\begin{aligned}
&\text{Minimize:} && \sum_{j \in J} c_j x_j \\
&\text{Subject to:} && d_i \leq \sum_{j \in J} a_{ij} x_j \leq e_i \quad \forall i \in I \\
&&& x_j \geq 0 \quad \forall j \in J
\end{aligned}$$

When one of the constraints has both an upper and lower bound, it is called a *range constraint*. Such a constraint occurs, for instance, when a minimum amount of a nutrient is required in a blend and, at the same time, there is a limited amount available.

The most obvious way to model such a range constraint is to replace it by two constraints:

*Handling
a range
constraint*

$$\begin{aligned}
&\sum_{j \in J} a_{ij} x_j \geq d_i \quad \text{and} \\
&\sum_{j \in J} a_{ij} x_j \leq e_i \quad \forall i \in I
\end{aligned}$$

However, as each constraint is now stated twice, both must be modified when changes occur. A more elegant way is to introduce extra variables. By introducing new variables u_i one can rewrite the constraints as follows:

$$u_i + \sum_{j \in J} a_{ij} x_j = e_i \quad \forall i \in I$$

The following bound is then imposed on u_i :

$$0 \leq u_i \leq e_i - d_i \quad \forall i \in I$$

It is clear that $u_i = 0$ results in

$$\sum_{j \in J} a_{ij} x_j = e_i$$

while $u_i = e_i - d_i$ results in

$$\sum_{j \in J} a_{ij} x_j = d_i$$

A summary of the formulation is:

The equivalent linear program

$$\begin{aligned} \text{Minimize:} & \quad \sum_{j \in J} c_j x_j \\ \text{Subject to:} & \quad u_i + \sum_{j \in J} a_{ij} x_j = e_i \quad \forall i \in I \\ & \quad x_j \geq 0 \quad \forall j \in J \\ & \quad 0 \leq u_i \leq e_i - d_i \quad \forall i \in I \end{aligned}$$

6.5 A constraint with unknown-but-bounded coefficients

This section considers the situation in which the coefficients of a linear inequality constraint are unknown-but-bounded. Such an inequality in terms of uncertainty intervals is not a deterministic linear programming constraint. Any particular selection of values for these uncertain coefficients results in an unreliable formulation. In this section it will be shown how to transform the original nondeterministic inequality into a set of deterministic linear programming constraints.

This section

Consider the constraint with unknown-but-bounded coefficients \tilde{a}_j

Unknown-but-bounded coefficients

$$\sum_{j \in J} \tilde{a}_j x_j \leq b$$

where \tilde{a}_j assumes an unknown value in the interval $[L_j, U_j]$, b is the fixed right-hand side, and x_j refers to the solution variables to be determined. Without loss of generality, the corresponding bounded uncertainty intervals can be written as $[a_j - \Delta_j, a_j + \Delta_j]$, where a_j is the midpoint of $[L_j, U_j]$.

Replacing the unknown coefficients by their midpoint results in a deterministic linear programming constraint that is not necessarily a reliable representation of the original nondeterministic inequality. Consider the simple linear program

Midpoints can be unreliable

$$\begin{array}{ll} \text{Maximize:} & x \\ \text{Subject to:} & \tilde{a}x \leq 8 \end{array}$$

with the uncertainty interval $\tilde{a} \in [1, 3]$. Using the midpoint $a = 2$ gives the optimal solution $x = 4$. However, if the true value of \tilde{a} had been 3 instead of the midpoint value 2, then for $x = 4$ the constraint would have been violated by 50%.

Consider a set of arbitrary but fixed x_j values. The requirement that the constraint with unknown-but-bounded coefficients must hold for the unknown values of \tilde{a}_j is certainly satisfied when the constraint holds for all possible values of \tilde{a}_j in the interval $[a_j - \Delta_j, a_j + \Delta_j]$. In that case it suffices to consider only those values of \tilde{a}_j for which the term $\tilde{a}_j x_j$ attains its maximum value. Note that this situation occurs when \tilde{a}_j is at one of its bounds. The sign of x_j determines which bound needs to be selected.

Worst-case analysis

$$\begin{aligned} \tilde{a}_j x_j &\leq a_j x_j + \Delta_j x_j & \forall x_j \geq 0 \\ \tilde{a}_j x_j &\leq a_j x_j - \Delta_j x_j & \forall x_j \leq 0 \end{aligned}$$

Note that both inequalities can be combined into a single inequality in terms of $|x_j|$.

$$\tilde{a}_j x_j \leq a_j x_j + \Delta_j |x_j| \quad \forall x_j$$

As a result of the above worst-case analysis, solutions to the previous formulation of the original constraint with unknown-but-bounded coefficients \tilde{a}_j can now be guaranteed by writing the following inequality without reference to \tilde{a}_j .

An absolute value formulation

$$\sum_{j \in J} a_j x_j + \sum_{j \in J} \Delta_j |x_j| \leq b$$

In the above absolute value formulation it is usually too conservative to require that the original deterministic value of b cannot be loosened. Typically, a tolerance $\delta > 0$ is introduced to allow solutions x_j to violate the original right-hand side b by an amount of at most $\delta \max(1, |b|)$.

A tolerance ...

The term $\max(1, |b|)$ guarantees a positive increment of at least δ , even in case the right-hand side b is equal to zero. This modified right-hand side leads to the following δ -tolerance formulation where a solution x_j is feasible whenever it satisfies the following inequality.

... relaxes the right-hand side

$$\sum_{j \in J} a_j x_j + \sum_{j \in J} \Delta_j |x_j| \leq b + \delta \max(1, |b|)$$

This δ -tolerance formulation can be rewritten as a deterministic linear programming constraint by replacing the $|x_j|$ terms with nonnegative variables y_j , and requiring that $-y_j \leq x_j \leq y_j$. It is straightforward to verify that these last two inequalities imply that $y_j \geq |x_j|$. These two terms are likely to be equal when the underlying inequality becomes binding for optimal x_j values in a linear program. The final result is the following set of deterministic linear programming constraints, which captures the uncertainty reflected in the original constraint with unknown-but-bounded coefficients as presented at the beginning of this section.

The final formulation

$$\begin{aligned} \sum_{j \in J} a_j x_j + \sum_{j \in J} \Delta_j y_j &\leq b + \delta \max(1, |b|) \\ -y_j &\leq x_j \leq y_j \\ y_j &\geq 0 \end{aligned}$$

6.6 A probabilistic constraint

This section considers the situation that occurs when the right-hand side of a linear constraint is a random variable. As will be shown, such a constraint can be rewritten as a purely deterministic constraint. Results pertaining to probabilistic constraints (also referred to as chance-constraints) were first published by Charnes and Cooper ([Ch59]).

This section

Consider the following linear constraint

Stochastic right-hand side

$$\sum_{j \in J} a_j x_j \leq B$$

where $J = \{1, 2, \dots, n\}$ and B is a random variable. A solution $x_j, j \in J$, is feasible when the constraint is satisfied for all possible values of B .

For open-ended distributions the right-hand side B can take on any value between $-\infty$ and $+\infty$, which means that there cannot be a feasible solution. If the distribution is not open-ended, suppose for instance that $B_{\min} \leq B \leq B_{\max}$, then the substitution of B_{\min} for B results in a deterministic model. In most

Acceptable values only

practical applications, it does not make sense for the above constraint to hold for all values of B .

Specifying that the constraint $\sum_{j \in J} a_j x_j \leq B$ must hold for all values of B is equivalent to stating that this constraint must hold with probability 1. In practical applications it is natural to allow for a small margin of failure. Such failure can be reflected by replacing the above constraint by an inequality of the form

A probabilistic constraint

$$\Pr \left[\sum_{j \in J} a_j x_j \leq B \right] \geq 1 - \alpha$$

which is called a *linear probabilistic constraint* or a *linear chance-constraint*. Here \Pr denotes the phrase "Probability of", and α is a specified constant fraction ($\in [0, 1]$), typically denoting the maximum error that is allowed.

Consider the density function f_B and a particular value of α as displayed in Figure 6.2.

Deterministic equivalent

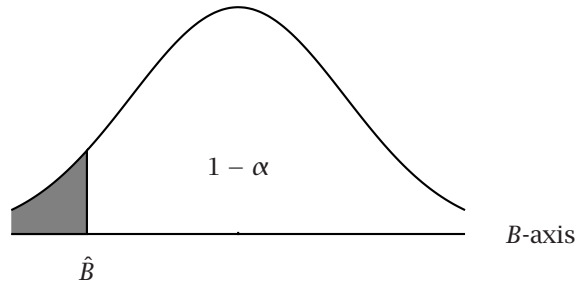


Figure 6.2: A density function f_B

A solution $x_j, j \in J$, is considered feasible for the above probabilistic constraint if and only if the term $\sum_{j \in J} a_j x_j$ takes a value beneath point \hat{B} . In this case a fraction $(1 - \alpha)$ or more of all values of B will be larger than the value of the term $\sum_{j \in J} a_j x_j$. For this reason \hat{B} is called the *critical value*. The probabilistic constraint of the previous paragraph has therefore the following deterministic equivalent:

$$\sum_{j \in J} a_j x_j \leq \hat{B}$$

The critical value \hat{B} can be determined by integrating the density function from $-\infty$ until a point where the area underneath the curve becomes equal to α . This point is then the value of \hat{B} . Note that the determination of \hat{B} as described in this paragraph is equivalent to using the inverse cumulative distribution function of f_B evaluated at α . From probability theory, the cumulative distribution

Computation of critical value

function F_B is defined by $F_B(x) = \Pr[B \leq x]$. The value of F_B is the corresponding area underneath the curve (probability). Its inverse specifies for each particular level of probability, the point \hat{B} for which the integral equals the probability level. The cumulative distribution function F_B and its inverse are illustrated in Figure 6.3.

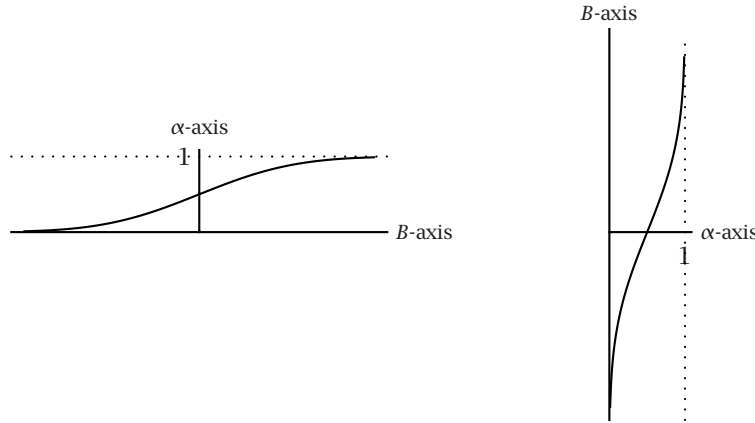


Figure 6.3: Cumulative distribution function F and its inverse.

As the previous paragraph indicated, the critical \hat{B} can be determined through the inverse of the cumulative distribution function. AIMMS supplies this function for a large number of distributions. For instance, when the underlying distribution is normal with mean 0 and standard deviation 1, then the value of \hat{B} can be found as follows:

*Use
AIMMS-supplied
function*

$$\hat{B} = \text{InverseCumulativeDistribution}(\text{Normal}(0,1), \alpha)$$

Consider the constraint $\sum_j a_j x_j \leq B$ with a stochastic right-hand side. Let $B = N(0, 1)$ and $\alpha = 0.05$. Then the value of \hat{B} based on the inverse cumulative distribution is -1.645. By requiring that $\sum_j a_j x_j \leq -1.645$, you make sure that the solution x_j is feasible for 95% of all instances of the random variable B .

Example

The following figure presents a graphical overview of the four linear probabilistic constraints with stochastic right-hand sides, together with their deterministic equivalent. The shaded areas correspond to the feasible region of $\sum_{j \in J} a_j x_j$.

*Overview of
probabilistic
constraints*

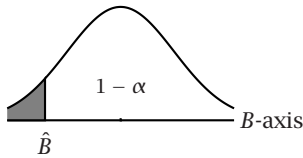
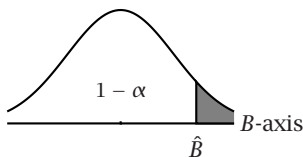
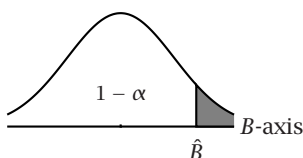
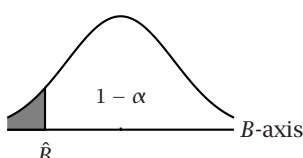
$\Pr \left[\sum_{j \in J} a_j x_j \leq B \right] \geq 1 - \alpha$ $\sum_{j \in J} a_j x_j \leq \hat{B}$	
$\Pr \left[\sum_{j \in J} a_j x_j \leq B \right] \leq \alpha$ $\sum_{j \in J} a_j x_j \geq \hat{B}$	
$\Pr \left[\sum_{j \in J} a_j x_j \geq B \right] \geq 1 - \alpha$ $\sum_{j \in J} a_j x_j \geq \hat{B}$	
$\Pr \left[\sum_{j \in J} a_j x_j \geq B \right] \leq \alpha$ $\sum_{j \in J} a_j x_j \leq \hat{B}$	

Table 6.1: Overview of linear probabilistic constraints

6.7 Summary

This chapter presented a number of techniques to transform some special models into conventional linear programming models. It was shown that some curve fitting procedures can be modeled, and solved, as linear programming models by reformulating the objective. A method to reformulate objectives which incorporate *absolute values* was given. In addition, a trick was shown to make it possible to incorporate a *minimax objective* in a linear programming model. For the case of a *fractional objective* with linear constraints, such as those that occur in financial applications, it was shown that these can be transformed into linear programming models. A method was demonstrated to specify a *range constraint* in a linear programming model. At the end of this chapter, it was shown how to reformulate constraints with a stochastic right-hand side to deterministic linear constraints.

Chapter 7

Integer Linear Programming Tricks

As in the previous chapter “Linear Programming Tricks”, the emphasis is on abstract mathematical modeling techniques but this time the focus is on *integer* programming tricks. These are not discussed in any particular reference, but are scattered throughout the literature. Several tricks can be found in [Wi90]. Other tricks are referenced directly.

This chapter

Only *linear* integer programming models are considered because of the availability of computer codes for this class of problems. It is interesting to note that several practical problems can be transformed into linear integer programs. For example, integer variables can be introduced so that a nonlinear function can be approximated by a “piecewise linear” function. This and other examples are explained in this chapter.

*Limitation to
linear integer
programs*

7.1 A variable taking discontinuous values

This section considers an example of a simple situation that cannot be formulated as a linear programming model. The value of a variable must be either zero or between particular positive bounds (see Figure 7.1). In algebraic notation:

$$x = 0 \quad \text{or} \quad l \leq x \leq u$$

This can be interpreted as two constraints that cannot both hold simultaneously. In linear programming only simultaneous constraints can be modeled.

*A jump in the
bound*

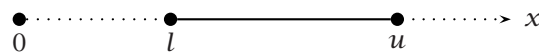


Figure 7.1: A discontinuous variable

This situation occurs when a supplier of some item requires that if an item is ordered, then its batch size must be between a particular minimum and maximum value. Another possibility is that there is a set-up cost associated with the manufacture of an item.

Applications

To model discontinuous variables, it is helpful to introduce the concept of an *indicator variable*. An indicator variable is a binary variable (0 or 1) that indicates a certain state in a model. In the above example, the indicator variable y is linked to x in the following way:

Modeling discontinuous variables

$$y = \begin{cases} 0 & \text{for } x = 0 \\ 1 & \text{for } l \leq x \leq u \end{cases}$$

The following set of constraints is used to create the desired properties:

$$\begin{aligned} x &\leq uy \\ x &\geq ly \\ y &\text{ binary} \end{aligned}$$

It is clear that $y = 0$ implies $x = 0$, and that $y = 1$ implies $l \leq x \leq u$.

7.2 Fixed costs

A fixed cost problem is another application where indicator variables are added so that two mutually exclusive situations can be modeled. An example is provided using a single-variable. Consider the following linear programming model (the sign “ \geq ” denotes either “ \leq ”, “ $=$ ”, or “ \geq ” constraints).

The model

Minimize: $C(x)$

Subject to:

$$\begin{aligned} a_i x + \sum_{j \in J} a_{ij} w_j &\geq b_i \quad \forall i \in I \\ x &\geq 0 \\ w_j &\geq 0 \quad \forall j \in J \end{aligned}$$

Where: $C(x) = \begin{cases} 0 & \text{for } x = 0 \\ k + cx & \text{for } x > 0 \end{cases}$

As soon as x has a positive value, a fixed cost is incurred. This cost function is not linear and is not continuous. There is a jump at $x = 0$, as illustrated in Figure 7.2.

In the above formulation, the discontinuous function is the objective, but such a function might equally well occur in a constraint. An example of such a fixed-cost problem occurs in the manufacturing industry when set-up costs are charged for new machinery.

Application

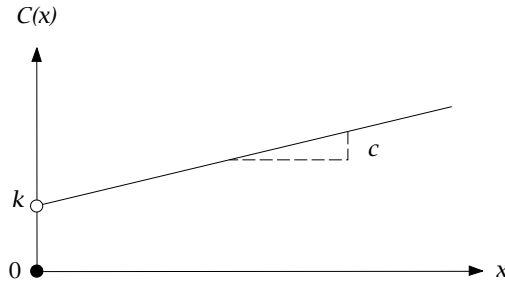


Figure 7.2: Discontinuous cost function

A sufficiently large upper bound, u , must be specified for x . An indicator variable, y , is also introduced in a similar fashion:

Modeling fixed costs

$$y = \begin{cases} 0 & \text{for } x = 0 \\ 1 & \text{for } x > 0 \end{cases}$$

Now the cost function can be specified in both x and y :

$$C^*(x, y) = ky + cx$$

The minimum of this function reflects the same cost figures as the original cost function, except for the case when $x > 0$ and $y = 0$. Therefore, one constraint must be added to ensure that $x = 0$ whenever $y = 0$:

$$x \leq uy$$

Now the model can be stated as a mixed integer programming model. The formulation given earlier in this section can be transformed as follows.

The equivalent mixed integer program

$$\begin{aligned} &\text{Minimize:} && ky + cx \\ &\text{Subject to:} && \\ & && a_i x + \sum_{j \in J} a_{ij} w_j \geq b_i \quad \forall i \in I \\ & && x \leq uy \\ & && x \geq 0 \\ & && w_j \geq 0 \quad \forall j \in J \\ & && y \text{ binary} \end{aligned}$$

7.3 Either-or constraints

Consider the following linear programming model:

The model

$$\begin{aligned} &\text{Minimize:} && \sum_{j \in J} c_j x_j \\ &\text{Subject to:} && \end{aligned}$$

Subject to:

$$\sum_{j \in J} a_{1j} x_j \leq b_1 \quad (1)$$

$$\sum_{j \in J} a_{2j} x_j \leq b_2 \quad (2)$$

$$x_j \geq 0 \quad \forall j \in J$$

Where: at least one of the conditions (1) or (2) must hold

The condition that at least one of the constraints must hold cannot be formulated in a linear programming model, because in a linear program *all* constraints must hold. Again, a binary variable can be used to express the problem. An example of such a situation is a manufacturing process, where two modes of operation are possible.

Consider a binary variable y , and sufficiently large upper bounds M_1 and M_2 , which are upper bounds on the activity of the constraints. The bounds are chosen such that they are as tight as possible, while still guaranteeing that the left-hand side of constraint i is always smaller than $b_i + M_i$. The constraints can be rewritten as follows:

*Modeling
either-or
constraints*

$$(1) \quad \sum_{j \in J} a_{1j} x_j \leq b_1 + M_1 y$$

$$(2) \quad \sum_{j \in J} a_{2j} x_j \leq b_2 + M_2 (1 - y)$$

When $y = 0$, constraint (1) is imposed, and constraint (2) is weakened to $\sum_{j \in J} a_{2j} x_j \leq b_2 + M_2$, which will always be non-binding. Constraint (2) may of course still be satisfied. When $y = 1$, the situation is reversed. So in all cases one of the constraints is imposed, and the other constraint may also hold. The problem then becomes:

Minimize:

$$\sum_{j \in J} c_j x_j$$

Subject to:

$$\sum_{j \in J} a_{1j} x_j \leq b_1 + M_1 y$$

$$\sum_{j \in J} a_{2j} x_j \leq b_2 + M_2 (1 - y)$$

$$x_j \geq 0 \quad \forall j \in J$$

$$y \text{ binary}$$

*The equivalent
mixed integer
program*

7.4 Conditional constraints

A problem that can be treated in a similar way to either-or constraints is one that contains conditional constraints. The mathematical presentation is limited to a case, involving two constraints, on which the following condition is imposed.

The model

$$\begin{aligned} \text{If } (1) \quad & \left(\sum_{j \in J} a_{1j} x_j \leq b_1 \right) \text{ is satisfied,} \\ \text{then } (2) \quad & \left(\sum_{j \in J} a_{2j} x_j \leq b_2 \right) \text{ must also be satisfied.} \end{aligned}$$

Let A denote the statement that the logical expression “Constraint (1) holds” is true, and similarly, let B denote the statement that the logical expression “Constraint (2) holds” is true. The notation $\neg A$ and $\neg B$ represent the case of the corresponding logical expressions being false. The above conditional constraint can be restated as: A implies B . This is logically equivalent to writing $(A \text{ and } \neg B) \text{ is false}$. Using the negation of this expression, it follows that $\neg(A \text{ and } \neg B)$ is true. This is equivalent to $(\neg A \text{ or } B) \text{ is true}$, using Boolean algebra. It is this last equivalence that allows one to translate the above conditional constraint into an either-or constraint.

Logical equivalence

One can observe that

$$\text{If } \left(\sum_{j \in J} a_{1j} x_j \leq b_1 \right) \text{ holds, then } \left(\sum_{j \in J} a_{2j} x_j \leq b_2 \right) \text{ must hold,}$$

Modeling conditional constraints

is equivalent to

$$\left(\sum_{j \in J} a_{1j} x_j > b_1 \right) \text{ or } \left(\sum_{j \in J} a_{2j} x_j \leq b_2 \right) \text{ must hold.}$$

Notice that the sign in (1) is reversed. A difficulty to overcome is that the strict inequality “not (1)” needs to be modeled as an inequality. This can be achieved by specifying a small tolerance value beyond which the constraint is regarded as broken, and rewriting the constraint to:

$$\sum_{j \in J} a_{1j} x_j \geq b_1 + \epsilon$$

This results in:

$$\sum_{j \in J} a_{1j} x_j \geq b_1 + \epsilon, \text{ or } \sum_{j \in J} a_{2j} x_j \leq b_2 \text{ must hold.}$$

This last expression strongly resembles the either-or constraints in the previous section. This can be modeled in a similar way by introducing a binary

variable y , a sufficiently large upper bound M on (2), and a sufficiently lower bound L on (1). The constraints can be rewritten to get:

$$\begin{aligned}\sum_{j \in J} a_{1j}x_j &\geq b_1 + \epsilon - Ly \\ \sum_{j \in J} a_{2j}x_j &\leq b_2 + M(1 - y)\end{aligned}$$

You can verify that these constraints satisfy the original conditional expression correctly, by applying reasoning similar to that in Section 7.3.

7.5 Special Ordered Sets

There are particular types of restrictions in integer programming formulations that are quite common, and that can be treated in an efficient manner by solvers. Two of them are treated in this section, and are referred to as *Special Ordered Sets* (SOS) of type 1 and 2. These concepts are due to Beale and Tomlin ([Be69]).

This section

A common restriction is that out of a set of yes-no decisions, at most one decision variable can be yes. You can model this as follows. Let y_i denote zero-one variables, then

SOS1 constraints

$$\sum_i y_i \leq 1$$

forms an example of a SOS1 constraint. More generally, when considering variables $0 \leq x_i \leq u_i$, then the constraint

$$\sum_i a_i x_i \leq b$$

can also become a SOS1 constraint by adding the requirement that at most one of the x_i can be nonzero. In AIMMS there is a constraint attribute named `Property` in which you can indicate whether this constraint is a SOS1 constraint. Note that in the general case, the variables are no longer restricted to be zero-one variables.

A general SOS1 constraint can be classified as a logical constraint and as such it can always be translated into a formulation with binary variables. Under these conditions the underlying branch and bound process will follow the standard binary tree search, in which the number of nodes is an exponential function of the number of binary variables. Alternatively, if the solver recognizes it as a SOS1 constraint, then the number of nodes to be searched can be reduced. However, you are advised to only use SOS sets if there exists a natural order relationship among the variables in the set. If your model contains multiple SOS sets, you could consider specifying priorities for some of these SOS sets.

SOS1 and performance

To illustrate how the SOS order information is used to create new nodes during the branch and bound process, consider a model in which a decision has to be made about the size of a warehouse. The size of the warehouse should be either 10000, 20000, 40000, or 50000 square feet. To model this, four binary variables x_1 , x_2 , x_3 and x_4 are introduced that together make up a SOS1 set. The order among these variables is naturally specified through the sizes. During the branch and bound process, the split point in the SOS1 set is determined by the weighted average of the solution of the relaxed problem. For example, if the solution of the relaxed problem is given by $x_1 = 0.1$ and $x_4 = 0.9$, then the corresponding weighted average is $0.1 \cdot 10000 + 0.9 \cdot 50000 = 46000$. This computation results in the SOS set being split up between variable x_3 and x_4 . The corresponding new nodes in the search tree are specified by (1) the nonzero element is the set $\{x_1, x_2, x_3\}$ (i.e. $x_4 = 0$) and (2) $x_4 = 1$ (and $x_1 = x_2 = x_3 = 0$).

SOS1 branching

Another common restriction, is that out of a set of nonnegative variables, at most two variables can be nonzero. In addition, the two variables must be adjacent to each other in a fixed order list. This class of constraint is referred to as a type SOS2 in AIMMS. A typical application occurs when a non-linear function is approximated by a piecewise linear function. Such an example is given in the next section.

SOS2 constraints

7.6 Piecewise linear formulations

Consider the following model with a separable objective function:

The model

$$\begin{aligned} \text{Minimize:} \quad & \sum_{j \in J} f_j(x_j) \\ \text{Subject to:} \quad & \sum_{j \in J} a_{ij}x_j \geq b_i \quad \forall i \in I \\ & x_j \geq 0 \quad \forall j \in J \end{aligned}$$

In the above general model statement, the objective is a *separable function*, which is defined as the sum of functions of scalar variables. Such a function has the advantage that nonlinear terms can be approximated by piecewise linear ones. Using this technique, it may be possible to generate an integer programming model, or sometimes even a linear programming model (see [Wi90]). This possibility also exists when a constraint is separable.

Separable function

Some examples of separable functions are:

Examples of separable functions

$$\begin{aligned} x_1^2 + 1/x_2 - 2x_3 &= f_1(x_1) + f_2(x_2) + f_3(x_3) \\ x_1^2 + 5x_1 - x_2 &= g_1(x_1) + g_2(x_2) \end{aligned}$$

The following examples are not:

$$x_1x_2 + 3x_2 + x_2^2 = f_1(x_1, x_2) + f_2(x_2)$$

$$1/(x_1 + x_2) + x_3 = g_1(x_1, x_2) + g_2(x_3)$$

Consider a simple example with only one nonlinear term to be approximated, namely $f(x) = \frac{1}{2}x^2$. Figure 7.3, shows the curve divided into three pieces that are approximated by straight lines. This approximation is known as piecewise linear. The points where the slope of the piecewise linear function changes (or its domain ends) are referred to as breakpoints. This approximation can be expressed mathematically in several ways. A method known as the λ -formulation is described below.

*Approximation
of a nonlinear
function*

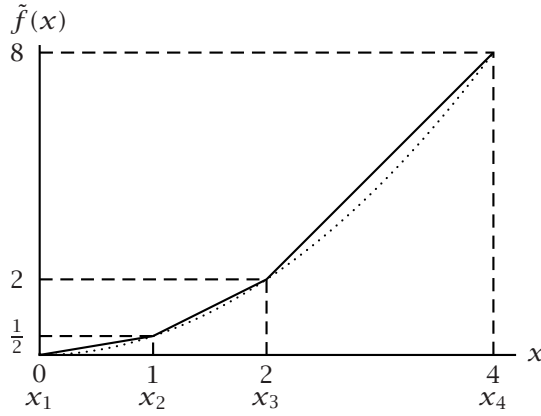


Figure 7.3: Piecewise linear approximation of $f(x) = \frac{1}{2}x^2$

Let x_1, x_2, x_3 and x_4 denote the four breakpoints along the x-axis in Figure 7.3, and let $f(x_1), f(x_2), f(x_3)$ and $f(x_4)$ denote the corresponding function values. The breakpoints are 0, 1, 2 and 4, and the corresponding function values are 0, $\frac{1}{2}$, 2 and 8. Any point in between two breakpoints is a weighted sum of these two breakpoints. For instance, $x = 3 = \frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 4$. The corresponding approximated function value $\tilde{f}(3) = 5 = \frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 8$.

Weighted sums

Let $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ denote four nonnegative weights such that their sum is 1. Then the piecewise linear approximation of $f(x)$ in Figure 7.3 can be written as:

λ -Formulation

$$\lambda_1 f(x_1) + \lambda_2 f(x_2) + \lambda_3 f(x_3) + \lambda_4 f(x_4) = \tilde{f}(x)$$

$$\lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 + \lambda_4 x_4 = x$$

$$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$$

with the added requirement that at most two adjacent λ 's are greater than zero. This requirement together with the last constraint form the SOS2 con-

straint referred to at the end of the previous section. The SOS2 constraints for all separable functions in the objective function together guarantee that the points $(x, \tilde{f}(x))$ always lie on the approximating line segments.

The added requirement that at most two adjacent λ 's are greater than zero can be modeled using additional binary variables. This places any model with SOS2 constraints in the realm of integer (binary) programming. For this reason, it is worthwhile to investigate when the added adjacency requirements are redundant. Redundancy is implied by the following conditions.

Adjacency requirements sometimes redundant

1. The objective is to *minimize* a separable function in which all terms $f_j(x_j)$ are *convex* functions.
2. The objective is to *maximize* a separable function in which all terms $f_j(x_j)$ are *concave* functions.

A function is convex when the successive slopes of the piecewise linear approximation are nondecreasing, and concave if these slopes are non-increasing. A concave cost curve can represent an activity with economies of scale. The unit costs decrease as the number of units increases. An example is where quantity discounts are obtained.

Convexity and concavity

The adjacency requirements are no longer redundant when the function to be approximated is non-convex. In this case, these adjacency requirements must be formulated explicitly in mathematical terms.

The case of a non-convex function

In AIMMS you do not need to formulate the adjacency requirements explicitly. Instead you need to specify `sos2` in the property attribute of the constraint in which the λ 's are summed to 1. In this case, the solver in AIMMS guarantees that there will be at most two adjacent nonzero λ 's in the optimal solution. If the underlying minimization model is convex, then the linear programming solution will satisfy the adjacency requirements. If the model is not convex, the solver will continue with a mixed integer programming run.

SOS2 in AIMMS

7.7 Elimination of products of variables

This section explains a method for linearizing constraints and objective functions in which the products of variables are incorporated. There are numerous applications that give rise to nonlinear constraints and the use of integer variables. These problems become very difficult, if not impossible, to solve.

This section

In general, a product of two variables can be replaced by one new variable, on which a number of constraints is imposed. The extension to products of more than two variables is straightforward. Three cases are distinguished. In the third case, a separable function results (instead of a linear one) that can then be approximated by using the methods described in the previous section.

*Replacing
product term*

Firstly, consider the binary variables x_1 and x_2 . Their product, x_1x_2 , can be replaced by an additional binary variable y . The following constraints force y to take the value of x_1x_2 :

*Two binary
variables*

$$\begin{aligned} y &\leq x_1 \\ y &\leq x_2 \\ y &\geq x_1 + x_2 - 1 \\ y &\text{ binary} \end{aligned}$$

Secondly, let x_1 be a binary variable, and x_2 be a continuous variable for which $0 \leq x_2 \leq u$ holds. Now a continuous variable, y , is introduced to replace the product $y = x_1x_2$. The following constraints must be added to force y to take the value of x_1x_2 :

*One binary and
one continuous
variable*

$$\begin{aligned} y &\leq ux_1 \\ y &\leq x_2 \\ y &\geq x_2 - u(1 - x_1) \\ y &\geq 0 \end{aligned}$$

The validity of these constraints can be checked by examining Table 7.1 in which all possible situations are listed.

x_1	x_2	x_1x_2	constraints	imply
0	$w : 0 \leq w \leq u$	0	$y \leq 0$ $y \leq w$ $y \geq w - u$ $y \geq 0$	$y = 0$
1	$w : 0 \leq w \leq u$	w	$y \leq u$ $y \leq w$ $y \geq w$ $y \geq 0$	$y = w$

Table 7.1: All possible products $y = x_1x_2$

Thirdly, the product of two continuous variables can be converted into a separable form. Suppose the product x_1x_2 must be transformed. First, two (continuous) variables y_1 and y_2 are introduced. These are defined as:

Two continuous variables

$$\begin{aligned} y_1 &= \frac{1}{2}(x_1 + x_2) \\ y_2 &= \frac{1}{2}(x_1 - x_2) \end{aligned}$$

Now the term x_1x_2 can be replaced by the separable function

$$y_1^2 - y_2^2$$

which can be approximated by using the technique of the preceding section. Note that in this case the non-linear term can be eliminated at the cost of having to approximate the objective. If $l_1 \leq x_1 \leq u_1$ and $l_2 \leq x_2 \leq u_2$, then the bounds on y_1 and y_2 are:

$$\frac{1}{2}(l_1 + l_2) \leq y_1 \leq \frac{1}{2}(u_1 + u_2) \quad \text{and} \quad \frac{1}{2}(l_1 - u_2) \leq y_2 \leq \frac{1}{2}(u_1 - l_2)$$

The product x_1x_2 can be replaced by a single variable z whenever

Special case

- the lower bounds l_1 and l_2 are nonnegative, and
- one of the variables is not referenced in any other term except in products of the above form.

Assume, that x_1 is such a variable. Then substituting for z and adding the constraint

$$l_1x_2 \leq z \leq u_1x_2$$

is all that is required to eliminate the nonlinear term x_1x_2 . Once the model is solved in terms of z and x_2 , then $x_1 = z/x_2$ when $x_2 > 0$ and x_1 is undetermined when $x_2 = 0$. The extra constraints on z guarantee that $l_1 \leq x_1 \leq u_1$ whenever $x_2 > 0$.

7.8 Summary

In practical applications, integer linear programming models often arise when discontinuous restrictions are added to linear programs. In this chapter, some typical examples have been shown, along with methods to reformulate them as integer programs. The binary “indicator variable” plays an important role. With the aid of binary variables it is possible to model discontinuities in variables or objectives, as well as either-or constraints and conditional constraints. By conducting a piecewise linear approximation of a nonlinear program, containing a separable nonlinear objective function, it may be possible to generate a linear programming model or perhaps an integer programming model. At the end of the chapter, methods for eliminating products of variables are described.

Part III

Basic Optimization Modeling Applications

Chapter 8

An Employee Training Problem

This chapter introduces a personnel planning problem and its corresponding multi-period model. The model includes a (stock) balance constraint which is typical in multi-period models involving state and control type decision variables. A time lag notation is introduced for the backward referencing of time periods. Initially, a simplified formulation of the model is solved and it is shown that rounding a fractional linear programming solution can be a good alternative to using an *integer* programming solver. Finally, the full model complete with random variables is considered, and an approach based on the use of probabilistic constraints is presented.

This chapter

Problems of this type can be found in, for instance, [Wa75], [Ep87], and [Ch83].

References

Linear Program, Integer Program, Control-State Variables, Rounding Heuristic, Probabilistic Constraint, Worked Example.

Keywords

8.1 Hiring and training of flight attendants

Consider the following personnel planning problem. The personnel manager of an airline company must decide how many new flight attendants to hire and train over the next six months. Employment contracts start at the beginning of each month. Due to regulations each flight attendant can only work up to 150 hours a month. Trainees require two months of training before they can be put on a flight as regular flight attendants. During this time a trainee is effectively available for 25 hours a month. Due to limited training capacity, the maximum number of new trainees each month is 5.

Hiring and training

Throughout the year, flight attendants quit their jobs for a variety of reasons. When they do, they need to notify the personnel manager one month in advance. The uncertainty in the number of flight attendants leaving the company in future months, makes it difficult for the personnel manager to make long term plans. In the first part of this chapter, it is assumed that all the leave notifications for the next six months are known. This assumption is relaxed at the end of the chapter.

Job notification

The airline problem is studied using the following data. At the beginning of December, 60 flight attendants are available for work and no resignations were received. Two trainees were hired at the beginning of November and none in December. The cost of a flight attendant is \$5100 a month while the cost of a trainee is \$3600 a month. The flight attendant requirements in terms of flight hours are given in Table 8.1 for the months January to June. The table also includes the (known) number of flight attendants who will hand in their resignation, taking effect one month thereafter.

Example data

months	required hours	resignations
January	8,000	2
February	9,000	
March	9,800	2
April	9,900	
May	10,050	1
June	10,500	

Table 8.1: Flight attendant requirements and resignations

8.2 Model formulation

The main decision to be made by the personnel manager is the number of trainees to be hired at the beginning of each month. Prior to this decision, both the number of trainees and the number of flight attendants are unknown, and these represent two types of decision variables. An important observation is that once the decision regarding the trainees has been made then the number of available flight attendants is determined. This leads to the distinction between *control variables* and *state variables*. Control variables are independent decision variables while state variables are directly dependent on the control variables. In mathematical models both are considered as decision variables.

Control and state variables

The distinction between control and state variables often leads to the use of so-called *balance constraints*. These are often equalities defining the state variable in terms of the control variables. Such constraints typically occur in multi-period models in which state variables such as stocks of some sort are recorded over time.

Balance constraints

A verbal model statement of the problem is based on the notion of a balance constraint for the number of attendants plus a constraint reflecting the personnel requirement in terms of flight hours.

Verbal model statement

Minimize: total personnel costs,

Subject to:

- for all months: the number of flight attendants equals the number of flight attendants of the previous month minus the number of flight attendants that resigned the previous month plus the number of trainees that have become flight attendants,
- for all months: the number of attendant flight hours plus the number of trainee hours must be greater than or equal to the monthly required flight attendant hours.

The verbal model statement of the personnel planning problem can be specified as a mathematic model using the following notation.

Notation

Index:

t time periods (months) in the planning interval

Parameters:

c monthly cost of one flight attendant
 d monthly cost of one trainee
 u monthly number of hours of one flight attendant
 v monthly number of hours of one trainee
 m maximum number of new trainees each month
 r_t required flight attendant hours in t
 l_t number of flight attendants resigning in t

Variables:

x_t number of flight attendants available in t
 y_t number of trainees hired in t

It is important to explicitly specify the precise time that you assume the decisions take place. It makes a difference to your model constraints whether something takes place at the beginning or at the end of a period. A good rule is to be consistent for all parameters and variables in your model. Throughout this section it is assumed that all events take place at the beginning of each monthly period.

Time specification

The flight attendant balance constraint is a straightforward book keeping identity describing how the number of flight attendants varies over time.

Attendant balance

$$x_t = x_{t-1} - l_{t-1} + y_{t-2} \quad \forall t$$

The index t refers to periods in the planning period. The time-lag references $t - 1$ and $t - 2$ refer to one and two periods prior to the current period t , respectively. When the current period happens to be the first period, then the references x_{t-1} and y_{t-2} refer to the past. They do not represent decision variables but instead they define the initial conditions of the problem. When

Lag and lead notation ...

using the balance equation, care must be taken to ensure that it is only used for time periods within the planning interval. In this chapter the planning interval is specified by the months January through June.

In AIMMS there is a special type of set called *horizon*, which is especially designed for time periods. In each horizon, there is a *past*, a *planning interval* and a *beyond*. Any variable that refers to a time period outside the planning interval (i.e. past and beyond) is automatically assumed to be data (fixed). This feature simplifies the representation of balance constraints in AIMMS. ... in AIMMS

The personnel requirement constraint is not stated in terms of persons but in term of hours. This requires the translation from persons to hours. Requirement constraint

$$ux_t + v(y_t + y_{t-1}) \geq r_t \quad \forall t$$

Note that the trainees who started their training at the beginning of the current month or the previous month only contribute $v < u$ hours to the total availability of flight attendant hours.

The objective function is to minimize total personnel cost. Objective function

Minimize:

$$\sum_t cx_t + d(y_t + y_{t-1})$$

The following mathematical statement summarizes the model. Model summary

Minimize:

$$\sum_t cx_t + d(y_t + y_{t-1})$$

Subject to:

$$\begin{aligned} x_t &= x_{t-1} - l_{t-1} + y_{t-2} & \forall t \\ ux_t + v(y_t + y_{t-1}) &\geq r_t & \forall t \\ x_t &\geq 0, \text{ integer} & \forall t \\ 0 \leq y_t &\leq m, \text{ integer} & \forall t \end{aligned}$$

8.3 Solutions from conventional solvers

The above model is an integer programming model since all decision variables must assume integer values. You may relax this requirement for the variable x_t (the number of flight attendants) because in the balance constraint defining x_t in terms of y_t (the number of new trainees), this value of x_t is automatically integer when y_t is integer. Integer requirement

The model, when instantiated with the small data set, can easily be solved with any integer programming solver. The solution found by such a solver is displayed in Table 8.2 and its corresponding optimal objective function value is \$2,077,500.

Optimal integer solution

	flight attendants	trainees
November		2
December	60	
January	62	5
February	60	2
March	65	2
April	65	4
May	67	
June	70	

Table 8.2: The optimal integer solution

As the sizes of the underlying data sets are increased, it may become impractical to find an optimal integer solution using a conventional integer programming solver. Under these conditions, it is not atypical that the conventional solver finds one or more suboptimal solutions in a reasonable amount of computing time, but subsequently spends a lot of time trying to find a solution which is better than the currently best solution. In practical applications, this last best solution may very well be good enough. One way to obtain such suboptimal solutions is to specify optimality tolerances.

Suboptimal integer solution

Whenever you are solving large-scale integer programming models, you are advised to use solution tolerance settings in an effort to avoid long computational times. In AIMMS you can specify both a *relative optimality tolerance* and an *absolute optimality tolerance*. The relative optimality tolerance `MIP_relative_optimality_tolerance` is a fraction indicating to the solver that it should stop as soon as an integer solution within 100 times `MIP_relative_optimality_tolerance` percent of the global optimum has been found. Similarly to the relative optimality tolerance, the absolute optimality tolerance `MIP_absolute_optimality_tolerance` is a number indicating that the solver should terminate as soon as an integer solution is within `MIP_absolute_optimality_tolerance` of the global optimum.

Setting optimality criteria

Another feature available in AIMMS that can be used to reduce the solving time for large integer programs is *priority setting*. By assigning priority values to integer variables, you directly influence the order in which variables are fixed during the search by a solver. For instance, by setting low positive priority values for the γ_f variables (the number of new trainees to be hired), and let-

Setting priorities

ting these values increase as time progresses, the branch and bound solution method will decide on the number of trainees to be hired in the same order as in the set of months. Experience has demonstrated that setting integer priorities to mimic a natural order of decisions, is likely to decrease computational time. This is particularly true when the size of the data set grows.

8.4 Solutions from rounding heuristics

By *relaxing* (i.e. neglecting) the integer requirement on the variables x_t and y_t , the personnel planning model becomes a linear program. In general, linear programs are easier to solve than integer programs and this is particularly true with large data instances. However, the optimal solution is not necessarily integer-valued. The question then arises whether a simple rounding of the linear programming solution leads to a good quality suboptimal integer solution. This question is investigated using the optimal linear programming solution presented in Table 8.3.

*Examining
relaxed solution*

	flight attendants	trainees
November		2
December	60	
January	62.00	4.76
February	60.00	2.24
March	64.76	1.20
April	65.00	4.80
May	66.20	
June	70.00	

Table 8.3: The optimal LP solution

Rounding the values of both x_t (the flight attendants) and y_t (the trainees) in the relaxed solution violates the balance constraint. Simply rounding the y_t variables upwards and re-calculating the x_t variables does result in a feasible solution for this data set.

Rounding up

A tighter variant of the heuristic of the previous paragraph is to round downwards as long as there is a reserve of at least one trainee, and to round upwards when there is no such a reserve. The solution obtained from both rounding variants are contained in Table 8.4. Note that none of the two rounded solutions are as good as the optimal integer solution. A skeleton algorithm for the second variant can be written as

*Rounding with
reserves*

```

FOR (t) DO
  IF reserve < 1
    THEN y(t) := ceil[y(t)]
    ELSE y(t) := floor[y(t)]
  ENDIF
  Update reserve
ENDFOR

```

This skeleton algorithm can form the basis for an implementation in AIMMS. Of course, the above heuristic is only one of many possible rounding procedures. A more robust heuristic should register not only the reserve created by rounding, but also the number of extra training hours required. However, this refinement is trickier than you might think at first!

	LP		Rounded up		Rounded with reserves	
	x_t	y_t	x_t	y_t	x_t	y_t
November		2		2		2
December	60		60		60	
January	62.00	4.76	62	5	62	5
February	60.00	2.24	60	3	60	3
March	64.76	1.20	65	2	65	1
April	65.00	4.80	66	5	66	5
May	66.20		68		67	
June	70.00		72		71	
Total costs	2,072,196		2,112,300		2,094,900	

Table 8.4: The rounded solutions

8.5 Introducing probabilistic constraints

Until now, it was assumed that the number of resignations was known in advance for each month in the planning interval. Without this assumption, the number of resignations each month is not a parameter but instead a random variable with its own distribution. To obtain an insight into the distribution, it is necessary to statistically analyse the resignation data. The analysis should be based on both historic records and information about personnel volatility in the current market place.

*Uncertain
resignations*

Assume that such a statistical data analysis concludes that resignation data is independently normally distributed with means and variances as presented in Table 8.5. This table also contains critical resignation levels used in the following probabilistic analysis.

*Observed
distributions*

	mean	variance	$\alpha = 0.01$	$\alpha = 0.02$	$\alpha = 0.05$
January	1.15	0.57	2.48	2.32	2.09
February	0.80	0.58	2.15	1.99	1.75
March	1.30	0.73	3.00	2.80	2.50
April	1.45	0.75	3.19	2.99	2.68
May	0.85	0.50	2.01	1.88	1.67
June	1.40	0.79	3.24	3.02	2.70

Table 8.5: Normally distributed resignation data and critical values

In light of the uncertainty in the resignation data, the personnel manager needs to hire enough extra trainees to make sure that there are enough flight attendants under most circumstances. It is not economical to cover the unlikely extreme scenario's in which the number of resignations is far beyond the average. Eliminating the extreme scenario's can be accomplished through the use of probabilistic constraints.

Unlikely scenarios

Consider a slightly modified version of the flight attendant balance constraint, in which the expression on the left can be interpreted as the exact number of flight attendants that can leave without causing a shortage or surplus.

Probabilistic constraints

$$x_{t-1} - y_{t-2} - x_t = l_{t-1} \quad \forall t$$

By aiming for a surplus, it is possible to avoid a shortage under most resignation scenarios. The flight attendant balance constraint can be altered into any of the following two equivalent probabilistic constraints:

$$\begin{aligned} \Pr[x_{t-1} - y_{t-2} - x_t \geq l_{t-1}] &\geq 1 - \alpha & \forall t \\ \Pr[x_{t-1} - y_{t-2} - x_t \leq l_{t-1}] &\leq \alpha & \forall t \end{aligned}$$

The value of α is assumed to be small, indicating there is frequently a surplus (the first form) or there is infrequently a shortage (the second form).

As explained in Section 6.6, both probabilistic constraints have the same deterministic equivalent, namely

Deterministic equivalence

$$x_{t-1} - y_{t-2} - x_t \geq \bar{l}_{t-1} \quad \forall t$$

where \bar{l}_t could be one of the critical values from Table 8.5 depending on the level of α selected by management.

The new model with the deterministic equivalent of the probabilistic constraints can now be summarized as follows.

Summary of new model

Minimize:

$$\sum_t cx_t + d(y_t + y_{t-1})$$

Subject to:

$$\begin{aligned} x_t &\leq x_{t-1} - \bar{l}_{t-1} + y_{t-2} & \forall t \\ ux_t + v(y_t + y_{t-1}) &\geq r_t & \forall t \\ x_t, y_t &\geq 0, \text{ integer} & \forall t \end{aligned}$$

This new model strongly resembles the model in Section 8.2. Note that the parameter values of \bar{l}_t are likely to be fractional in the above version. This implies that the integer requirement on both x_t and y_t are necessary, and that the above balance constraints for flight attendants are likely not to be tight in the optimal solution.

*Integer
requirement*

8.6 Summary

In this chapter, a multi-period planning model was developed to determine the hiring and training requirements for an airline company. Since the number of new flight attendants must be integer, the resulting model is an integer programming model. For large data sets, one option to obtain an integer solution is to round an LP solution. If you understand the model, it is possible to develop a sensible rounding procedure. Such a procedure might be considerably more efficient than using an integer programming solver. Towards the end of the chapter, the uncertainty in the number of flight attendants resigning during the planning interval was modeled. By modifying the resulting probabilistic constraints to deterministic constraints, an ordinary integer optimization model was found.

Exercises

- 8.1 Implement the mathematical program described at the end of Section 8.2 using the example data provided in Section 8.1. Verify that the optimal integer solution produced with AIMMS is the same as the solution provided in Table 8.2.
- 8.2 Solve the mathematical program as a linear program (by using `rmip` as the mathematical program type), and implement the rounding heuristic described in Section 8.4. Verify that your findings coincide with the numbers displayed in Table 8.4.
- 8.3 Extend the mathematical program to include probabilistic constraints using the observed distribution data as described in Section 8.5 and compare the results for the different α -values as presented in Table 8.5.

Chapter 9

A Media Selection Problem

This chapter introduces a simplified media selection problem and formulates it as a binary programming model. An initial model is extended to include various strategic preference specifications and these are implemented by adding logical constraints. The problem is illustrated using a worked example and its integer solutions are reported. At the end of the chapter the problem is described as a *set covering problem*. The two related binary models of *set partitioning* and *set packing models* are also discussed in general terms.

This chapter

Examples of media selection problems are found in the marketing and advertising literature. Two references are [Ba66] and [Ch68a].

References

Integer Program, Logical Constraint, Worked Example.

Keywords

9.1 The scheduling of advertising media

Optimization is used in the field of marketing to optimally allocate advertising budgets between possible advertising outlets. These problems are known as media selection problems.

Media selection problems

Consider a company which wants to set up an advertising campaign in preparation for the introduction of a new product. Several types of audiences have been identified as target audiences for the new product. In addition, there is a selection of media available to reach the various targets. However, there is no medium that will reach all audiences. Consequently, several media need to be selected at the same time in order to cover all targets. The company wants to investigate various strategic advertising choices. The goal is *not* to stay within an a priori fixed budget, but to minimize the total cost of selecting media for each of the strategic choices.

Problem description

This chapter illustrates the problem using a small data set involving six target audiences (labeled type 1 through type 6) and eight potential medias. The data is contained in Table 9.1. The media descriptions are self-explanatory. The crosses in the table indicate which target audiences can be reached by

Example

each particular medium. Note that a cross does not say anything about the effectiveness of the medium. The right hand column gives the cost to use a particular medium. The table is deliberately small for simplicity reasons. In practical applications both the data set and the reality behind the scheduling problem is more extensive.

media	audience						costs [\$]
	type 1	type 2	type 3	type 4	type 5	type 6	
Glossy magazine	×			×			20,000
TV late night		×	×				50,000
TV prime time		×				×	60,000
Billboard train	×					×	45,000
Billboard bus			×				30,000
National paper				×		×	55,000
Financial paper		×			×		60,000
Regional paper	×				×		52,500

Table 9.1: Reachability of audiences by media

9.2 Model formulation

The aim is to construct a model to determine which media should be selected so that all audiences are reached. It does not matter if an audience is covered more than once, as long as it is covered at least once. Moreover, the company does not wish to spend more money on the campaign than necessary. The objective function and constraints are expressed in the following qualitative model formulation:

Verbal model

Minimize: *total campaign costs,*

Subject to:

for all audience types: the number of times an audience type is covered must be greater than or equal to one.

The above verbal model statement can be specified as a mathematical model using the following notation.

Notation

Indices:

t *target audiences*
 m *advertising media*

Parameters:

N_{tm} *incidence: audience t is covered by medium m*
 c_m *cost of selecting advertising medium m*

Variables:

x_m *binary, indicating whether advertising medium m is selected*

Advertising media should be selected to ensure that all audiences are reached at least once. This is guaranteed by the following covering constraint.

Covering constraint

$$\sum_m N_{tm} x_m \geq 1 \quad \forall t$$

The objective function is to minimize the cost of covering all target audiences at least once.

Objective function

Minimize:

$$\sum_m c_m x_m$$

The following mathematical statement summarizes the model.

Model summary

Minimize:

$$\sum_m c_m x_m$$

Subject to:

$$\begin{aligned} \sum_m N_{tm} x_m &\geq 1 & \forall t \\ x_m &\in \{0, 1\} & \forall m \end{aligned}$$

The problem is a binary programming model since all decision variables are binary. Using the terminology introduced in Chapter 2.2, it is also a zero-one programming problem.

The small model instance provided in this chapter can easily be solved using conventional integer programming code. Table 9.2 provides the solution values for both the integer program and the linear program. In the case of the latter solution, unlike in Chapter 8, it does not make sense to round up or down. The cost of the campaign amounts to \$155,000 for the integer solution, and \$150,000 for the (unrealistic) linear programming solution. Note that the audience of type 1 is covered twice in the integer solution, while all other audiences are reached once.

Model results

Advertising media	x_{IP}	x_{LP}
Glossy magazine	1	0.5
TV late night		
TV prime time		
Billboard train	1	0.5
Billboard bus	1	1.0
National paper		0.5
Financial paper	1	1.0
Regional paper		

Table 9.2: Optimal solution values for integer and linear program

9.3 Adding logical conditions

Logical relationships between different decisions or states in a model can be expressed through logical constraints. In the media selection problem, logical constraints can be imposed relatively simply because the decision variables are already binary. Some modeling tricks for integer and binary programming model were introduced in Chapter 7. This section provides some additional examples of modeling with logical conditions.

Logical constraints

Suppose the marketing manager of the company decides that the campaign should, in all cases, incorporate some TV commercials. You can model this condition as follows.

Must include television commercials

$$x_{TV \text{ late night}} + x_{TV \text{ prime time}} \geq 1$$

This constraint excludes the situation where both $x_{TV \text{ late night}}$ and $x_{TV \text{ prime time}}$ are zero. When this constraint is added to the model, the optimal solution includes late night TV commercials as well as advertisements in national and regional newspapers for the advertising campaign. The campaign costs increase to \$157,500.

Suppose that if a billboard media is selected, then a television media should also be selected. Perhaps the effects of these media reinforce each other. A precise statement of this condition in words is:

If billboard then television

- *If at least one of the billboard possibilities is selected, then at least one of the possibilities for TV commercials must be selected.*

The following AIMMS constraint can be used to enforce this condition.

$$\begin{aligned} x_{TV \text{ late night}} + x_{TV \text{ prime time}} &\geq x_{\text{Billboard train}} \\ x_{TV \text{ late night}} + x_{TV \text{ prime time}} &\geq x_{\text{Billboard bus}} \end{aligned}$$

Note that these inequalities still allow the inclusion of TV commercials even if no billboard medias are selected.

Next, consider the following condition which imposes a one-to-one relationship between billboards and television.

*Billboard if
and only if
television*

- *If at least one of the billboard possibilities is selected, then at least one of the possibilities for TV commercials must be selected, and if at least one of the possibilities for TV commercials is selected, then at least one of the billboard possibilities must be selected.*

As this condition consists of the condition from the previous section plus its converse, its formulation is as follows.

$$\begin{aligned}x_{\text{TV late night}} + x_{\text{TV prime time}} &\geq x_{\text{Billboard train}} \\x_{\text{TV late night}} + x_{\text{TV prime time}} &\geq x_{\text{Billboard bus}} \\x_{\text{Billboard train}} + x_{\text{Billboard bus}} &\geq x_{\text{TV late night}} \\x_{\text{Billboard train}} + x_{\text{Billboard bus}} &\geq x_{\text{TV prime time}}\end{aligned}$$

After solving the model with these inequalities, the glossy magazine, TV commercials at prime time, billboards at bus-stops, and advertisements in regional newspapers are selected for the campaign. The campaign cost has increased to \$162,500. Just like the initial integer solution, the audience of type 1 has been covered twice.

Consider a condition that prevents the selection of any billboard media if prime time TV commercials are selected. A verbal formulation of this condition is:

*If television
prime time then
no billboards*

- *If TV commercials at prime time are selected then no billboards should be selected for the campaign.*

Note that, where the previous inequalities implied the selection of particular media, this condition excludes the selection of particular media. The above statement can be modeled by adding a single logical constraint.

$$x_{\text{Billboard train}} + x_{\text{Billboard bus}} \leq 2(1 - x_{\text{TV prime time}})$$

Note that if $x_{\text{TV prime time}}$ is equal to 1, then both $x_{\text{Billboard train}}$ and $x_{\text{Billboard bus}}$ must be 0. Adding this constraint to the media selection model and solving the model yields an optimal integer solution in which the glossy magazine, late night TV commercials, billboards at railway-stations, and advertisement in regional newspapers are selected for the campaign. The corresponding campaign cost increase to \$167,500.

Suppose that the marketing manager wants the financial paper to be included in the campaign whenever both late night TV commercials and the glossy magazine are selected. The condition can be stated as follows.

If late night television and magazine then financial paper

- *If late night TV commercials and the glossy magazine are selected then the financial paper should be selected for the campaign.*

This condition can be incorporated into the model by adding the following logical constraint.

$$x_{\text{Financial paper}} \geq x_{\text{TV late night}} + x_{\text{Glossy magazine}} - 1$$

Note that this constraint becomes $x_{\text{Financial paper}} \geq 1$ if both $x_{\text{TV late night}}$ and $x_{\text{Glossy magazine}}$ are set to 1. After adding this constraint to the model, the advertisements in regional newspapers from the previous solution are exchanged for advertisements in the financial paper, and the corresponding campaign cost increases to \$175,000. Now, audiences of type 1 and 2 are covered twice.

The final extension to the model is to add a constraint on the audiences. In the last solution, the number of audiences that are covered twice is equal to two. The marketing manager has expressed his doubts on the reliability of the reachability information, and he wants a number of audience types to be covered more than once. Specifically, he wants the following.

At least three audiences should be covered more than once

- *At least three audience types should be covered more than once.*

To formulate the above logical requirement in mathematical terms an additional binary variable y_t is introduced for every audience type t . This variable can only be one when its associated audience t is covered more than once. The sum of all y_t variables must then be greater than or equal to three. Thus, the model is extended with the following variables and constraints.

$$\begin{aligned} 2y_t &\leq \sum_m N_{tm}x_m & \forall t \\ \sum_t y_t &\geq 3 \\ y_t &\in \{0, 1\} \end{aligned}$$

Note that the expression $\sum_m N_{tm}x_m$ denotes the number of times the audience of type t is covered, and must be at least two for y_t to become one. When solving the media selection model with this extension, all media except prime time TV commercials and advertisements in the national paper and the regional papers are selected. The audiences of type 1, 2 and 3 are covered twice, and the total campaign cost is \$205,000.

9.4 Set covering and related models

The media selection problem can be considered to be a *set covering* problem. A general statement of a set covering problem follows. Consider a set $S = \{s_1, s_2, \dots, s_n\}$ and a set of sets U which consists of a number of subsets of S . An example would be

Set covering

$$S = \{s_1, s_2, s_3, s_4, s_5, s_6\} \text{ and}$$

$$U = \{u_1, u_2, u_3, u_4\} = \{\{s_1, s_2\}, \{s_3, s_4, s_6\}, \{s_2, s_3, s_5\}, \{s_5, s_6\}\}$$

Let each of these subsets of S have an associated cost, and consider the objective to determine the least-cost combination of elements of U such that each element of S is contained in this combination at least once. Every combination which contains each element of S is called a *cover*. In this example, $\{u_1, u_2, u_4\}$, $\{u_1, u_2, u_3\}$ and $\{u_1, u_2, u_3, u_4\}$ represent the only covers. It is not difficult to determine the least expensive one. However, when the sets S and U are large, solving an integer program becomes a useful approach.

In order to specify an appropriate model, the binary decision variable y_u must be defined.

Notation

$$y_u = \begin{cases} 1 & \text{if } u \in U \text{ is part of the cover} \\ 0 & \text{otherwise} \end{cases}$$

Furthermore coefficients a_{su} must be introduced.

$$a_{su} = \begin{cases} 1 & \text{if } s \in S \text{ is contained in } u \in U \\ 0 & \text{otherwise} \end{cases}$$

When the costs are defined to be c_u for $u \in U$, the model statement becomes:

The integer program

Minimize:

$$\sum_{u \in U} c_u y_u \quad (\text{combination costs})$$

Subject to:

$$\sum_{u \in U} a_{su} y_u \geq 1 \quad \forall s \in S$$

$$y_u \text{ binary} \quad \forall u \in U$$

Note that all constraint coefficients and decision variables have a value of zero or one. Only the cost coefficients can take arbitrary values. For the special case of uniform cost coefficients, the objective becomes to minimize the number of members of U used in the cover.

When all elements of S must be covered exactly once, the associated problem is termed a *set partitioning problem*. As the name suggests, the set S must now be partitioned at minimum cost. The corresponding integer programming model is similar to the above model, except for the signs of the constraints. These are “=” rather than “ \geq ”.

The set partitioning problem

When all elements of S can be covered at most once, the associated problem is termed a *set packing problem*. The corresponding integer programming model is similar to the model stated previously, except for two changes. The signs of the constraints are “ \leq ” rather than “ \geq ”, and the direction of optimization is “maximize” instead of “minimize”.

The set packing problem

There are several applications which can be essentially classified as covering, partitioning or packing models.

Applications

1. If audience types are considered to be members of the set S , and advertising media members of the class U , you obtain the media selection problem which is an example of set covering.
2. Consider an airline crew scheduling problem where flights are members of S , and “tours” (combinations of flights which can be handled by a single crew) are members of the set U . Then, depending on whether crews are allowed to travel as passengers on a flight, either a set covering or a set partitioning model arises.
3. Let the set S contain tasks, and let the set U contain all combinations of tasks that can be performed during a certain period. Then, if each task needs to be performed only once, a set partitioning problem arise.
4. Finally, if the set S contains cities, and the class U contains those combinations of cities that can be served by, for instance a hospital (or other services such as a fire department or a university), then a set covering model can determine the least cost locations such that each city is served by this hospital.

9.5 Summary

In this chapter a media selection problem was introduced and formulated as a binary programming model. An initial model was extended by including a variety of logical constraints to represent various advertising strategies. The optimal objective function value and corresponding integer solution were reported for each subsequent model. At the end of the chapter, the media selection problem was described as a *set covering problem*. The related *set partitioning* and *set packing* problems were discussed in general terms.

Exercises

- 9.1 Implement the initial mathematical program described in Section 9.2 using the example data of Table 9.1. Solve the model as a linear program and as an integer program, and verify that the optimal solutions produced with AIMMS are the same as the two optimal solutions presented in Table 9.2.
- 9.2 Extend the mathematical program to include the logical constraints described in Section 9.3, and verify that the objective function values (the total campaign cost figures) produced with AIMMS are the same as the ones mentioned in the corresponding paragraphs.
- 9.3 Formulate the following requirements as constraints in AIMMS.
- If at least one of the billboard possibilities is selected, then both of the possibilities for TV commercials must be selected.
 - At least five of the six audience types need to be covered.
 - Again, at least five of the six audience types need to be covered. If, however, not all six audience types are covered, then either the regional paper or the national paper should be selected.

Develop for each requirement a separate experiment in which you either modify or extend the initial mathematical program described in Section 9.2. Verify for yourself that the integer solution correctly reflects the particular requirement.

Chapter 10

A Diet Problem

This chapter introduces a simplified diet problem with an example small data set. The problem is transformed into an integer programming model with range constraints. The main purpose of the chapter is to illustrate the use of measurement units to achieve data consistency and to improve data communication. Reference is made to the special features in AIMMS that support unit analysis. One feature is the availability of unit-valued parameters. These parameters are useful when indexed identifiers contain individual entries measured in different units. Another feature is the availability of unit conventions that allow users with different backgrounds to view and enter data in their own choice of measurement units without having to change either the model or its data.

This chapter

The example in this chapter is based on two articles that appeared in OR/MS Today ([Bo93, Er94]). Problems of this type can also be found in, for instance, [Ch83] and [Wa75].

References

Integer Program, Measurement Units, Worked Example.

Keywords

10.1 Example of a diet problem

The example discussed in this chapter is a McDonald's diet problem. It belongs to the class of blending problems in which various feed items (for animals) or food items (for humans) are put together to form a diet. In most applications there are weight, nutrition, and taste requirements, and clearly cost minimization is an objective.

Diet problems in general

The McDonald's diet problem has been used in popular literature as an example for building an introductory optimization model. The McDonald's situation is familiar, and the problem structure is simple enough for translation into a mathematical model. In addition, McDonald's provides a brochure with detailed nutritional information for every item on the menu.

McDonald's

The example considers a small data set, which includes 9 different food types and 4 different nutrients. The 9 food types form a small but representative selection of the McDonald's menu. The 4 nutrients are calories, protein, fat, and carbohydrates. The goal is to determine a daily diet to cover the afternoon and the evening meals. Table 10.1 contains the nutritional values for each food type, nutritional requirements, food prices, and bounds on individual servings.

Example

	Calo- ries [kcal]	Pro- tein [gram]	Fat [gram]	Carbo- hydrates [gram]	max. ser- vings	Price [Hfl]
Big Mac	479	25	22	44	2	5.45
Quarter Pounder	517	32.4	25	40.4	2	4.95
Vegetable Burger	341	11.7	10.6	50	2	3.95
French Fries	425	5	21	54	2	1.95
Salad	54	4	2	5	2	3.95
Lowfat Milk	120	9	4	12	2	1.75
Coca Cola	184	—	—	46	2	2.75
Big Mac Menu	1202.4	31.3	48.7	158.5	2	8.95
Quarter Pounder Menu	1240.4	38.7	51.7	154.9	2	8.95
Minimum Requirement	3000	65		375		
Maximum Allowance			117			

Table 10.1: Data for different food types

10.2 Model formulation

In this section the diet problem is translated into an integer program with a single symbolic range constraint.

This section

A verbal model statement of the problem is as follows.

Verbal model statement

Minimize: *the total cost of the menu,*

Subject to:

- *for all nutrients: the quantity of nutrient in the menu satisfies the minimum and maximum requirements,*
- *for all food types: an upper bound on the number of servings.*

The verbal model statement of the diet problem can be specified as a mathematical model using the following notation.

Notation

Indices:

f *food types*
 n *nutrients*

Parameters:

v_{fn}	value of nutrient n in one unit of food f
u_f	upper bound on number of servings of food f
\overline{m}_n	maximum allowance of nutrient n in the menu
\underline{m}_n	minimum requirement of nutrient n in the menu
p_f	price of one unit of food f

Variable:

x_f	number of servings of food f in menu
-------	--

The objective is to minimize the cost of the menu measured in Dutch guilders (Hfl).

*Objective
function*

$$\text{Minimize: } \sum_f p_f x_f$$

The following equation expresses the range constraints using symbolic notation. For the McDonald's problem, inspection of the last two lines of Table 10.1 shows that the amounts of calories, protein and carbohydrates must meet minimum requirements, while the amount of fat in the diet is limited by a maximum allowance.

*Nutrient
requirements*

$$\underline{m}_n \leq \sum_f v_{fn} x_f \leq \overline{m}_n \quad \forall n$$

From a syntactical point of view the above range constraint indicates the existence of both a lower and an upper bound on the amount of each nutrient in the diet. However, it is not clear what values should be used to represent the blank entries for the parameters \underline{m}_n and \overline{m}_n in Table 10.1. Should they to be interpreted as zero, infinity, or minus infinity? In the AIMMS modeling language are there several ways to specify the semantics of symbolic range constraints. By restricting the domain of definition of the above nutrient requirement constraint you can avoid the generation of particular individual instances of this range constraint. By setting the default of \underline{m}_n to minus infinity and the default of \overline{m}_n to plus infinity, all blank entries have a meaningful interpretation. The corresponding inequalities are of course non-binding, and AIMMS will not pass these redundant inequalities to the underlying solver.

*Syntax versus
semantics*

Simple upper bounds on the amount of each food type f to be consumed can be expressed in the form of symbolic inequality constraints. Such translation of simple bounds into inequalities is not generally recommended as it leads to the explicit introduction (and subsequent generation) of extra constraints which are likely to be eliminated again by the underlying solution algorithm. In AIMMS you can avoid these extra symbolic constraints by specifying upper bounds as part of the declaration of the corresponding variables. In general, you should avoid formulating simple bounds on variables as explicit symbolic constraints in your model.

*Modeling
bounds on
variables*

The following mathematical statement summarizes the model developed in this section.

Model summary

Minimize:

$$\sum_f p_f x_f$$

Subject to:

$$\begin{aligned} \underline{m}_n &\leq \sum_f v_{fn} x_f \leq \overline{m}_n & \forall n \\ x_f &\in \{0 \dots u_f\}, \text{ integer} & \forall f \end{aligned}$$

Note that the amount of food x_f is bounded from above and restricted to integer values. In the McDonald's diet problem it does not make sense to allow fractional values. In other diet applications, such as animal feed problems with bulk food items, the integer requirement can be dropped as fractional amounts of feed have a meaningful interpretation.

Integer values

10.3 Quantities and units

In this section the role and importance of measurement units is highlighted. Some special features in AIMMS, such as unit-valued parameters and unit conventions, are explained. The diet model is used for illustrative purposes. A more extensive discussion of quantities and units can be found in [Bi99].

This section

Measurement plays a central role in observations of the real world. Measurements give *quantity information* that is expressed in terms of *units*. A unit is a predefined amount of a quantity. Quantity information describes *what* is being measured, such as time, mass or length. For a given quantity it is possible to define various units for it. Take time as an example. A time period can be expressed equivalently in terms of minutes, hours or days. Table 10.2 shows some of the quantities used in the diet model.

Quantities

quantity	application in diet model
mass	to measure the amount of protein, fat and carbohydrates, and the weight of the food types
energy	to measure the amount of calories
currency	to measure the cost

Table 10.2: Quantities used in the diet model

To provide a meaningful description of a quantity, it is necessary to express its measurement in terms of a well defined unit. For each quantity it is possible to define a *base unit*. For instance, in the International System of Units, the quantity ‘length’ has the base unit of ‘meter’ (denoted by [m]). From each base unit, it is also possible to define *derived units*, which are expressed in terms of the base unit by means of a linear relationship. The base units and derived units for the diet model are provided in Table 10.3.

Units ...

quantity	base unit	derived units
mass	[kg]	[gram]
energy	[J]	[kJ], [kcal]
currency	[\$]	[Hfl]
unitless	[-]	

Table 10.3: Units used in the diet model

When all parameters are specified in terms of their units, AIMMS can check the expressions and constraints for unit consistency. For example, AIMMS will not allow two parameters, one in kilograms and the other in joules, to be added together. More generally, AIMMS will report an error when terms in an expression result in unit inconsistency. Therefore, if you specify the units for each parameter, variable, and constraint in the diet model, AIMMS will carry out this extra check on the correctness of your model.

... for consistency

Apart from unit consistency, expressing your model formulation with complete unit information will help to ensure proper communication with external data sources and other users. That is, when you want to link with other models or databases which may use different units, or, when you need to communicate with other users who may use other definitions.

... for proper communication

Quantities used in animal feed blending problems are typically expressed in ‘mass’ units such as [ton] or [kg]. Therefore, a nutrient such as calories is often measured in [kcal/ton] or [kcal/kg]. This convention is not as natural for the McDonald’s problem. Instead, nutrient values are specified per individual item (e.g. per Big Mac or per Coca Cola). Expressing the calories for a ‘Big Mac Menu’ in [kcal/gram] is not immediately useful since the weight of the Menu is generally not known. For the McDonald’s problem, [kcal/item] or [kcal/BigMac] or just [kcal] is a more meaningful expression, and one could argue the plausibility of each of these choices.

Feed units and food units

Throughout the remainder of this section, the last option of expressing quantities of food in terms of items will be used. This is a typical choice when dealing with discrete quantities that can be expressed in terms of the natural numbers. For instance, it is sufficient to state that the number of calories in a Big Mac is 479 [kcal], and that the number of Big Macs in the diet is at most 2.

Choice of food units

The relationship between base units and the derived units in Table 10.3 must be clearly defined. In the AIMMS modeling language, the following syntax is used to specify *unit conversions*.

Unit conversions

```
[gram]  -> [kg]      : # -> # / 1000
[kJ]     -> [J]       : # -> # * 1000
[kcal]   -> [J]       : # -> # * 4.1868E+03
[Hfl]    -> [$]       : # -> # * exchange_rate
```

The interpretation of these conversions (indicated with an \rightarrow) is straightforward. For instance, [gram] is converted to [kg] by considering any number # measured in [gram], and dividing this number by 1000. Note that the unit conversion from [Hfl] to [\$] associated with the quantity ‘currency’ is parameterized with an identifier ‘exchange_rate’. This identifier needs to be declared as an ordinary parameter in the model.

When specifying a model using the symbolic indexed format, it is possible that not every element of an indexed identifier is measured in the same unit. For instance, in the diet model, the parameter v_{fn} , the value of nutrient n of food f is measured in both [gram] and [cal] as shown below. In such a situation there is a need for an indexed unit-valued parameter to complement the use of symbolic indexed identifiers in your model. In the diet model, the unit-valued parameter U_n provides this key role, and it is specified in the following table.

Unit-valued parameters

Nutrient	U_n
Calories	[cal]
Protein	[gram]
Fat	[gram]
Carbohydrates	[gram]

Table 10.4: Values for the unit-valued parameter U_n

With the nutrient units specified, unit consistency can be enforced by attaching units in Table 10.5 to all (symbolic) model identifiers introduced thusfar.

In order to determine the total weight of the optimal diet, the following additional parameters are declared.

Total weight of diet

Parameters:

w_f *weight for food type f*
 W *total weight of the optimal diet*

Model Identifier	Unit
x_f	[-]
u_f	[-]
p_f	[Hfl]
\overline{m}_n	U_n
\underline{m}_n	U_n
v_{fn}	U_n

Table 10.5: Model identifiers and their associated units

where

$$W = \sum_f w_f x_f$$

Note that the variable x_f represents the number of items (a unitless quantity) and consequently, the units of W are the same as the units of w_f (namely [gram]).

Food Type	w_f [gram]	Food Type	w_f [gram]
Big Mac	200	Lowfat Milk	250
Quarter Pounder	200	Coca Cola	400
Vegetable Burger	133	Big Mac Menu	730
French Fries	130	Quarter Pounder Menu	730
Salad	127		

Table 10.6: Weight for each food type

The optimal diet satisfying the nutrient requirements for the afternoon and the evening meal costs 24.60 [Hfl] and contains one ‘Vegetable Burger’, one ‘Coca Cola’ and two ‘Quarter Pounder Menus’ (which include drinks). This diet contains 3006 [kcal] of energy, 89 [gram] of protein, 114 [gram] of fat, and 406 [gram] of carbohydrates. The formulation in this chapter has not included any requirements reflecting taste or minimum quantities of foods to be consumed. By adding such requirements, you can study the increase in cost and the effect on nutrient consumption.

Solution

There are model-based applications which may be used by end-users from around the world. In that case, it is important that all users can work with their particular *unit convention*, and view the model data in the units associated with that convention. In the AIMMS modeling language it is possible to define one or more unit conventions, and all data transfer from and to an external medium is interpreted according to the units that are specified in the convention. By switching between unit conventions, different end-users can use their own choice of units.

Unit conventions

To illustrate the use of conventions, consider the following two convention definitions. The ‘DutchUnits’ convention specified by attaching the unit [Hfl] to the ‘Currency’ quantity and the unit [kJ] to the ‘Energy’ quantity. The ‘AmericanUnits’ convention specified by attaching the unit [\$] to the ‘Currency’ quantity and the unit [kcal] to the ‘Energy’ quantity. When the ‘AmericanUnits’ convention is selected by a user, and given the exchange rate of 0.50, AIMMS will report the optimal cost as 12.3 [\$]. In the ‘DutchUnits’ convention, the total amount of energy in the optimal diet will be reported as 12,586 [kJ].

*Conventions
applied*

10.4 Summary

In this chapter a simplified diet problem was introduced together with a small data set. The problem was transformed into an integer programming model with symbolic range constraints. The role of measurement units to obtain data consistency and proper data communication was highlighted. Special reference was made to the use of the unit-valued parameters and unit conventions available in AIMMS.

Exercises

- 10.1 Implement the mathematical program summarized at the end of Section 10.2 using the example data provided in Section 10.1. Verify that the solution coincides with the one presented in Section 10.3.
- 10.2 Introduce quantities, units and a unit parameter into your AIMMS model as suggested in Section 10.3, and include them in the graphical display of your model results.
- 10.3 Introduce the two unit conventions into your AIMMS model as suggested at the end of Section 10.3. Design a page in AIMMS, so that the user can select the convention of his choice and the input-output data adjusts itself accordingly.

Chapter 11

A Farm Planning Problem

In this chapter you will find a worked example of a simplified farm planning problem in a developing country. Every year a farmer must decide what crops to grow, thereby taking into account such limiting resources as land, labor and water. His objective is to maximize farm revenue. This simplified farm planning problem can be translated into a linear optimization model. Such a model is typically found as a submodel in larger models which focus on agricultural questions concerning an entire region. Even though the model developed here is relatively small in terms of symbolic constraints, the number of identifiers is relatively large.

This chapter

The material of this chapter has been adapted from “Modeling for Agricultural Policy and Project Analysis” by G.P. Kutcher, A. Meeraus, and G.T. O’Mara [Ku88]. A text book on agricultural modeling is [Ha86].

References

Linear Program, Measurement Units, Sensitivity Analysis, What-If Analysis, Worked Example.

Keywords

11.1 Problem description

The three main inputs to agricultural production are *land*, *labor* and *water*. Their requirements vary throughout the year for different crop activities. In this chapter all requirements will be specified on a monthly basis, and are assumed to be known with certainty.

Three main inputs

Crop rotation is the practice of growing different crops, in sequence, on the same land. The specification of all relevant crop rotations and time shifts in real-world applications is a non-trivial task, and requires the aid of an agronomist. In this chapter, a monthly calendar is maintained. The periods in which crops can be grown, are given in Table 11.1. An entry in this table denotes the fraction of a month that land will be occupied by a particular crop. The total amount of land available is assumed to be 10 hectares.

Land

	wheat	beans	onions	cotton	maize	tomatoes
Jan	1	1	1			
Feb	1	1	1			
Mar	1	1	1	.5		
Apr	1	1	1	1		
May	1		.25	1	.25	
Jun				1	1	
Jul				1	1	.75
Aug				1	1	1
Sep				1	1	1
Oct				1	.5	1
Nov	.5	.25	.5	.75		.75
Dec	1	1	1			

Table 11.1: Land occupation [-]

Labor is one of the major requirements for crop activities, and is supplied by the farmer's family as well as outside workers. The use of labor is disaggregated over the year, because labor is employed for different activities (land preparation, planting, maintenance and harvesting) requiring different intensities. For each crop pattern there will be a set of labor requirements that vary each month. Table 11.2 gives the labor requirements in the same pattern as land use seen in Table 11.1. Each number in the table represents the amount of labor [hr] required that month for growing one hectare of the corresponding crop.

*Labor
requirements*

	wheat	beans	onions	cotton	maize	tomatoes	hours
Jan	14	6	41				160
Feb	4	6	40				160
Mar	8	6	40	40			184
Apr	8	128	155	40			176
May	137		19	72	34		168
Jun				16	40		176
Jul				12	57	136	176
Aug				16	64	120	176
Sep				8	35	96	176
Oct				46	9	56	168
Nov	19	60	89	34		48	176
Dec	11	6	37				176

Table 11.2: Crop labor requirements [hr/ha] and monthly hours [hr/man]

In addition to the already available family labor, outside workers can be hired as permanent workers or temporary workers. Permanent family labor consists of 1.5 person, and costs \$ 4,144 per man for an entire year. Permanent outside

Workers

labor costs \$ 5,180 per man for a year, while temporary labor costs is \$ 4 per hour. The number of working hours differs per month and is listed in the last column of Table 11.2. Note that the fractional value of 1.5 for family labor can be viewed as an average over several similar farms or as an indication of parttime labor activity.

Water is another major requirement for agricultural production, and can be supplied from surface water distribution or from groundwater. In this chapter it is assumed that the total amount of water available to the farmer each month is restricted to 5 kcub. Furthermore, there is an overall annual limit on the use of water equal to 50 kcub. The price of water is fixed for the entire year, and amounts to \$ 10 per kcub. The crop water requirements for the farm are given in Table 11.3.

Water

	wheat	beans	onions	cotton	maize	tomatoes
Jan	0.535	0.438	0.452			
Feb	0.802	0.479	0.507			
Mar	0.556	0.505	0.640	0.197		
Apr	0.059	0.142	0.453	0.494		
May				1.047	0.303	
Jun				1.064	0.896	
Jul				1.236	1.318	0.120
Aug				0.722	0.953	0.241
Sep				0.089	0.205	0.525
Oct						0.881
Nov	0.373	0.272				0.865
Dec	0.456	0.335	0.305			

Table 11.3: Crop water requirements [kcub/ha]

Most farms in developing countries place an emphasis on the production of their own food, since it makes them self-sufficient. Rather than fixing certain crops for family consumption, a slightly more general treatment is to allow a choice from a set of a priori constructed consumption bundles. Each such bundle is composed of different crops, and is by itself sufficient to feed the family. A menu of balanced meals can be derived from it. By considering a combination of alternative bundles, a farmer can take his overall revenue into account while deciding on his own consumption.

*Family
consumption*

Consider three alternative consumption bundles (in tons per year) that the farmer can choose from. The first bundle contains 1.20 tons of beans, 0.15 tons of maize, and 0.25 tons of tomatoes. The second bundle contains 0.73 tons of beans, 1.50 tons of maize, and 0.25 tons of tomatoes. The third bundle contains 0.70 tons of beans, 1.00 ton of maize, and 0.75 tons of tomatoes. It is

*Consumption
data*

assumed that any combination of these three bundles may be selected by the farmer.

Based on previous years the expected yield for each crop can be estimated in advance. For instance, the *yield* of growing cotton on one hectare of land will be 1.5 tons of cotton. Similar figures exist for the yield of the other crops. Furthermore, the *price* of crops determines the farm revenue, and price indications are also assumed to be known. In table 11.4 the relevant yields and prices are presented.

*Yields and
revenues*

	yield [ton/ha]	price [\$/ton]
wheat	1.50	1000
beans	1.00	2000
onions	6.00	750
cotton	1.50	3500
maize	1.75	700
tomatoes	6.00	800

Table 11.4: Crop yields and prices

11.2 Model formulation

In this section the above description is translated into an optimization model. First, an informal verbal presentation of the model is provided, followed by the extensive notation needed to describe all aspects of the model. The objective function and each constraint is developed separately. A model summary is listed at the end.

This section

By considering the basic choices of the farmer and his limited resources, it is fairly straightforward to describe his objective and constraints in a compact verbal manner.

*Verbal model
statement*

Maximize: *total net farm revenue which is the revenue from sales minus the costs associated with labor and water.*

Subject to:

- *for all months: the land used for cropping activities must be less than or equal to the land available,*
- *for all months: the labor needed for cropping activities must be less than or equal to available family labor plus hired permanent labor plus hired temporary labor,*
- *for all months: water needed for cropping activities must be less than or equal to the monthly water limit,*

- the annual amount of water needed must be less than or equal to the annual water limit, and
- for all crops: the amount produced must be equal to the amount to be consumed plus the amount to be sold.

The following notation is based as much as possible on the use of a single letter for each identifier for reasons of compactness. Such compact notation is not recommended for practical models built with a system such as AIMMS, because short names do not contribute to the readability and maintainability of computerized models.

Notation

Indices:

c	<i>crops</i>
t	<i>months</i>
b	<i>consumption bundles</i>

Parameters (crop related):

y_c	<i>yield of crop c [ton/ha]</i>
p_c	<i>price of crop c [\$/ton]</i>
d_{cb}	<i>amount of crop c in bundle b [ton]</i>

Parameters (land related):

L	<i>land available [ha]</i>
l_{tc}	<i>fraction of month t that crop c occupies land [-]</i>

Parameters (labor related):

v_{tc}	<i>labor required of crop c during month t [hr/ha]</i>
r^F	<i>annual wage rate family labor [\$/man]</i>
r^P	<i>annual wage rate permanent labor [\$/man]</i>
r^T	<i>hourly wage rate temporary labor [\$/hr]</i>
h_t	<i>working hours in month t [hr/man]</i>
V^F	<i>family labor available [man]</i>

Parameters (water related):

W	<i>annual amount of water available [kcub]</i>
w_t	<i>limit on use of water in month t [kcub]</i>
R_{tc}	<i>water requirement for crop c in month t [kcub/ha]</i>
p^W	<i>price of water [\$/kcub]</i>

Variables:

x_c	<i>amount of crop c planted [ha]</i>
V^P	<i>permanent labor hired [man]</i>
V_t^T	<i>temporary labor hired in t [hr]</i>
s_c	<i>sales of crop c [ton]</i>
z_b	<i>fraction of bundle b consumed [-]</i>

Land use for crops is described in Table 11.1, in which an entry denotes the fraction of a month that land will be occupied by a particular crop. It may take you a while to get used to the definition of this table, but it represents a compact manner to describe both the timing and the use of land for the production of crops. A fraction of 1 means that land is to be occupied during the entire month for the corresponding crop. A fraction of less than 1 indicates that land is used either during the last portion of a month when the crop is sown, or during the first portion of a month when the crop is harvested and the land is cleared for another crop. With the use of Table 11.1, the resulting land limitation constraint assumes a simple format.

Land limitation

$$\begin{aligned} \sum_c l_{tc} x_c &\leq L \quad \forall t \quad [\text{ha}] \\ x_c &\geq 0 \end{aligned}$$

Labor requirements for growing a particular crop on one hectare of land can be found in Table 11.2. In addition to the available family labor, both permanent and/or temporary labor may need to be hired. Note that permanent and temporary labor is not expressed in the same unit. That is why the conversion factor h_t [hr/man] is used to express the following constraint on labor in terms of hours.

Labor requirements

$$\begin{aligned} \sum_c v_{tc} x_c &\leq h_t(V^F + V^P) + V_t^T \quad \forall t \quad [\text{hr}] \\ V^P &\geq 0, \quad V_t^T \geq 0 \quad \forall t \end{aligned}$$

There is a monthly and an annual restriction on the use of water. The mathematical form of both constraints is similar to the constraints on land and labor discussed previously.

Water requirements

$$\begin{aligned} \sum_c R_{tc} x_c &\leq w_t \quad \forall t \quad [\text{kub}] \\ \sum_{tc} R_{tc} x_c &\leq W \quad [\text{kub}] \end{aligned}$$

The amount of each crop produced during the year is meant to be sold, with the exception of those amounts that are to be used for family consumption. As indicated before, a combination of consumption bundles is to be selected to satisfy the family needs. Such a combination can be formulated as a convex combination, and is implemented by attaching a nonnegative weight to each bundle, and letting the sum of the weights be equal to one. This approach makes sure that there is enough food for the family, but allows for variation

Family consumption

in the weights to obtain maximal farm revenue from sales.

$$\begin{aligned} y_c x_c &= \sum_b d_{cb} z_b + s_c \quad \forall c \quad [\text{ton}] \\ \sum_b z_b &= 1 \\ s_c &\geq 0 \quad \forall c \\ z_b &\geq 0 \quad \forall b \end{aligned}$$

The objective is to maximize farm profit over the year, which is equal to revenue from sales minus the costs associated with labor and water. Note that in this simplified model, the cost of seedlings is ignored.

*Objective
function*

$$\sum_c p_c s_c - r^F V^F - r^P V^P - r^T \sum_t V_t^T - p^W \sum_{tc} R_{tc} x_c$$

The mathematical description of the model can now be summarized as follows.

*Mathematical
model summary*

Maximize:

$$\sum_c p_c s_c - r^F V^F - r^P V^P - r^T \sum_t V_t^T - p^W \sum_{tc} R_{tc} x_c$$

Subject to:

$$\begin{aligned} \sum_c l_{tc} x_c &\leq L & \forall t \\ \sum_c v_{tc} x_c &\leq h_t (V^F + V^P) + V_t^T & \forall t \\ \sum_c R_{tc} x_c &\leq w_t & \forall t \\ \sum_{tc} R_{tc} x_c &\leq W \\ y_c x_c &= \sum_b d_{cb} z_b + s_c & \forall c \\ \sum_b z_b &= 1 \\ x_c &\geq 0 & \forall c \\ V^P &\geq 0 \\ V_t^T &\geq 0 & \forall t \\ s_c &\geq 0 & \forall c \\ z_b &\geq 0 & \forall b \end{aligned}$$

11.3 Model results

In this section you will find a summary of the optimal solution that you yourself should observe after having implemented the model in AIMMS. In addition, there are some comments regarding experiments that you could carry out to investigate the effect of parameter changes on the value of the optimal solution.

This section

The optimal cropping pattern and the use of crop returns are summarized in Table 11.5. As expected, the crops that are not sold, are used for family consumption. The optimal combination of bundles was found to be bundle 1 (7%) and bundle 3 (93%). Note that beans and maize are only grown to meet family consumption. The optimal yearly net revenue turns out to be \$ 49,950.

Optimal solution

	crop x_c [ha]	yield $y_c x_c$ [ton]	sales s_c [ton]	consumption $\sum_b d_{cb} z_b$ [ton]
wheat				
beans	0.74	0.74		0.74
onions	6.33	37.96	37.96	
cotton	2.94	4.40	4.40	
maize	0.54	0.94		0.94
tomatoes	5.55	33.29	32.58	0.71

Table 11.5: Optimal crop-related solution

Besides the fixed amount of permanent family labor of 1.5 man, the optimal solution also indicates a need for permanent hired labor of 0.54 man. As stated before, these fractional numbers could be interpreted as an average over several similar farms, or as indication of permanent parttime work. The optimal amount of temporary labor hired is expressed in Table 11.6.

Use of labor

	V_t^T [hr]		V_t^T [hr]
Jan		Jul	461.72
Feb		Aug	388.43
Mar		Sep	216.25
Apr	833.86	Oct	108.24
May	7.45	Nov	614.86
Jun		Dec	

Table 11.6: Optimal temporary labor hired

In AIMMS you can turn on the property `ShadowPrice` for each individual symbolic constraint. If you do so for the labor requirement constraint, you will observe that the shadow prices for January, February, June and December are all zero. This indicates an overcapacity of permanent labor during these months. In March the amount of available permanent labor is exactly enough to meet labor requirements. In all other months, temporary labor is hired to meet labor requirements.

Shadow price of labor

The water requirement constraints are only binding during the months March, July and November. The corresponding shadow prices for these constraints are nonzero and reflect the binding use of water. The annual use of water amounts to 47.32 kcub, which is less than the total annual limit.

Use of water

If the annual supply of water is not entirely known a priori, the farmer may want to look how the optimal solution changes as a result of different values. You can implement such an experiment in AIMMS by creating a parametric curve with annual water supply along the x -axis and crop assignments along the y -axis. Figure 11.1 represents such a curve with the annual water limit between 40 and 50 kcub. The observed sensitivity is not so strong, so that the farmer can make a decision even if the annual water limit varies within the above range.

Changing the water limit

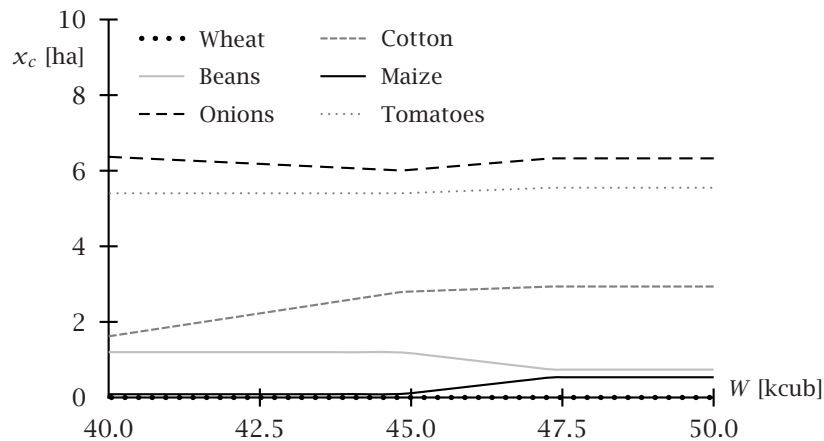


Figure 11.1: Optimal crops as function of annual water limit

A similar experiment can be made when the farmer has the opportunity to use additional land. The parametric curve in Figure 11.2 shows the sensitivity of optimal cropping patterns with respect to land. As more land becomes available, the amount of cotton increases while the amount of onions and maize decreases. You may verify for yourself that total net revenue increases by more than 20 % as the amount of available land doubles.

Changing the land availability

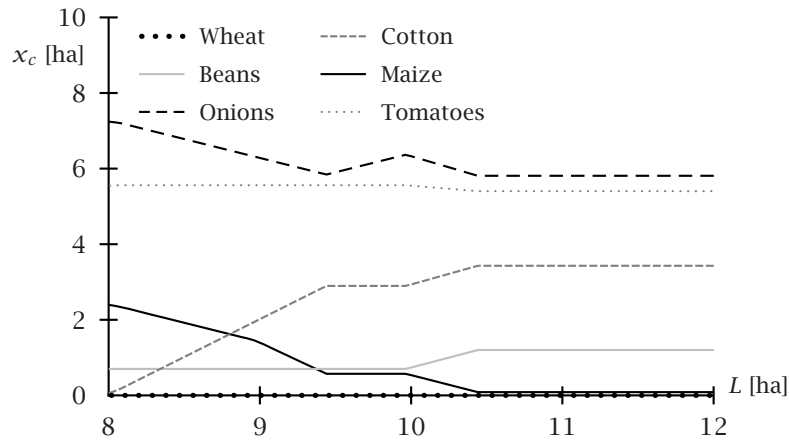


Figure 11.2: Optimal crops as function of available land

11.4 Summary

In this chapter a simplified farming problem was introduced together with a small data set for computational purposes. The corresponding model turned out to be a straightforward linear program with constraints on land, labor, water and family consumption. What-if experiments were performed to determine the sensitivity of selected parameters on the optimal solution.

Exercises

- 11.1 Implement the mathematical program described at the end of Section 11.2 using the example data provided in Section 11.1. Verify that the optimal solution produced with AIMMS is the same as the solution provided in Tables 11.5 and 11.6.
- 11.2 Add all quantity and unit information to your model in AIMMS, and check whether the units are consistent throughout the model.
- 11.3 Use the parametric curve object in AIMMS to reproduce the sensitivity experiments described in Section 11.3.

Chapter 12

A Pooling Problem

In this chapter you will encounter a simplified example of a refinery pooling problem. Intermediate product streams, each with their own particular properties, are fed into a limited number of available tanks. These tanks are referred to as pool tanks. Through pooling, the input products no longer exist in their original form, but are mixed to form new pooled products with new property values. These new products are subsequently used to blend final products. These final blends must satisfy specific quality requirements, which means that their property values must be between a priori specified ranges. The pooling problem can be translated into a nonlinear programming model. This model has the nasty property that there are likely to exist multiple locally optimal solutions. Good starting values are then required to steer the algorithm away from poor local optima. An instance of the pooling problem is provided for illustrative purposes.

This chapter

Pooling problems have received some attention in the informal literature of the sixties and seventies. A typical reference is [Ha78] which describes a heuristic approach based on recursive programming techniques. In later years, global optimization techniques have been proposed by [Fl89], but these have found limited application in large-scale practical applications.

References

Nonlinear Program, Multiple Optima, Worked Example.

Keywords

12.1 Problem description

In this section a simplified version of the pooling problem is discussed and illustrated. The paragraphs aim to give you a basic feel for the problem at hand. Despite its simplifications, the problem is still of interest, because it captures the difficulties associated with pooling problems in general.

This section

In a refinery, crude oils of different types are first *distilled* in one or more crude distillers. This process results in the production of several intermediate product streams that are immediately *pooled* in dedicated pool tanks. Any further processing consists of *blending* pooled products into final products. This three-step process is illustrated in Figure 12.1. In this chapter the time-

*Refinery
operations
summarized*

phasing of product flows is completely ignored in order to prevent the problem and the resulting model from becoming too complicated.

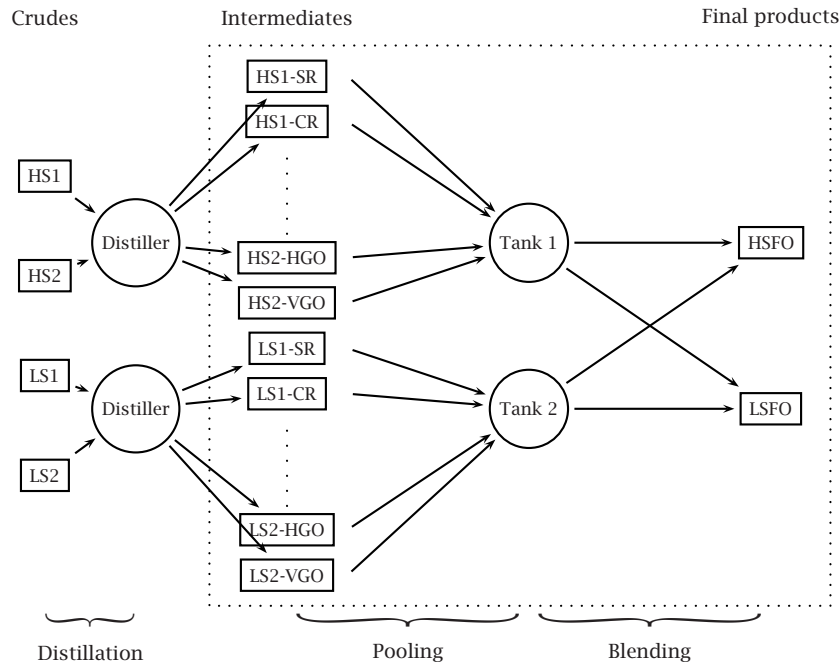


Figure 12.1: A simplified refinery

Crude oils are by no means identical. Their composition strongly depends on their location of origin. In Figure 12.1 there are four crudes: two of them are high-sulphur (HS) crudes and two of them low-sulphur (LS) crudes. Their sulphur content is referred to as a *product property*. Product properties are to a certain extent retained during the distillation phase. That is why the labels HS and LS are also attached to the intermediate product streams for each originating crude 1 and 2.

Product properties ...

Product properties cannot be measured in a uniform manner. There are properties, such as sulphur content, that are naturally expressed as a percentage of total volume or total mass. Other properties, such as viscosity and pour point, are not naturally measured in these terms. For instance, pour point is the lowest temperature, expressed as a multiple of 3 degrees Celsius, at which oil is observed to flow when cooled and examined under prescribed conditions. Special care is then required to compute such a property for a mixture of products.

... and their measurement

When two or more intermediate products are pooled in a single tank, a new product will result. The properties of this new product will be related to the properties of the originating products, but there will always be some form of dilution of each property. When there are multiple pool tanks, it is desirable to minimize the dilution effect across pool tanks. The resulting variability in pool properties is needed to meet the property requirements of the final products. For instance, in Figure 12.1 all high-sulphur intermediates are not pooled with any low-sulphur intermediates in order to maintain sufficient variability in sulphur property values across the two pool tanks.

*Mixing
properties
causes dilution*

In Figure 12.1, each of the two crude distillers produces four intermediate products, namely, a short residue (SR), a cracked residue (CR), a heavy gas oil (HGO) and a visbroken gas oil (VGO). In order to track the properties of the originating four crude oils, the name of each intermediate is prefixed with the name of the crude. In this case, such a naming convention results in sixteen different product names. Only one final product, namely fuel oil (FO), is produced in two qualities, resulting in two final product names. You can imagine how the number of possible product names can explode in large real-world applications where there are more products and several additional properties. The number of product names becomes even larger when the entire refinery process is considered over several discrete time periods, and properties are tracked over time.

*Intermediate
and final
products*

Theoretically, it is attractive to store all intermediate products in their own intermediate product tanks. This delays any dilution of product properties until final products have to be blended to specification. In practice, however, the unique intermediate product streams outnumber the available tanks, and product pooling is required. For the sake of keeping the problem fairly simple, it is assumed that the flow of intermediate products into the pool tanks equals the flow required to blend the final products, and that each pool tank has limited capacity.

Pooled products

In the example of this chapter it is assumed that the volume of each intermediate product to be used must stay within specified bounds. This is a slight simplification, because in actuality, intermediate products are produced in fixed relative proportions, and their absolute volume is related to the volume of the emanating crude. However, the distillation phase is not part of the problem in this chapter, which justifies the assumption of fixed bounds on inputs.

*Limitation on
intermediates*

The price of a final product is not entirely constant, but is assumed to be dependent on its associated property values. This implies that price becomes an unknown in the pooling model to be build. The objective function is to maximize the total sales value of final products to be made.

*Maximizing
sales value*

The pooling problem considered in this chapter is to maximize the sales value of end products by deciding how much of each intermediate stream, within limits, is to be placed in each of the pool tanks. The extra restrictions are that the new pool mixtures are sufficient to produce the required amount of final products, and that the properties of the new mixtures are sufficient to satisfy final product requirements.

Pooling problem summarized

12.2 Model description

In this section the basic rules for blending on volume and blending on weight are developed before stating the mathematical formulation of the underlying pooling problem.

This section

For the sake of simplicity, consider two intermediate products (1 and 2) to be mixed into a new pooled product (3). Let the symbol x denote the amount of product, and let the symbol p denote a particular property. The following two equalities express proportional blending.

Proportional blending ...

$$\begin{aligned}x_3 &= x_1 + x_2 \\p_3 x_3 &= p_1 x_1 + p_2 x_2\end{aligned}$$

The first identity is the product balance equation, which is linear. The second identity is the property determination equation, and is nonlinear when both x_3 and p_3 are considered to be unknown.

Proportional blending is properly defined when the units of measurement are consistent. Consistency is obtained, for instance, when product amounts are measured in terms of mass, and the product property is measured as a percentage of mass. Similarly, consistency is also obtained when product amounts are measured in terms of volume, and the product property is measured as a percentage of volume. If required, it is always possible to transform volume into mass or vice versa using product densities.

... requires consistent measurements

As has been mentioned in the previous section, there are several product properties, such as viscosity and pour point, that are not measured as a percentage of mass or volume. These properties play a vital role in determining the quality of a blend. In practice, a nonlinear function of such properties is constructed such that the resulting transformed property values can be viewed as a percentage of either volume or mass. The determination of such specialized nonlinear functions is based on laboratory experiments and curve fitting techniques.

If this is not the case ...

Let $f(p)$ denote a nonlinear function of one of the properties discussed in the previous paragraph. Assume that x is expressed in terms of mass, and that $f(p)$ is measured as a percentage of mass. Then the following identities express proportional blending.

... then use transformed measurements

$$\begin{aligned}x_3 &= x_1 + x_2 \\ f(p_3)x_3 &= f(p_1)x_1 + f(p_2)x_2\end{aligned}$$

You could of course use a variable for $f(p)$ and apply the inverse of f to obtain p after you have found the solution of the underlying model. This is what is typically done in practice.

By considering the basic pooling problem described in this chapter, it is fairly straightforward to describe the objective and constraints in a compact verbal manner.

Verbal model statement

Maximize: *total sales value of final products*

Subject to:

- *for all pool tanks: the bounded flow entering a pool tank must be equal to the flow leaving a pool tank,*
- *for all properties and pool tanks: the property values of pooled product are determined by the property values of the products entering the pool tank,*
- *for all properties and final products: the property values of final product are determined by the property values of the products coming from the pool tanks,*
- *for all final products: the quantities of final product must be between specified bounds,*
- *for all properties and final products: the property values of final product must be between specified bounds,*

The following notation is based as much as possible on the use of a single letter for each identifier for reasons of compactness. Such compact notation is not recommended for practical models built with a system such as AIMMS, because short names do not contribute to the readability and maintainability of computerized models.

Notation

Indices:

p	<i>properties</i>
i	<i>intermediates</i>
t	<i>pool tanks</i>
f	<i>final products</i>

Parameters:

v_{pi}	<i>value of property p in intermediate i</i>
\underline{r}_i	<i>minimal amount of intermediate i to be used</i>
\overline{r}_i	<i>maximal amount of intermediate i to be used</i>

\underline{r}_f	minimal required amount of final product f
\bar{r}_f	maximal required amount of final product f
\underline{w}_{pf}	minimal value of property p in final product f
\bar{w}_{pf}	maximal value of property p in final product f
c_t	capacity of pool tank t

Variables:

v_{pt}	value of property p in pool tank t
v_{pf}	value of property p in final product f
x_{it}	flow of intermediate i to pool tank t
x_{tf}	flow of pool tank t to final product f
s_t	total stock of pooled product in pool tank t
π_f	sales price of final product f

As discussed in the previous section, the amount of each intermediate product to be pooled is restricted from above and below. Instead of writing a single flow balance constraint for each pool tank, there are separate equations for both inflow and outflow using the same stock variable. It is then straightforward to specify a simple bound on the stock level in each pool tank. The following five constraints capture these limitations on flow from and to the pool tanks.

Flow constraints

$$\begin{aligned}
 \sum_t x_{it} &\geq \underline{r}_i \quad \forall i \\
 \sum_t x_{it} &\leq \bar{r}_i \quad \forall i \\
 s_t &= \sum_i x_{it} \quad \forall t \\
 s_t &= \sum_f x_{tf} \quad \forall t \\
 s_t &\leq c_t \quad \forall t
 \end{aligned}$$

The property value determination constraints are essentially the proportional blending equalities explained at the beginning of this section. These constraints are only specified for pooled and final products, because the property values of all intermediate products are assumed to be known.

Property value determination constraints

$$\begin{aligned}
 v_{pt} \sum_i x_{it} &= \sum_i v_{pi} x_{it} \quad \forall (p, t) \\
 v_{pf} \sum_t x_{tf} &= \sum_t v_{pt} x_{tf} \quad \forall (p, f)
 \end{aligned}$$

Due to market requirements with respect to quantity and quality, both the amount of final product and the associated property values must be between a priori specified bounds.

*Final product
requirement
constraints*

$$\begin{aligned}\sum_t x_{tf} &\geq \underline{r}_f \quad \forall f \\ \sum_t x_{tf} &\leq \bar{r}_f \quad \forall f \\ v_{pf} &\geq \underline{w}_{pf} \quad \forall (p, f) \\ v_{pf} &\leq \bar{w}_{pf} \quad \forall (p, f)\end{aligned}$$

As has been indicated in the previous section, the price of a final product is not entirely constant, but is assumed to be dependent on its associated property values. In the specification below, only an abstract functional reference F to property dependence is indicated. In the worked example of the next section a particular function is used for numerical computations. The objective function to be maximized can then be written as follows.

*Objective
function*

$$\begin{aligned}\sum_f \pi_f \sum_t x_{tf} \\ \pi_f = F(v_{pf})\end{aligned}$$

12.3 A worked example

In this section you will find a description of model input data that is consistent with the entities in Figure 12.1. In addition, you will find some comments based on running a few computational experiments with AIMMS.

This section

The variable x_{it} denotes the flow of intermediate i to pool tank t . In Figure 12.1, these intermediate flows are restricted such that all high and low sulphur products are pooled into separate pool tanks. In AIMMS, you can model this by simply specifying an index domain as part of the declaration of the variable x . Such a domain is then a parameter with nonzero entries for the allowed combinations of i and t .

*Domain
restrictions on x*

The symbol v is used both as a parameter and a variable depending on the index references. In AIMMS, you can implement this dual status by declaring the symbol to be a variable, and then changing its status to non-variable for selected index combinations. You can accomplish this change by writing an assignment statement inside a procedure using the NonVar suffix. A typical assignment is

*Variable status
of v*

```
v(p,i).NonVar := 1;
```

Both the lower and upper bound on the amount in [kton] of each intermediate product to be pooled are displayed in Table 12.1. In this table you also find the property values associated with the properties sulphur and V50, both measured as a percentage of mass. The property V50 is a derived measure of viscosity for which the proportional blending rule is appropriate.

*Intermediate
product data*

	\underline{r}_i	\overline{r}_i	v_{pi}	
	[kton]	[kton]	Sulphur [%]	V50 [%]
HS1-SR	1	3	5.84	43.7
HS1-CR		3	5.40	36.8
HS1-HGO		3	0.24	12.8
HS1-VGO		3	2.01	15.4
HS2-SR	1	3	5.85	47.3
HS2-CR		3	5.38	39.2
HS2-HGO		3	0.26	13.1
HS2-VGO		3	2.04	15.9
LS1-SR	1	3	0.64	39.9
LS1-CR		3	0.57	38.2
LS1-HGO		3	0.02	13.5
LS1-VGO		3	0.14	16.3
LS2-SR	1	3	0.93	38.1
LS2-CR		3	0.85	34.1
LS2-HGO		3	0.03	13.2
LS2-VGO		3	0.26	15.5

Table 12.1: Intermediate product data

The only data solely linked to pool tanks is their capacity. In this simplified example it is assumed that the capacity of each pool tank is 15 [kton].

Pool tank data

In Table 12.1 you will find the upper and lower bounds on both the quantities and property values of the final products to be blended.

*Final product
requirements*

	\underline{r}_f	\overline{r}_f	\underline{w}_{pf} \overline{w}_{pf}		\underline{w}_{pf} \overline{w}_{pf}	
	Min [kton]	Max [kton]	Sulphur		V50	
			Min [%]	Max [%]	Min [%]	Max [%]
LSFO	10	11		1.5	30.0	34.0
HSFO	11	17		3.5	32.0	40.0

Table 12.2: Final product requirements

In this particular example, the unit sales price of each final product is made dependent on its sulphur content, with low sulphur levels worth more than high sulphur levels. The base price of one [ton] of low sulphur fuel oil with the highest level of permitted sulphur is equal to 150 dollars. Similarly, the base price of one [ton] of high sulphur fuel oil with the highest level of permitted sulphur is equal to 100 dollars. These base prices can only increase as the relative level of sulphur decreases. The following formula for π_f in terms of the base price π_f^I is used.

Price of final product

$$\pi_f = \pi_f^I \left(2 - \frac{v_{\text{Sulphur},f}}{w_{\text{Sulphur},f}} \right)$$

Once you have implemented the model of the previous section in AIMMS, you are likely to obtain an error indicating that "all Jacobian elements in the row are very small". This message comes directly from the solver, and is most likely a reflection of some initial variable values at their default value of zero. The algorithm uses a matrix with derivative values for all constraints in terms of all variables. Several of these derivative values correspond with the product of two variables. From basic calculus you know that the term xy has zero partial derivatives with respect to both x and y when these are at their default value of zero.

Initial model run

The remedy to fix the problem mentioned in the previous paragraph is straightforward. By initializing the flow variables x away from their lower bound of zero, the error message will disappear. In this example you could consider random flow values for the intermediate products between their bounds, and distribute the corresponding content of the pool tanks equally over the final products. As a result, the flow balance constraint are already satisfied. As it turns out, specifying initial values for variables in nonlinear mathematical programs is just one of the ways to make the solution process more robust. Setting bounds on variables, and scaling your data such that solution values become of similar magnitude, are all useful ways to improve the likelihood that a solver will find a correct solution.

Initialize flow values

As you start to experiment with initial values for the x variables, you might find that the solver still has difficulties finding a feasible solution in some of the cases that you try. As it turns out, you can reduce the number of times the solver is unable to find a feasible solution by also computing the initial values of the v variables using both the values of x and the property value determination equalities.

Derive initial property values

If you have not found different optimal solution values after experimenting with various initial values for x and v , you may want to write an experiment in which you let the system generate random initial values for the x variables and compute the corresponding values of v . It is easy to write such a procedure in AIMMS, and make a page to display the various objective function values in a table. In this example, two distinct local optimal objective function values were found. They are 3624.0 and 4714.4.

*Multiple
solutions exist*

12.4 Summary

In this chapter a simplified pooling problem was introduced together with a small data set for computational experiments. The problem was transformed into a nonlinear programming model with proportional blending constraints to determine new product property values. Initial values of the solution variables were needed not only to support the solution finding process, but also to determine different locally optimal solutions.

Exercises

- 12.1 Implement the mathematical program described in Section 12.2 using the example data provided in Section 12.3.
- 12.2 Experiment with several initial values by writing a procedure in AIMMS as described in Section 12.3. Try to find at least two different local optima.
- 12.3 Investigate the effect of removing all pool tanks on the objective function value. Without pool tanks, final products are blended on the basis of intermediate product streams only.

Part IV

Intermediate Optimization Modeling Applications

Chapter 13

A Performance Assessment Problem

In this chapter, you will encounter the problem of determining the performance of a set of comparable organizations. Such evaluation problems are nontrivial. The basic concept of relative efficiency of one organizational unit in relation to the other units is introduced. Based on this concept, the underlying problem can be translated into a collection of linear programming models using relative efficiency as an optimization criterion. Efficient organizations can then be identified, and form a reference for the other organizations. An example with seven input-output categories and 30 related organizations is provided for illustrative purposes.

This chapter

The term Data Envelopment Analysis (DEA) is the general heading under which many papers on the assessment of comparable organizations have been written. The term was introduced [Ch78]. Since that time, several books on the topic have been written. One such reference is [No91]. Unfortunately, neither the term Data Envelopment Analysis nor its abbreviation DEA creates an immediate mental picture when considering performance assessment of comparable organizations. For that reason, the term is not used any further in this chapter.

References

Linear Program, Mathematical Reformulation, What-If Analysis, Worked Example.

Keywords

13.1 Introduction and terminology

In large distributed organizations, senior management routinely wants to evaluate the relative performance of similar decision making units (DMU's) under their control. One example of such a distributed organization in the private sector is a bank with branch offices operating as autonomous DMU's. Another example is a retail chain with similar outlets as DMU's. In the public sector you may think of a Board of Education overseeing many schools, a health program governing several hospitals, or a state prison system managing their prisons.

*Decision making
units (DMU's)*

Senior management has specific objectives in mind when evaluating the organization's DMU's. Typical issues of concern in the private sector are increasing sales, reducing costs, identifying good performers, etc. Typical issues in the public sector are improving service levels, staff utilization and the management of large expenditures. Through performance evaluations, senior management gains insight into the operation of the individual DMU's under its control. For the case where the overall organization has to shrink in terms of the number of DMU's, these evaluations can be used as a basis for eliminating the truly poor performers.

Management issues

When measuring the performance of its DMU's, management should not limit the analysis to a few isolated measures such as profit or cost. Instead, a wide range of input and output factors should be considered in order to get a comprehensive insight into how well an organization is really performing in comparison to others.

Outputs and inputs

For every type of application there are both specific and generic performance measures. Some of them are easy to measure, while others may be difficult to capture in quantitative terms.

Performance measures ...

Performance measures encountered in private sector applications are often financial in nature. Typical examples are total revenue, revenue growth, uncontrollable cost, controllable costs, total liabilities, net capital employed, etc. Examples of non-financial performance measures are competition, age of unit, catchment population, customer service, pitch, etc.

... in the private sector

Examples of performance measures in the public sector are staff utilization, productivity, throughput, accuracy, customer satisfaction, number of publications, client/staff ratio's, etc.

... and in the public sector

Efficiency can be described as the use made of resources (inputs) in the attainment of outputs. A DMU is 100% absolute efficient if none of its outputs can be increased without either increasing other input(s) or decreasing other output(s). A 100% relative efficiency is attained by a particular DMU once any comparison with other relevant DMU's does not provide evidence of inefficiency in the use of any input or output. In the sequel this concept of relative efficiency will be translated into a workable mathematical formulation.

Absolute and relative efficiency

13.2 Relative efficiency optimization

In this chapter, the above relative efficiency measure of a DMU is defined mathematically by the ratio of a weighted sum of outputs to a weighted sum of inputs. This ratio can be maximized by allowing the best possible selection of

A ratio measure ...

nonnegative weights for each DMU separately. This implies the existence of as many optimization models as there are DMU's. Of course, the weights cannot be arbitrarily large, and thus have to be restricted as explained in this section.

A DMU is said to be efficient relative to other DMU's if the value of its ratio efficiency measure is at least as large as those of the other DMU's using the same weights.

... expresses
relative
efficiency

The following verbal model description expresses an optimization model for each DMU separately.

Verbal model

Maximize: *relative efficiency measure for a particular DMU,*

Subject to:

- *for all DMU's the corresponding relative efficiency measure is restricted to be less than or equal to 1.*

In the above formulation, the restriction of each ratio measure to be less than or equal to 1 is meant to indicate that both the numerator and the denominator are equally important in determining the relative efficiency.

The following symbols will be used.

Notation

Indices:

d	<i>decision making units</i>
i	<i>observed input categories</i>
j	<i>observed output categories</i>

Parameters:

a_{id}	<i>observed input level (> 0) of input i for DMU d</i>
b_{jd}	<i>observed output level (> 0) of output j for DMU d</i>
p	<i>element parameter with specific DMU as its value</i>

Variables:

x_{id}	<i>weight to be given to input i for DMU d</i>
y_{jd}	<i>weight to be given to output j for DMU d</i>

The relative efficiency ratio measure is defined for each DMU separately. For this reason, the element parameter p (a standard concept in AIMMS) is used in the second index position of each identifier. As stated previously, the objective is to maximize the ratio of a weighted sum of outputs to a weighted sum of inputs. This can be written as follows.

The objective
for each DMU
separately

$$\left(\sum_{j \in J} b_{jp} y_{jp} \right) / \left(\sum_{i \in I} a_{ip} x_{ip} \right)$$

The optimal selection of the nonnegative weights x_{ip} and y_{jp} are used to compare the performance of the other DMU's based on their values of the various input and output factors. By restricting the corresponding ratio to 1 for all DMU's (including DMU p), relative efficiency can be at most 1 (which can be interpreted as 100%).

Ratio constraint

$$\frac{\sum_{j \in J} b_{jd} y_{jp}}{\sum_{i \in I} a_{id} x_{ip}} \leq 1$$

A first mathematical formulation of the model can be stated as follows.

Summary of first formulation

Maximize:

$$\left(\sum_{j \in J} b_{jp} y_{jp} \right) / \left(\sum_{i \in I} a_{ip} x_{ip} \right)$$

Subject to:

$$\begin{aligned} \frac{\sum_{j \in J} b_{jd} y_{jp}}{\sum_{i \in I} a_{id} x_{ip}} &\leq 1 & \forall d \in D \\ x_{ip} &\geq 0 & \forall i \in I \\ y_{jp} &\geq 0 & \forall j \in J \end{aligned}$$

In the previous formulation you may alter any optimal solution by multiplying the weight variables with a constant. Such multiplication does not alter the input-output ratio's. By forcing the weighted sum of inputs (i.e. the denominator) in the objective function to be equal to 1 you essentially remove this degree of freedom. In addition, the nonlinear ratio constraints can be transformed into linear constraints by multiplying both sides of the inequalities with their positive denominator. The denominator is always positive, because (a) all input and output levels are assumed to be positive, and (b) the nonnegative input weights cannot all be 0 when the weighted sum of inputs in the objective function is equal to 1.

Some simple manipulations

The resulting linear programming formulation is now as follows.

Resulting linear program

Maximize:

$$\sum_{j \in J} b_{jp} y_{jp}$$

Subject to:

$$\begin{aligned} \sum_{j \in J} b_{jd} y_{jp} &\leq \sum_{i \in I} a_{id} x_{ip} & \forall d \in D \\ \sum_{i \in I} a_{ip} x_{ip} &= 1 \\ x_{ip} &\geq 0 & \forall i \in I \\ y_{jp} &\geq 0 & \forall j \in J \end{aligned}$$

The optimal value of the objective function for the particular DMU referenced through p is either 1 or less than 1. In the latter case, there must be one or more other DMU's which have a relative efficiency equal to 1 based on these same weights. If this were not the case, all output weights could be multiplied with a scalar greater than 1. This increases the optimal value of the objective function, which is a contradiction in terms. The subset of other DMU's with relative efficiency equal to 1 is referred to as the *reference set* of p .

*Concept of
reference set*

13.3 A worked example

Consider a chain of 30 stores with total revenue of roughly US\$ 72 million and total cost of roughly US\$ 68 million. The overall profit-before-tax slightly exceeds 5.5%, and senior management considers this too low for their type of business. As a result, they decide to initiate a study to assess the performance of their stores. In particular, they would like to make an initial selection of those stores that are relatively poor performers. These stores can then be investigated further prior to making any decisions regarding selling, closing or improving one or more of these poor performers.

Background

In Table 13.1 you will find the input and output factors for a total of 30 DMU's numbered (for simplicity) from 1 to 30. The number of factors is kept small in this example, but in real applications you may encounter several additional factors not mentioned here. The two main factors determining profit are 'Revenue' and 'Total Cost', measured in 1000's of dollars. The total cost figures have been split into 'Staff Cost' (variable cost) and 'Non-staff Cost' (fixed cost). The three non-financial performance measures are 'Customer Service', 'Competition' and 'Age of Store'. Customer service is expressed as a rating between 1 (lowest) and 10 (highest). Competition is a count of the number of competitors within a fixed driving distance. The age of a store is expressed in terms of months.

Available data

The study is initiated by senior management, and they control the way that the assessment is performed by choosing the input and output factors to be considered. As will be illustrated, such a choice will have a definite impact on the results and their conclusions. On the other hand, the weights associated with each of the factors are chosen to optimize the relative efficiency of each

*Management
versus DMU's*

	Input-Output Factors						
	Total	Staff	Non-staff	Age of	Competition	Customer	Revenue
	Cost [10 ³ \$/yr]	Cost [10 ³ \$/yr]	Cost [10 ³ \$/yr]	Store [month]	[-]	Service [-]	[10 ³ \$/yr]
DMU-01	1310	238	1072	18	11	8	1419
DMU-02	2091	459	1632	46	12	8	3064
DMU-03	930	154	776	36	9	5	987
DMU-04	3591	795	2796	34	9	7	3603
DMU-05	2729	571	2158	35	13	9	2742
DMU-06	2030	497	1533	57	7	6	2536
DMU-07	3247	558	2689	36	5	9	4320
DMU-08	2501	571	1930	40	12	7	3495
DMU-09	2299	407	1892	17	8	8	2461
DMU-10	2413	306	2107	23	16	5	1851
DMU-11	1450	458	992	49	18	10	1935
DMU-12	2758	494	2264	35	9	8	3558
DMU-13	1857	360	1497	59	25	5	2088
DMU-14	3195	618	2577	51	9	8	3963
DMU-15	3505	759	2746	38	12	9	3918
DMU-16	1408	313	1095	24	9	8	1693
DMU-17	1127	253	874	17	5	7	1196
DMU-18	1637	340	1297	21	13	7	1945
DMU-19	2305	551	1754	27	7	6	3207
DMU-20	1781	303	1478	34	29	4	1622
DMU-21	3122	642	2480	26	5	10	2334
DMU-22	2597	465	2132	20	11	6	1387
DMU-23	1817	335	1482	28	4	9	1969
DMU-24	3483	825	2658	53	11	6	3422
DMU-25	1954	424	1530	11	15	3	1189
DMU-26	1120	159	961	4	5	6	810
DMU-27	1408	248	1160	36	7	3	1081
DMU-28	3420	672	2748	44	7	9	3088
DMU-29	2242	364	1878	18	11	5	1796
DMU-30	2643	490	2153	27	6	7	3243

Table 13.1: Observed values per factor

particular DMU separately. It is thus possible that a particular weight can be zero, thereby eliminating the effect of the corresponding factor relative to the other factors. If a DMU with its own optimal weights cannot be 100% relative efficient, then it is not part of the reference set and thus subject to further scrutiny.

In each application a decision must be made as to which factors are output and which factors are input. In general, an output factor is a factor that refers to aspects of achievement, while an input factor is a factor that aids or hinders the production of the outputs. In this example, 'Revenue' and 'Customer

*Output versus
input*

Service' are achievement factors, while all other categories are considered as input factors. A priori, the category 'Age of Store' cannot be seen as always hindering or always aiding any of the achievement factors. As was indicated in the previous paragraph, it is senior management who decides on the particular assessment experiments, and they will only consider those inputs and outputs that are meaningful to them and their decisions.

In practical applications the number of input and output factors can be quite large. In order to make meaningful choices for assessment, it is customary to examine the correlations between the factors. The reasons to exclude a certain factor from consideration may be the fact that another factor is already considered between which there exists a high correlation. Consider the correlations in Table 13.2. As it turns out, there is a high correlation between 'Total Cost', 'Staff Cost' and 'Non-staff Cost'. In addition, there is a dependency between these cost categories, because the latter two add up to the first. This dependency shows up in the third experiment where all factors are considered and the optimal weight associated with 'Total Cost' is zero for each DMU.

Correlated data

	Total Cost	Staff Cost	Non-staff Cost	Age of Store	Competition	Customer Service	Revenue
Total Cost	1.000	0.916	0.994	0.331	-0.134	0.355	0.826
Staff Cost		1.000	0.865	0.466	-0.110	0.410	0.822
Non-staff Cost			1.000	0.283	-0.137	0.329	0.802
Age of Store				1.000	0.265	0.156	0.511
Competition					1.000	-0.343	-0.162
Customer Service						1.000	0.484
Revenue							1.000

Table 13.2: Correlations between the factors

In this example only three experiments are discussed. In practice, a large number of experiments will be performed. The first experiment uses the factor 'Revenue' as output and the factor 'Total Cost' as input. All other factors are ignored. In this experiment, the objective function is therefore a simple ratio, and you would expect only the most profitable DMU to be 100% relative efficient. This is indeed the case, and the reference set consists of DMU-02. The seven poor performers (starting from the worst) are DMU-22, DMU-25, DMU-26, DMU-21, DMU-10, DMU-27, and DMU-29.

First experiment

The second experiment does not focus exclusively on 'Revenue' as output, but also considers 'Customer Service' as output. After all, high customer service may lead to improved sales in the future. 'Total Cost' remains the only input. In this experiment, DMU-11 joins DMU-02 as 100% relative efficient. Note that when factors are added to an experiment and no factors are deleted, then no DMU's leave the reference set and only new ones can enter. One of the seven

Second experiment

poor performers, namely DMU-26, has improved its position relative to other DMU's. The order of the poor performers has also changed. The seven poor performers (starting from the worst) are now DMU-22, DMU-25, DMU-10, DMU-27, DMU-21, DMU-29, and DMU-20.

In the third and last experiment, both 'Revenue' and 'Customer Service' are considered as output, while all other factors are used as input. Such an experiment offers each DMU plenty of opportunity to improve its relative efficiency. As a result, twelve DMU's have become 100% relative efficient and form the reference set illustrated in Table 13.3. There is also some movement in the set of poor performers. Starting from the worst, they are DMU-22, DMU-27, DMU-25, DMU-24, DMU-20, DMU-28, and DMU-29.

*Third
experiment*

	Relative efficiency		Relative efficiency		Relative efficiency
DMU-01	1.0000	DMU-23	1.0000	DMU-10	0.8204
DMU-02	1.0000	DMU-26	1.0000	DMU-13	0.8106
DMU-03	1.0000	DMU-08	0.9841	DMU-05	0.7999
DMU-07	1.0000	DMU-30	0.9657	DMU-29	0.7360
DMU-09	1.0000	DMU-18	0.9652	DMU-28	0.7342
DMU-11	1.0000	DMU-12	0.9612	DMU-20	0.7099
DMU-16	1.0000	DMU-06	0.9339	DMU-24	0.7046
DMU-17	1.0000	DMU-14	0.9032	DMU-25	0.6876
DMU-19	1.0000	DMU-15	0.8560	DMU-27	0.5990
DMU-21	1.0000	DMU-04	0.8372	DMU-22	0.5271

Table 13.3: Optimal relative efficiencies with all factors considered

On the basis of these few experiments, senior management can only make some preliminary conclusions. It is certainly true that DMU-22 has been consistently the worst performer of all. In addition, the three poorest performers have been so in all three experiments, and DMU-29 has always been on the edge of being a poor performer. However, the extent to which any of these results can be interpreted in a context which is relevant to managing the organization, is not clear at this point. In practice, assessment questions can be analyzed through the type of mathematical and statistical analysis described in this chapter, but extensive and detailed subsequent analysis of selected DMU's is required before any sound decision by senior management regarding closure, investment, target setting, etc. can be made.

Conclusions

13.4 Computational issues

In this section you will find some observations regarding the quality of the numerical solution for the type of assessment model discussed in this chapter.

This section

Computational characteristics of the mathematical performance assessment model described in this chapter have been studied in the literature ([Ch96]). Numerical difficulties have been observed when

Numerical difficulties

- there are many DMU's,
- the number of inputs and outputs is large,
- the inputs and outputs are of different orders of magnitude, and
- the data sets for some DMU's are nearly identical.

It is not within the scope of this book to explain why these difficulties occur. Fortunately, there are some simple precautions you can take to reduce the likelihood of any numerical difficulties. Why these precautions may have a positive effect on the numerical quality of your solution is again outside the scope of this book.

If you want to be on the safe side, follow the two precautionary measures recommended in this paragraph. Their implementation will never lead to a deterioration of solution quality. The first precautionary measure is to scale your data such that the values of each factor center around the same constant value, say 1. The effect of this measure is that the weights will tend to assume the same *relative* order of magnitude. The second measure is to change the right-hand side of the constraint limiting the weighted input. Instead of using the value 1, you may want to use the total number of input factors used in the experiment. This will cause the *absolute* size of the weights to be of order 1. In this case the relative efficiency of each DMU is no longer measured as a fraction but as a multiple thereof. It is straightforward to implement these precautionary measures in AIMMS using the `Unit` attribute associated with parameters.

Precautionary measures

13.5 Summary

In this chapter a general framework for assessing the relative performance of multiple decision making units has been presented. The analysis uses the concept of relative efficiency. The approach is balanced in that senior management is allowed to construct the various assessment experiments, while each particular decision making unit gets its own optimal weights for each of the input-output factors selected for the experiments. The corresponding model is a linear program to be solved for each decision making unit. Some suggestions to improve numerical performance were made. A worked example with seven

input-output categories and thirty related decision making units was provided for illustrative purposes.

Exercises

- 13.1 Implement the mathematical program described at the end of Section 13.2 using the example data provided in Section 13.3. Repeat the three experiments described in Section 13.3, and observe the efficiency associated with the DMU's.
- 13.2 Implement the precautionary measures described in Section 13.4 as part of your model in AIMMS. Write a procedure that will work for any data set related to the input-output categories used in Section 13.3.
- 13.3 Can you explain for yourself why the optimal weight associated with the category 'Total Cost' in the third experiment is zero for each DMU?

Chapter 14

A Two-Level Decision Problem

This chapter studies a two-level decision problem. There is a government at the policy making level wanting to influence the behavior of several companies at the policy receiving level. The government intends to attain its objectives by implementing tax and subsidy policies. The individual companies make their own optimal decisions *given* the tax and subsidy rates announced by the government. A two-level model of the problem is presented, and subsequently solved using two alternative solution approaches. The first is intuitive and general, but requires extensive computations. The second approach is theoretical and less general, but is computationally efficient. Both approaches are described in detail, and can easily be implemented in AIMMS. A data set and some basic data manipulations are included for illustrative purposes.

This chapter

A two-level program similar to the one discussed in this chapter can be found in [\[Bi82\]](#).

Reference

Nonlinear Program, Auxiliary Model, Customized Algorithm, Mathematical Derivation, What-If Analysis, Worked Example.

Keywords

14.1 Problem description

Consider a set of manufacturing companies situated on a river, producing similar products. The demand for these products is such that each company operates at maximum capacity for maximum profitability. Unfortunately, these manufacturing plants all produce waste water, which is dumped into the river. This practice has caused the water quality to deteriorate.

Waste water

Through public and governmental pressure all the companies have installed waste treatment plants. These plants use filters for waste removal. The exact amount of waste removed is controlled by varying the number of filters and their cleaning frequency. The cost of removing waste from waste water becomes increasingly expensive as the waste is more dilute.

Waste treatment

The government monitors both the production of waste water and the amount of waste removed at each manufacturing site. The resulting data forms the basis of ecological water studies from which recommendations for maintaining river water quality are made.

Government monitoring

The government has various options to control the amount of waste dumped into the river. One option is to specify an annual quota for each manufacturer. However, this requires extensive negotiations with each manufacturer and may lead to court actions and resistance. A second and more workable option is to introduce a tax incentive scheme using its legislative power. The idea is to introduce a tax on waste and a corresponding subsidy on waste removal such that the total tax revenue covers not only the total subsidy expenditure, but also the cost of monitoring the plants. The effect of such a scheme, assuming a cost minimizing response from the manufacturing companies, is a reduction in the amount of waste dumped into the river.

Taxes and subsidies

The above proposed tax incentive scheme results in a two-level decision problem. At the higher *policy making* level, the government must decide a joint tax and subsidy policy to steer the waste removal decisions by the manufacturing companies at the lower *policy receiving* level. At this lower level, decisions are only made on the basis of cost minimization and are independent of any government objectives.

Two-level decision problem

It is assumed that the government has complete knowledge of the problem, while the manufacturing companies have only limited knowledge. The government has access to all waste production and removal data, and has insight into the cost associated with operating waste treatment plants. The individual manufacturing companies operate independently. They have no insight into the manufacturing operations of their competitors, nor are they aware of the precise objectives of the government.

Different knowledge levels

Consider an example consisting of four manufacturing companies with Table 14.1 representing the total amount of waste produced, the waste concentration per unit of waste water, and an operating efficiency coefficient for each waste treatment plant.

Example

	waste water [$10^3 m^3$]	waste concentration [kg/m^3]	efficiency coef. [$\$/kg/m^6$]
Company 1	2,000	1.50	1.0
Company 2	2,500	1.00	0.8
Company 3	1,500	2.50	1.3
Company 4	3,000	2.00	1.0

Table 14.1: Waste production by the manufacturing companies

If none of the companies removed waste, then the total annual amount of waste dumped is 15,250 [10^3 kg]. It will be assumed that the government has set its target level to 11,000 [10^3 kg], and that the total monitoring cost is estimated to be 1,000 [10^3 \$].

14.2 Model formulation

Following is a verbal model statement of the problem. The two-level relationship is reflected by the fact that the policy receiving submodels are nothing more than constraints in the policy making model.

Verbal Model

Goal: *the government wants to find a tax and subsidy rate*

Subject to:

- *total revenue from taxation equals total subsidy expenditure plus an amount covering the waste monitoring activities,*
- *the total amount of waste dumped by all companies is less than or equal to a fixed target level, and*
- *for each company the amount of waste dumped by that company is the result of individual cost minimization reflecting waste removal cost, taxes and subsidies.*

In the above formulation, the government decides on the tax and subsidy rates, while the individual companies decide on the level of waste removal *given* the government decision. As stated previously, these companies are not aware of any of the financial or environmental targets set by the government.

The verbal model statement of the two-level waste removal problem can be specified as a mathematical model using the following notation.

Notation ...

Parameters:

L	<i>target level of waste to be dumped annually [10^3 kg]</i>
K	<i>total annual cost of government monitoring [10^3 \$]</i>

... at the policy making level

Variables:

T	<i>tax rate for waste produced [\$ / kg]</i>
S	<i>subsidy rate for waste removed [\$ / kg]</i>

Index:

j	<i>manufacturing companies</i>
-----	--------------------------------

... at the policy receiving level

Parameters:

d_j	<i>waste concentration observed at j [kg/m^3]</i>
q_j	<i>waste water produced annually by j [10^3 m^3]</i>
c_j	<i>efficiency coefficient of company j [$\text{\\$} \cdot \text{kg}/\text{m}^6$]</i>

Variable:

x_j removal of waste water by j [kg/m³]

The tax-subsidy balance states that total annual government receipts from taxation must equal the total cost of government monitoring plus its total subsidy expenditure.

*Tax-subsidy
balance*

$$T \sum_j q_j (d_j - x_j) = K + S \sum_j q_j x_j$$

The total waste limitation constraint requires that the annual amount of waste dumped into the river is less than or equal to the target level.

*Total waste
limitation*

$$\sum_j q_j (d_j - x_j) \leq L$$

The cost minimization model for each company j concerns the choice of x_j given T and S . The objective function can be written as

*Policy receiving
submodels*

$$\text{Minimize:}_{0 \leq x_j \leq d_j | T, S} q_j \left[\left(\frac{c_j}{d_j - x_j} - \frac{c_j}{d_j} \right) + T(d_j - x_j) - Sx_j \right] \quad \forall j$$

The term $\frac{c_j}{d_j - x_j} - \frac{c_j}{d_j}$ denotes the cost associated with the removal of waste from each unit of waste water by manufacturing company j and it is non-linear. The functional form of this term is based on the filtering technology used in the waste treatment plants. Filtering cost becomes essentially infinite if all waste has to be removed from waste water. The coefficient c_j reflects the operating efficiency of each waste treatment plant, and its numeric value is based on historic data.

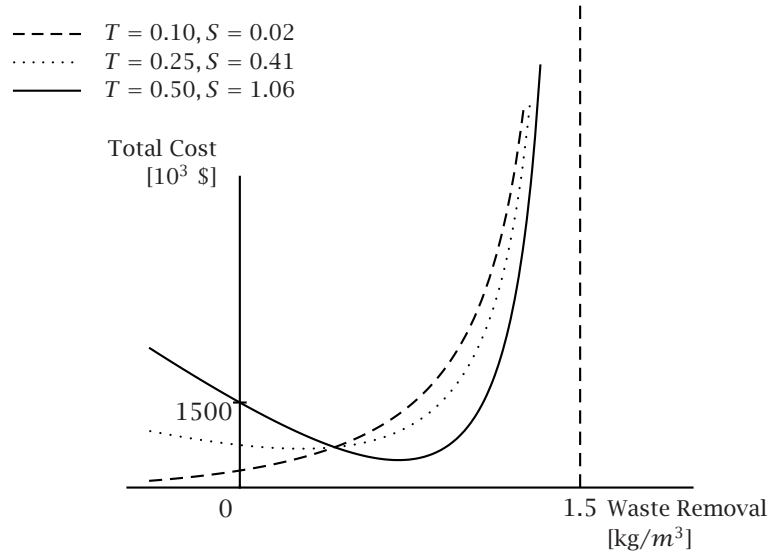
*Waste removal
cost term*

The term $T(d_j - x_j)$ denotes the company's tax expenditure associated with the left-over concentration of waste. The term Sx_j denotes the subsidy for waste removal per unit of waste water.

*Tax incentive
cost terms*

Note that the policy receiving models are unconstrained minimization models. The cost functions are strictly convex for all values of S and T . This implies the existence of a unique minimum and thus a unique response from the companies to the government. The curves in Figure 14.1 represent convex function contours, and illustrate the total cost of manufacturing company 1 as a function of the waste removal variable x_1 for several combined values of T and S .

Strict convexity

Figure 14.1: Cost function for company 1 for several T and S combinations

The following mathematical statement summarizes the model.

Model summary

Find: T, S

Subject to:

$$T \sum_j q_j (d_j - x_j) = K + S \sum_j q_j x_j$$

$$\sum_j q_j (d_j - x_j) \leq L$$

$$\text{Minimize: } q_j \left[\left(\frac{c_j}{d_j - x_j} - \frac{c_j}{d_j} \right) + T(d_j - x_j) - Sx_j \right] \quad \forall j$$

14.3 Algorithmic approach

This section describes an algorithm to compute the government tax and subsidy rates. The algorithm is an iterative scheme in which the government tax rate is adjusted each iteration. The process continues until the response from all companies is such that the total amount of annual waste dumped roughly equals the target level of total waste. It is straightforward to implement this algorithm in the AIMMS modeling language.

This section

The government realizes that any form of waste removal is costly, and that extra removal cost negatively impacts on the competitive position of the individual companies. Therefore, the waste limitation inequality (restricting the total amount of waste to be dumped into the river to be at most L) can be viewed as an *equality*.

Waste equality

Let $G = \sum_j q_j d_j$ denote the total amount of waste that would be dumped into the river without any removal activities. As indicated in the previous paragraph, L is the exact amount of waste that the government wants to be dumped. Under this assumption the tax-subsidy balance becomes $TL = K + S(G - L)$. Once the government decides on a value of T , the corresponding value of S is then computed using the formula $S = (TL - K)/(G - L)$.

Derive subsidy from tax

The above observations, coupled with the unique cost minimizing response from the manufacturing companies, forms the basis of the following simple computational scheme for the government.

Iterate on tax

1. Decide on an initial tax rate T .
2. Compute the corresponding subsidy rate S .
3. Announce these rates to the companies at the policy making level.
4. Let them respond with their waste removal decisions.
5. Compute the total amount of waste going to be dumped.
6. If approximately equal to L , then stop.
7. If less than L , then decrease T and go to step 2.
8. If greater than L , then increase T and go to step 2.

Note that if the amount of dumped waste is too high, the tax rate should go up. This increases the corresponding subsidy rate, thereby providing the companies with a higher incentive to remove more waste. If the amount of dumped waste is less than L , then the tax rate should be decreased. This lowers the subsidy rate and indirectly also the level of waste removal.

Proper response

The cost minimizing response from the manufacturing companies is a continuous function of the parameters T and S . This means that a small change in T (and thus in S) will result in a small change in the value of the corresponding removal variables x_j . In addition, the corresponding waste removal cost function value for each manufacturing company increases monotonically with an increase in the value of T . As a result, a straightforward bisection search over the tax rate T will lead to a combined tax and subsidy rate for which the amount of waste to be dumped is approximately equal to L . For these rates, the tax-subsidy balance is satisfied by construction.

*Convergence
bisection search*

In a bisection search algorithm both a lower bound LB and an upper bound UB on the tax rate T are required. An initial lower bound on T is the value for which the corresponding S becomes exactly zero. An initial upper bound on the value of T is not easily determined which is why there is an initial *hunt phase*. In the hunt phase, both a good upper bound and an improved lower bound are found. The computational procedure (in pseudo code) is summarized next.

Hunt phase

```

LB := K / L ;
REPEAT
  T := 2 * LB ;
  S := ( T * L - K ) / ( G - L ) ;
  Solve for x_j given T and S ;
  Compute total waste (to be dumped) ;
  BREAK WHEN total waste < target level ;
  LB := T ;
ENDREPEAT ;
UB := T ;

```

Note that the current value of LB is adjusted upwards each iteration until a proper upper bound UB has been determined. At the end of the hunt phase $UB = T$ and $LB = T/2$.

The interval found during the hunt phase is used as the starting interval for the *bisection phase*. In this phase either the lower or upper bound is set equal to the midpoint of the current interval until the final value of T has been determined. A small positive tolerance level TOL is introduced to assist with proper termination of this phase.

Bisection phase

```

TOL := 1.0E-3 ;
REPEAT
  T := ( LB + UB ) / 2 ;
  S := ( T * L - K ) / ( G - L ) ;
  Solve for x_j given T and S ;
  Compute total waste (to be dumped) ;
  BREAK WHEN abs(total waste - target level) < TOL ;
  If total waste < target level, then UB := T ;
  If total waste > target level, then LB := T ;
ENDREPEAT ;

```

14.4 Optimal solution

This section reports on the results of solving the two-level decision example using AIMMS and with the above algorithm implemented. In addition to the initial optimal solution, several sensitivity graphs with respect to the target level L are included.

This section

When applying the above algorithmic procedure to the example, the cost minimization models need to be solved for given values of T and S . However, during the solution process, the nonlinear solver reported 'Derivative evaluation error ... Division by zero'. This error occurred during the evaluation of the term $(\frac{c_j}{d_j - x_j} - \frac{c_j}{d_j})$ because the solver had made too large a step while increasing the value of x_j , causing the value of x_j to get too close to value of d_j .

Initial solver failure

When building nonlinear programming models, it is a good strategy to bound variables away from values for which the functions are not well-defined. The solver takes these bounds into account at all times during the solution process, thereby avoiding the occurrence of extremely large or small values of the decision variables. For the example, limiting x_j to $d_j - \epsilon$ with $\epsilon = 0.001$ was sufficient.

Add extra bounds

After adding the extra bound on x_j , another computational problem occurred. The nonlinear solver reported ‘Solver found error ... Jacobian element too large = 1.0E+06’. This error occurred during the evaluation of the derivative of the term $(\frac{c_j}{d_j - x_j} - \frac{c_j}{d_j})$ during the solution process. This type of error often occurs when the underlying model is badly scaled. Under these conditions, the problem can usually be avoided by adjusting the measurement unit associated with parameters, variables and constraints so that the values of these identifiers become comparable in size.

Subsequent solver failure

However, in the case of the example now under investigation, it turned out that the error was not due to bad scaling. Instead, the derivative of the first term $(\frac{c_j}{d_j - x_j} - \frac{c_j}{d_j})$ grows too fast when x_j approaches d_j . One option is to bound x_j even further away from d_j . Another option is to provide a starting value from which the algorithm does not search for new values near d_j . The last option was tried, and the x_j variables were initialized to $d_j/2$. With these starting values, the nonlinear solver solved the cost minimizing models to optimality, and the iterative procedure to compute the optimal tax value converged properly.

Provide starting solution

The iterative procedure to determine the optimal tax rate consisted of 2 steps to complete the hunt phase and 10 steps to complete the bisection phase. During the hunt phase, the initial lower bound LB increased from $K/L = 0.0909$ (1,000/11,000) to 0.1820 with a final upper bound twice that size. The optimal value of T was computed to be about \$0.239 per unit waste with a corresponding tax rate S of \$0.384 per unit waste removed. The solution values are summarized in Table 14.2. The total tax income for the government is 2,632 [10^3 \$] and the total subsidy expense is 1,632 [10^3 \$].

Optimal solution

	x_j [kg/m ³]	$q_j x_j$ [10 ³ kg]	Cleaning Cost [10 ³ \$]	Tax - Subsidy [10 ³ \$]	Total Cost [10 ³ \$]
Company 1	0.233	466.567	245.552	427.000	672.552
Company 2	-	-	-	598.145	598.145
Company 3	1.056	1,583.582	570.155	-89.702	480.453
Company 4	0.733	2,199.851	868.328	64.557	932.885

Table 14.2: Optimal removal levels and corresponding costs

Once a model is operational, it is quite natural to investigate its properties further by running some experiments. For instance, the government might want to investigate the effect of different target levels on the behavior of the individual manufacturing companies. The experiments reported in this section are based on letting the target pollution level L vary from a restricting value of 5,000 [10^3 kg] to the non-restricting value of 17,500 [10^3 kg]. The results are described in the next several paragraphs.

The effect of target levels ...

The waste removal curves in Figure 14.2 indicate that under the tax incentive scheme of the government, the largest waste producer will remove most of the waste. Company 2 is the second largest manufacturing company, but with the cleanest overall production process. The tax incentive program is no longer applicable for this company once the target level L set by the government is 10,000 [10^3 kg] or more. Note that the order in which companies no longer remove waste is the same as the order of waste concentrations d_j .

... on waste removal

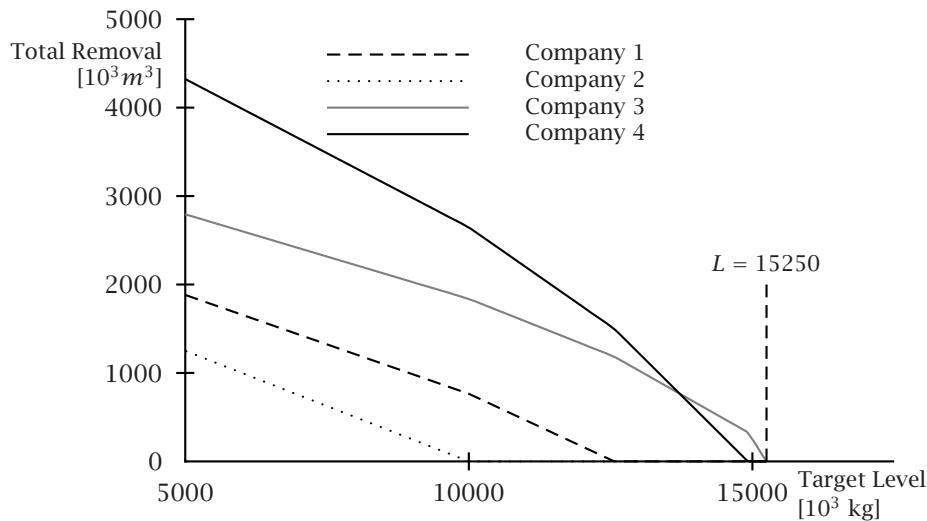


Figure 14.2: The amount of waste removed ($q_j x_j$) per company

Figure 14.3 shows that the total cost curve for each manufacturing company decreases as the value of L increases. This is to be expected as both the removal cost and the tax cost (due to a lower tax rate) will decrease as L increases. It is interesting to note that the companies with the lowest total waste (i.e. companies 1 and 2) have the highest total cost per unit waste when the target level L is low, and have the lowest total cost per unit waste when the target level L is no longer restricting.

... on total cost per unit waste

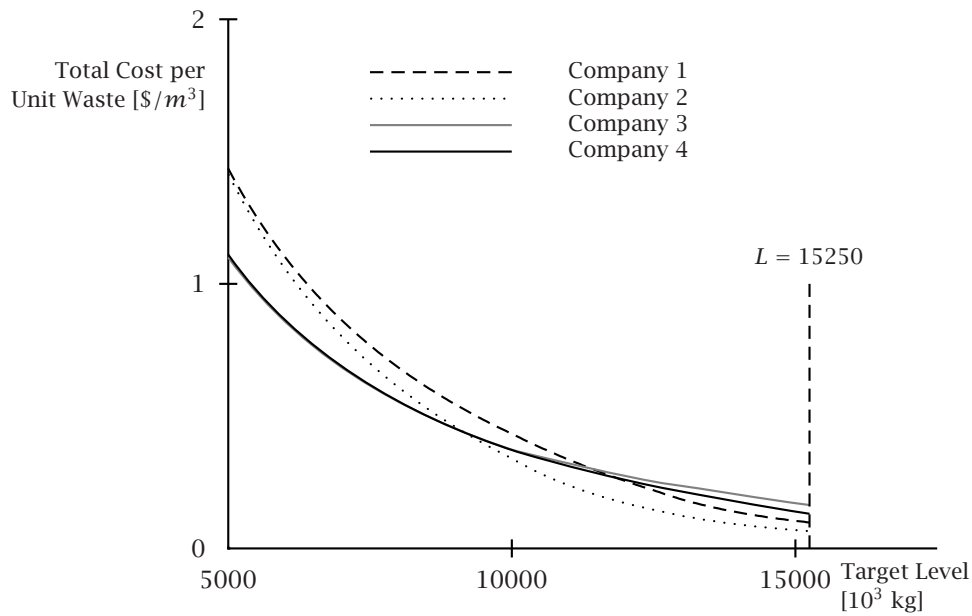


Figure 14.3: The total cost per unit waste
(i.e. $\frac{c_j}{d_j - x_j} - \frac{c_j}{d_j} + T(d_j - x_j) - Sx_j$)

The two curves in Figure 14.4 compare the tax and subsidy rates as a function of the target level L . Note that the tax rate is the highest when the target level is the lowest. This is necessary to induce the companies to remove a significant amount of their waste in order to reach that target level.

... on tax & subsidy rates

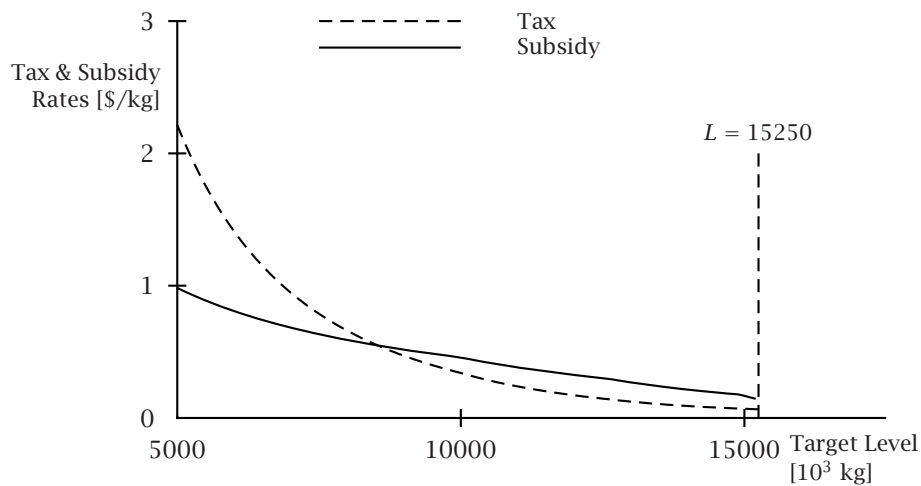


Figure 14.4: Optimal tax and subsidy rates

The curves in Figure 14.5 indicate that the companies with the lowest initial waste concentrations (i.e. companies 1 and 2) have a net expenditure to the government, while the companies with the highest initial waste concentrations (i.e. companies 3 and 4) have a net income from the government. It seems unfair that largest contributors to the overall waste dumped into the river receive the largest subsidies to remove it. On the other hand, their actions have the strongest effect on the amount of waste to be removed through filtering. As shown in the next section, given the status quo, the tax incentive program seems to lead to the most efficient solution for society as a whole.

... on payments
to/from the
government

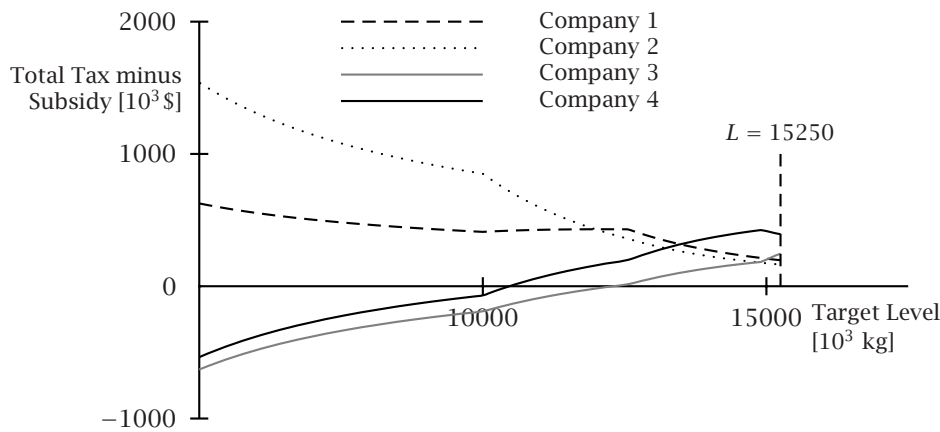


Figure 14.5: The total tax - subsidy per company
(i.e. $q_j(T(d_j - x_j) - Sx_j)$)

Observe that for each fixed value of the target level L , the sum of the four function values equals the total annual cost K of government monitoring. For the case when $L \geq 15,250[10^3\text{kg}]$, the subsidy is zero and the aggregated tax paid covers the monitoring cost.

14.5 Alternative solution approach

The solution approach described in Section 14.3 is an intuitive and correct way to solve the two-level model. In this section an alternative single-step solution method is presented. This alternative method is not as general as the previous algorithmic approach, but turns out to be very efficient in this case. The method consists of solving a single auxiliary model, followed by a simple computation to determine the corresponding values of T and S . The proof of correctness is based on examining the underlying optimality conditions.

This section

The cost minimization model for each individual manufacturing company j , from Section 14.2, is as follows.

*Optimality
policy receiving
models*

$$\text{Minimize: } q_j \left[\left(\frac{c_j}{d_j - x_j} - \frac{c_j}{d_j} \right) + T(d_j - x_j) - Sx_j \right] \quad \forall j$$

As previously stated, these optimization models are strictly convex. This implies the existence of a necessary and sufficient optimality condition for each company j . This condition results from setting the first derivative with respect to the variable x_j equal to zero.

$$q_j \left[\frac{c_j}{(d_j - x_j)^2} - (T + S) \right] = 0 \quad \forall j$$

There is a unique solution value x_j for each unique value of $T + S$.

Consider the following model in which there is no tax incentive scheme, and all companies work together to attain the goal set out by the government. The total cost of removing waste *for all companies combined* is minimized subject to the restriction that the overall waste production must equal the target level L . This situation is not fully realistic since the companies do not actually collaborate. However, such a model will produce the target waste level at the lowest overall cost.

*An auxiliary
model*

Minimize:

$$\sum_j q_j \left(\frac{c_j}{d_j - x_j} - \frac{c_j}{d_j} \right)$$

Subject to:

$$\sum_j q_j (d_j - x_j) = L$$

This model is a constrained convex minimization model with a unique solution. The necessary and sufficient conditions for optimality are derived from the associated Lagrangian function

*Optimality
auxiliary model*

$$L(\dots, x_j, \dots, \lambda) = \sum_j q_j \left(\frac{c_j}{d_j - x_j} - \frac{c_j}{d_j} \right) - \lambda \left[\sum_j q_j (d_j - x_j) - L \right]$$

By setting the first derivatives with respect to the variables x_j (for each j) and λ equal to zero, the following conditions result.

$$\begin{aligned} q_j \left[\frac{c_j}{(d_j - x_j)^2} + \lambda \right] &= 0 \quad \forall j \\ \sum_j q_j (d_j - x_j) &= L \end{aligned}$$

Note that the optimal value of λ is such that all equations are satisfied. In addition, observe that the optimality condition for the policy receiving models and the first set of optimality conditions for the above auxiliary model are similar in structure. By equating the quantities $-(T+S)$ and λ , these optimality conditions become identical.

Similarity in optimality conditions

By solving the auxiliary model and using the value of λ (the shadow price of the waste equality constraint) produced by the nonlinear solution algorithm, the optimal value of $T + S$ is determined. This, together with the known relationship between the values of T and S , give rise to two equations in two unknowns.

Optimal tax setting

$$\begin{aligned} -(T + S) &= \lambda \\ S &= (TL - K)/(G - L) \end{aligned}$$

The solution for T and S can be obtained after simple manipulations.

$$\begin{aligned} T &= (K - \lambda(G - L))/G \\ S &= (-K - \lambda L)/G \end{aligned}$$

The value of λ obtained from solving the auxiliary model for the example data provided in Section 14.1 is -0.6232. The resulting value of T is 0.239 [\$/kg] and the corresponding value of S is 0.384 [\$/kg]. These values, when provided as input to the individual minimizing manufacturing companies, produce the same response x_j as the values of x_j obtained by solving the auxiliary model.

Verifying the results

14.6 Summary

This chapter presented a worked example of a two-level decision problem. The example was of a government wanting to control the annual amount of waste dumped into a river by setting a tax for waste production and a subsidy for waste removal. The manufacturing companies select their own waste removal level based on cost minimization considerations. A two-level model was developed in detail and two distinct solution procedures were proposed. The first procedure is a bisection algorithm to determine the optimal tax rate for the government. This approach consists of a hunt phase to determine a search interval, followed by a bisection phase to determine the optimal solution. The second solution procedure is based on solving an auxiliary model in which all individual companies collaborate to meet collectively the target waste level set by the government. It was demonstrated that the shadow price on the waste limitation constraint provided enough information to determine the optimal tax rate. Both the optimal solution and the results of sensitivity experiments were reported in detail.

Exercises

- 14.1 Implement the policy receiving submodel of Section 14.2 using the data provided in Section 14.1, together with a fixed tax rate of 0.25 and a fixed subsidy rate of 0.41. Verify that the solution produced with AIMMS coincides with a point along the curve presented in Figure 14.1.
- 14.2 Implement the algorithm approach described in Section 14.3, and perform the experiments explained in Section 14.4 to study the effects of changing target levels.
- 14.3 Implement the alternative solution approach described in Section 14.5, and verify whether the results of the two algorithmic approaches coincide.

Chapter 15

A Bandwidth Allocation Problem

This chapter introduces a bandwidth allocation problem and presents two different ways to formulate a binary programming model of it. The incentive to develop the second model formulation arose when the implementation of the first model became unwieldy. For both models, techniques for reducing the complexity of the constraint matrix are demonstrated. Both representations can easily be implemented using the AIMMS modeling language.

This chapter

Integer Program, Mathematical Reformulation, Worked Example.

Keywords

15.1 Problem description

As a result of the growing number of mobile communication systems, there is an increasing need to allocate and re-allocate bandwidth for point-to-point communications. Bandwidth allocations typically remain operational for seconds/minutes (in cellular communications), days/weeks (in military communication systems) or months/years (in television and radio communication systems). During these operational periods the volume of traffic usually changes significantly, which causes point-to-point capacity and interference problems. Consequently, bandwidth allocation is a recurring process in practice. In this chapter a specific bandwidth allocation problem is examined.

*Bandwidth
planning
problems*

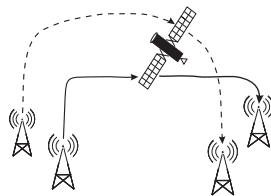


Figure 15.1: A satellite communication system

Consider a satellite communication system, as shown in Figure 15.1, where the ground stations either transmit or receive messages via the satellite. A *link* in such a communication system is any pair of communicating ground stations. The *bandwidth domain* is the specific range of channels available for allocation. Such a range can be divided up into fixed-width portions, referred to as *channels*. Any specific link requires a pre-specified number of adjacent channels, which is referred to as a *bandwidth interval*. The concepts of “channel” and “bandwidth interval” are illustrated in Figure 15.2. *Link interference* represents a combined measure of the transmitter and receiver interference as caused by other existing communications systems.

Basic terminology

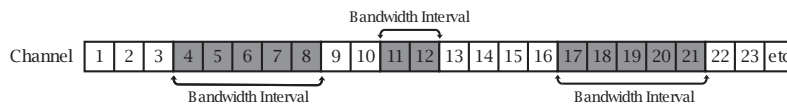


Figure 15.2: Channels and intervals in a bandwidth domain

A *bandwidth allocation* is the assignment of a bandwidth interval to at most one link in the communication system. An *optimal bandwidth allocation* is one in which some measure of total interference is minimized.

The problem summarized

For a given link, the overall level of transmitter and receiver interference is dependent on the interference over its entire bandwidth interval. The model formulation in this chapter assumes that interference data is available on a per channel basis for each link. Furthermore, for each interval-link combination it is assumed that the overall interference experienced by the link is equal to the value of the maximum channel interference that is found in the interval.

Constructing link interference

This chapter illustrates the bandwidth allocation problem using a small example data set consisting of three communication links with seven adjacent channels available for transmission. The first link requires one channel for transmission, while both the remaining two links must be allocated a bandwidth interval containing three channels. Table 15.1 presents the interference level for each link on a per channel basis. Using this data, the overall interference of each interval-link is found by identifying the maximum channel interference in the corresponding bandwidth interval. These values are presented in Table 15.1.

Running example

Based on the simple verbal description of the problem, there is the initial expectation that it should be straightforward to formulate a model of the problem. However, for large instances of the problem this is not the case. This chapter presents two model formulations. The first formulation relies on enumerating all possible intervals and can have implementation difficulties. The second formulation presents a better symbolic representation which improves implementation.

Model formulation

	Channel-link interference			Interval-link interference		
	Link 1	Link 2	Link 3	Link 1	Link 2	Link 3
Channel 1	4	5	6	4	8	8
Channel 2	8	8	1	8	8	8
Channel 3	7	1	8	7	2	8
Channel 4	9	2	7	9	8	7
Channel 5	1	1	1	1	8	3
Channel 6	5	8	2	5	-	-
Channel 7	4	5	3	4	-	-

Table 15.1: Channel and interval interference data

15.2 Formulation I: enumerating bandwidth intervals

This formulation relies on first enumerating all possible intervals of a fixed size. Next, binary variables (yes/no choices) are defined to assign them to communication links of the same size. With seven channels, three links, and two different interval widths, there are twelve possible positioned intervals (seven of width one, and five of width three) numbered from 1-12 as shown in Figure 15.3. These positioned interval numbers are used throughout this section. It should be noted that these numbers have the disadvantage that they do not show any relationship to either the size or the channel numbers contained in the interval.

All bandwidth intervals

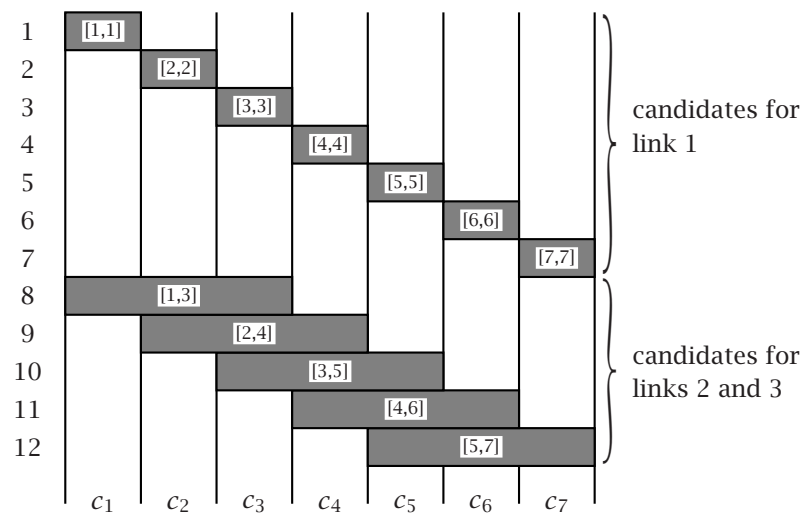


Figure 15.3: Twelve possible positioned intervals

Together with the positioned interval numbers, the following notation is used. *Notation*

Indices:

p *positioned intervals*
 l *links*

Parameter:

d_{pl} $\begin{cases} 1 & \text{if positioned interval } p \text{ has width required} \\ & \text{by link } l \\ 0 & \text{otherwise} \end{cases}$

Variable:

x_{pl} $\begin{cases} 1 & \text{if positioned interval } p \text{ is assigned to link } l \\ 0 & \text{otherwise} \end{cases}$

Using the above notation and a previous introduction to the assignment model (see Chapter 5), the following two symbolic constraints can be formulated.

Two apparent constraints

$$\begin{aligned} \sum_p d_{pl} x_{pl} &= 1 & \forall l \\ \sum_l d_{pl} x_{pl} &\leq 1 & \forall p \end{aligned}$$

The first constraint makes sure that a link l uses exactly one of the positioned intervals, and the second constraint makes sure that each positioned interval p is allocated at most once.

In case you are not (yet) comfortable with the above symbolic notation, Table 15.2 presents the constraints for the example problem in tabular format. The variable names are on top, and the coefficients are written beneath them. At this point there are 15 individual constraints (3+12) and 17 individual variables (7+5+5).

The constraints in tabular form

The formulation developed so far is incomplete. It is missing a mechanism that will ensure that selected positioned intervals do not overlap. Without such a mechanism the variables $x_{5,1}$ and $x_{11,2}$ can both be equal to 1, while their corresponding positioned intervals [5,5] and [4,6] overlap.

What is missing?

To handle this situation, there are at least two approaches. One approach is to build constraints that restrict allocations so there is no overlap of their corresponding bandwidth intervals. The other (less obvious) approach is to define for each variable which channels are covered, and to add a constraint for each channel to limit its use to at most once. Even though these approaches sound distinct, they eventually lead to the same model formulations. This equivalence is shown in the next two subsections.

Two approaches to avoid overlap

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
p	1	2	3	4	5	6	7	8	9	10	11	12	8	9	10	11	12		
l	1	1	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3		
1	1	1	1	1	1	1	1											=	1
2								1	1	1	1	1						=	1
3													1	1	1	1	1	=	1
1	1																	≤	1
2		1																≤	1
3			1															≤	1
4				1														≤	1
5					1													≤	1
6						1												≤	1
7							1											≤	1
8								1					1					≤	1
9									1					1				≤	1
10										1					1			≤	1
11											1					1		≤	1
12												1					1	≤	1

Table 15.2: The individual assignment constraints

15.2.1 Preventing overlap using pairs of allocations

This approach relies on building constraints that prevent allocations with overlap. One way to accomplish this is to first identify all pairs of positioned intervals that overlap, and then to write a constraint for each pair to prevent the overlap. This approach is a form of enumeration. For our example, there are at most $\binom{17}{2} = (17^2 - 17)/2 = 136$ pairs of associated decision variables must be considered to form constraints that restrict overlap. An analysis concludes there are only 63 pairs of overlapping intervals.

Consider pairs of allocations

Fortunately, the 63 constraints identified to avoid overlap can be combined to form a much smaller set of equivalent constraints. To illustrate how, consider the following three restrictions.

Constraint reduction

$$\begin{aligned}
 x_{3,1} + x_{9,2} &\leq 1 \\
 x_{3,1} + x_{10,3} &\leq 1 \\
 x_{9,2} + x_{10,3} &\leq 1
 \end{aligned}$$

The three positioned intervals represented by numbers 3, 9 and 10 correspond to bandwidth intervals [3,3], [2,4] and [3,5] respectively. Since all three intervals include channel 3, it is possible to add the three constraints together to obtain the following single constraint.

$$2x_{3,1} + 2x_{9,2} + 2x_{10,3} \leq 3$$

Since all of the variables are either zero or one, this constraint can be rewritten after dividing by 2 and then rounding the right-hand side downwards.

$$x_{3,1} + x_{9,2} + x_{10,3} \leq 1$$

This constraint replaces the previous three constraints, and allows for exactly the same *integer solutions*. When viewed in terms of *continuous variables*, this single constraint represents a much tighter formulation. Consider for instance the point (0.5, 0.5, 0.5). This is a feasible point for the first three constraints, but not for the combined constraint. In general, tighter constraints are preferred, because they help the solution algorithm to converge faster.

New constraint is tighter

The reduction scheme can be applied to groups of allocations where their bandwidth intervals have pairwise overlap. The process becomes somewhat more involved when the size of a group of allocations with pairwise overlap increases. The step of extracting the 63 overlapping intervals from all possible intervals takes time, but the subsequent step to reduce the constraint set to just 7 new constraints takes even longer. The new constraints are numbered 1-7, and listed in tabular format in Table 15.3.

Reduced constraint set

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
p	1	2	3	4	5	6	7	8	9	10	11	12	8	9	10	11	12	
l	1	1	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	
1	1							1					1					≤ 1
2		1						1	1				1	1				≤ 1
3			1					1	1	1			1	1	1			≤ 1
4				1					1	1	1			1	1	1		≤ 1
5					1					1	1	1			1	1	1	≤ 1
6						1					1	1				1	1	≤ 1
7							1					1					1	≤ 1

Table 15.3: The individual constraints on overlap

15.2.2 Preventing overlap using channel constraints

This approach generates the same seven constraints that resulted from the first approach but they are generated directly. The key observation is that there are seven channels, and that the seven constraints from the first approach actually represent a constraint for each of the seven channels.

Inspect results of first approach

Let constraint c be associated with the set of channels C ($C = \{1, 2, \dots, 7\}$). Then the coefficients in each constraint c correspond exactly to those variables with positioned intervals that overlap with channel c . Using this new viewpoint, it is possible to formulate one constraint for each channel directly.

Overlap with channels

Let c refer to channels and define the three-dimensional parameter a_{cpl} as the 'cover' matrix (c refers to rows and pl refers to columns). Specifically,

The 'cover' matrix

$$a_{cpl} = \begin{cases} 1 & \text{if the (positioned interval, link) pair } pl \text{ contains channel } c \\ 0 & \text{otherwise} \end{cases}$$

With this notation you can write the following symbolic constraint to ensure that intervals do not overlap. The individual constraints that make up this symbolic constraint are exactly the same as those in Table 15.3 from the first approach.

The new constraint

$$\sum_{(pl)} a_{cpl} x_{pl} \leq 1 \quad \forall c$$

15.3 Formulation II: avoiding bandwidth interval construction

In this section the model is reformulated using different notation. The new formulation avoids the process of positioned interval construction used in Section 15.2.1.

This section

In the previous formulation all possible bandwidth intervals were enumerated, which required both a construction process and a matching with links. This entire process can be avoided by letting the variables refer indirectly to the set of enumerated intervals. To do this, variables are defined with both channel and link indices. The channel index represents the channel number where the link bandwidth interval *begins*. The end of the interval is calculated from the known length of the link.

Revise variable definition

Parameter:

$$a_{\hat{c}cl} = \begin{cases} 1 & \text{if the interval for link } l \text{ starting at channel } c \text{ also covers channel } \hat{c} \\ 0 & \text{otherwise} \end{cases}$$

Notation

Variable:

$$x_{cl} = \begin{cases} 1 & \text{if the interval for link } l \text{ starts at channel } c \\ 0 & \text{otherwise} \end{cases}$$

The following model constraints can now be written.

Model constraints

$$\begin{aligned} \sum_c x_{cl} &= 1 \quad \forall l \in L \\ \sum_{cl} a_{\hat{c}cl} x_{cl} &\leq 1 \quad \forall \hat{c} \in C \end{aligned}$$

The first constraint makes sure that for each link there is exactly one channel at which its corresponding bandwidth interval can start. The second constraint

makes sure that for each channel at most one of all overlapping intervals can be allocated.

Note that the variable references in the above symbolic constraints do not reflect their domain of definition. For instance, the variable $x_{6,2}$ is not defined, because a bandwidth interval of length 3 for link 2 cannot start at channel 6. These domain restrictions, while not explicit in the above symbolic model, must be clearly specified in the actual model formulation. This can be easily implemented using the AIMMS modeling language.

Be aware of domain restrictions

As described at the beginning of this chapter, the objective of the bandwidth allocation problem is to minimize a specific measure of total communication interference. To this end, the following notation is defined.

The objective function

g_{cl}	<i>interference for link l whenever channel c is part of the interval allocated to this link</i>
w_{cl}	<i>maximum interference for link l whenever its interval starts at channel c</i>
J_{cl}	<i>collection of channels that comprise the particular interval for link l beginning at channel c</i>

The interference assigned to a link is the maximum channel interference w_{cl} that occurs in the bandwidth interval.

$$w_{cl} = \max_{\hat{c} \in J_{cl}} g_{\hat{c}l} \quad \forall (c, l)$$

The model formulation with objective function and constraints is summarized below.

The entire formulation

$$\begin{aligned} \min \quad & \sum_{cl} w_{cl} x_{cl} \\ \text{s.t.} \quad & \sum_c x_{cl} = 1 \quad \forall l \\ & \sum_{cl} a_{\hat{c}cl} x_{cl} \leq 1 \quad \forall \hat{c} \in C \\ & x_{cl} \text{ binary} \end{aligned}$$

In the algebraic description of the model, no attention has been paid to restrictions on the index domain of any of the identifiers. The domain conditions on the decision variables, for instance, should make sure that no bandwidth interval extends beyond the last available channel of the bandwidth domain.

Domain checking

In the optimal solution for the problem instance described in Section 15.1, link 1 is assigned to channel 1, link 2 is assigned to channels 2 through 4, and link 3 is assigned to channels 5 through 7. The corresponding total interference is 15.

Solution

15.3.1 Improving sparsity in overlap constraints

After analyzing the coefficient matrix associated with the individual overlap constraints, some simple mathematical manipulations can be performed to obtain a potentially significant reduction in the number of nonzero coefficients.

Analyzing the coefficient matrix

Before the constraint matrix is manipulated, it is necessary to add a zero-one slack variable to each overlap constraint. For the worked example case, this is illustrated in Table 15.4.

Add slacks to overlap constraints

c l	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	s	s	s	s	s	s	
	1	2	3	4	5	6	7	8	9	10	11	12	8	9	10	11	12	1	2	3	4	5	6	7
1	1							1						1				1						= 1
2		1						1	1					1	1				1					= 1
3			1					1	1	1				1	1	1				1				= 1
4				1					1	1	1				1	1	1				1			= 1
5					1					1	1	1				1	1	1				1		= 1
6						1					1	1					1	1				1		= 1
7							1					1					1					1		= 1

Table 15.4: Constraints on overlap as *equalities*

Consider rows i , $i \geq 2$ and subtract from each row its previous row $i - 1$. This results is a special matrix, in which each column has at most two coefficients. This matrix is presented in Table 15.5. Note that a column has either a 1 and -1 or just a 1.

Subtract previous rows

c	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	s	s	s	s	s	s
	1	2	3	4	5	6	7	8	9	10	11	12	8	9	10	11	12	1	2	3	4	5	6	7
1	1							1					1					1						
2	-1	1							1					1				-1	1					
3		-1	1							1					1				-1	1				
4			-1	1				-1			1		-1			1				-1	1			
5				-1	1				-1			1		-1			1				-1	1		
6					-1	1				-1					-1						-1	1		
7						-1	1				-1						-1					-1	1	

Table 15.5: Overlap constraints with a most 2 nonzeros per column

Whenever the length of a link is greater than or equal to three, there will be a reduction in the number of coefficients for that link. The overall savings are particularly significant in those applications where the average link width is much larger than three.

*Reduction in
nonzero
elements*

15.4 Summary

In this chapter the process of formulating and then reformulating a binary programming model was demonstrated. The need to reformulate the model arose because of difficulties encountered when implementing the first. In both formulations, techniques to reduce the complexity of the constraint matrix were illustrated. The process of model evolution, as demonstrated in this chapter, is quite common in practical modeling situations. This is especially true when the underlying model is either a binary or a mixed-integer programming model. For this class of models it is often worthwhile to consider alternative formulations which are easier to work with and which may result in strongly reduced solution times.

Exercises

- 15.1 Implement the bandwidth allocation model according to Formulation I as presented in Section 15.2 using the data provided in Section 15.1.
- 15.2 Implement the bandwidth allocation model according to Formulation II as presented in Section 15.3, and verify whether the two formulations in AIMMS produce the same optimal solution.
- 15.3 Implement the variant of Formulation II described in Section 15.3.1, in which there are at most two nonzero elements for each column in the overlap constraints. Check whether the optimal solution remains unchanged.

Chapter 16

A Power System Expansion Problem

In this chapter you will encounter a simplified power system expansion problem with uncertain electricity demand covering a single time period. Through this example you will be introduced to some methods for handling uncertainty in input data. The first approach to handle uncertainty starts with a deterministic model, and examines its sensitivity to changes in the values of uncertain parameters. This is referred to as *what-if analysis*, and is essentially a manual technique to deal with uncertainty. The second approach in this chapter goes one step further, and captures the input data associated with an entire what-if analysis into a single model formulation. This second approach is described in detail, and will be referred to as *stochastic programming*.

This chapter

There is a vast literature on stochastic programming, but most of it is only accessible to mathematically skilled readers. The example in this chapter captures the essence of stochastic programming, and has been adapted from Malcolm and Zenios [Ma92]. Two selected book references on stochastic programming are [In94] and [Ka94].

References

Linear Program, Stochastic Program, Two-Stage, Control-State Variables, What-If Analysis, Worked Example.

Keywords

16.1 Introduction to methods for uncertainty

The mathematical programming models discussed so far have had a common assumption that all the input information is known with certainty. This is known as “decision making under certainty.” The main problem was to determine which decision, from a large number of candidates, yields the best result, according to some criterium. Models that account for uncertainty are known as *stochastic* models—as opposed to *deterministic* models, which do not.

*Deterministic
versus
stochastic
models*

Stochastic models contain so-called *event parameters* which do not have a priori fixed values. An event is an act of nature which falls outside the control of a decision maker. A typical example of an event parameter is the demand of products inside a decision model that determines production levels. The exact demand values are not known when the production decision has to be made,

*Event
parameters*

but only become known afterwards. Beforehand, it only makes sense to consider various data realizations of the event parameter demand, and somehow take them into account when making the production decision.

The number of possible data realizations of event parameters is usually very large. In theory, it is customary to relate the event parameters to continuous random variables with infinitely many outcomes. In practical applications, however, it turns out that it is better from a computational point of view to assume a finite number of data realizations. One complete set of fixed values for all of the event parameters in a stochastic model is referred to as a *scenario*. Throughout the sequel, the assumption is made that there are only a manageable number of scenarios to be considered when solving a stochastic model. In addition, it is assumed that the likelihood of each scenario occurring is known in advance.

Assume few scenarios

A stochastic model contains two types of variables, namely control and state variables. *Control variables* refer to all those decision variables that must be decided at the beginning of a period prior to the outcome of the uncertain event parameters. *State variables*, on the other hand, refer to all those decision variables that are to be determined at the end of a period after the outcome of the uncertain event parameters are known. Thus, the variables representing production decisions are control variables, while variables representing stock levels are state variables.

Control and state variables

A first approach to uncertainty is to build a deterministic model and assume fixed values for the uncertain event parameters. You can then perform an extensive what-if analysis to observe changes in the control variables as the result of assuming different fixed values for the event parameters. The underlying motivation of this manual approach is to discover a single set of values for the control variables that seem to be a reasonable choice for all possible scenarios.

What-if approach

A second approach to uncertainty is to formulate an extended model in which all scenarios are incorporated explicitly. By using the scenario likelihoods, the objective function can be constructed in such a way as to minimize expected cost over all possible scenarios. The advantage of this approach is that not you but the model determines a single set of values for the control variables. This approach is referred to as stochastic programming.

Stochastic programming approach

The term *two-stage* will be attached to the stochastic programming approach to indicate that decisions need to be made prior to and after the realization of the uncertain events during a single time period. In Chapter 17 the generalization is made to multiple time periods, where the term *multi-stage* reflects a sequence of two-stage decisions.

Two-stage versus multi-stage

16.2 A power system expansion problem

In this section you will encounter a somewhat simplified but detailed example to determine new power plant design capacities to meet an increase in electricity demand. The example forms the basis for illustrating the approaches to stochastic modeling discussed in the previous section.

This section

The design capacity of a power plant can be defined as the maximum amount of energy per second it can produce. It is assumed that energy in the form of electricity is not stored. This implies that at any moment in time, total available supply of electricity must exceed the instantaneous electricity demand.

*Design capacity
...*

Assume that new investments in design capacity are under consideration in order to meet an increase in electricity demand. The demand is uncertain since actual demand will be strongly influenced by actual weather and economic factors. When the new capacity is installed, it remains available for an extensive period of time. This makes the design capacity investment decision a nontrivial one. When the design capacity exceeds demand, the overall capital cost is likely to be too high. Alternatively, when the capacity is insufficient, the plants can be expected to operate at peak capacity, and extra supply will need to be imported. Both these events are costly in terms of either purchase cost or operating cost.

*... must be
expanded*

In this example, electricity will be produced by three types of power plants, namely coal-fired, nuclear, and hydro plants. Each of these have their own specific capital and operating costs. In this example it is assumed that design capacity does not come in fixed sizes, but can be built in any size. Imported electricity is to be a last resource of supply, and will only be used when the installed design capacity is insufficient.

*Available plant
types and no
fixed sizes*

Electricity demand varies over days, weeks, seasons, and years. Rather than model demand over extended periods of time, a conscious choice has been made to only consider a single time period of one day. This choice simplifies the model to be built, but still allows for demand scenarios that typify demand throughout a planning horizon of years. The number of scenarios to be considered is assumed to be finite. Their number does not need to be large, as long as there are enough scenarios to form a representative sample of future demand.

*Uncertain
demand*

Determining a representative sample of daily future demand instances is nontrivial. When are scenarios sufficiently distinct? Can you keep their number under control? What is the effect on model solutions when particular scenarios are left out? How likely is each scenario? These questions need to be dealt with

*Scenario
selection*

in practical applications, but are not treated in this chapter. Instead, a limited number of scenarios and their probabilities are provided without additional motivation.

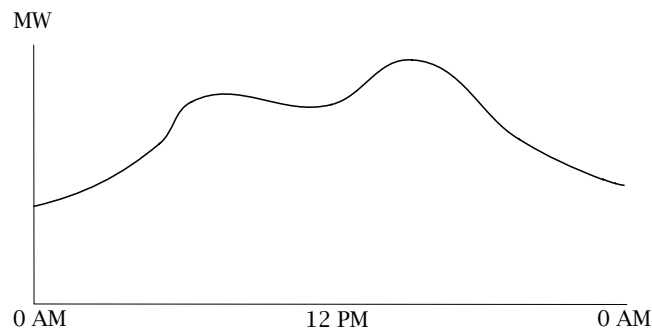


Figure 16.1: Daily load duration curve

The daily electricity demand is by no means constant. The demand curve is tightly linked to economic and household activities: more electricity is used at noon than at midnight. A *load duration curve*, such as Figure 16.1, reflects the electricity demand during one day. Rather than storing a continuous curve, it is more convenient to approximate such a curve by considering fixed-length time periods consisting of say one or two hours. This is shown in Figure 16.2 where there are twelve different demands, corresponding to one of twelve time periods.

Daily demand

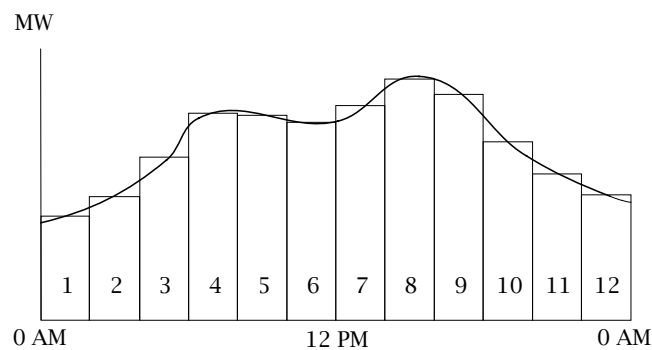


Figure 16.2: Daily load duration curve approximated by a step function

Twelve time periods could be considered a large number in a study dealing with the strategic problem of capacity expansion. Lengthening the time period beyond two hours, however, would cause the approximation errors to grow. A good compromise is to first rearrange the demand periods in decreasing order of demand magnitude. In this way you obtain a *cumulative load duration curve*, as in Figure 16.3, where the slope of the curve is nearly constant. Reducing the number of time steps at this point will result in a reduced approximation error in comparison to the situation without the rearrangement (see Figure 16.4). Demand can now be realistically modeled with just two rather than twelve demand periods. In the sequel, the two corresponding demand categories will be referred to as *base load* and *peak load*.

Two instead of twelve periods

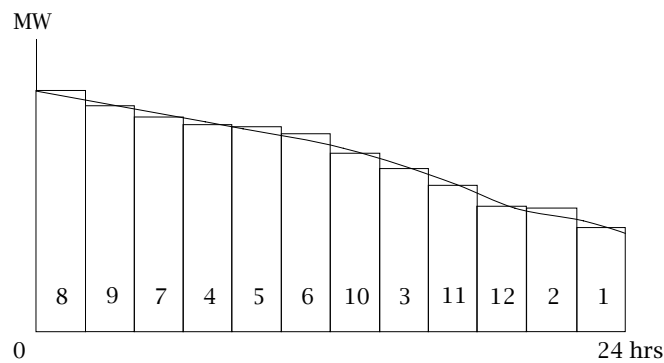


Figure 16.3: Cumulative load duration curve

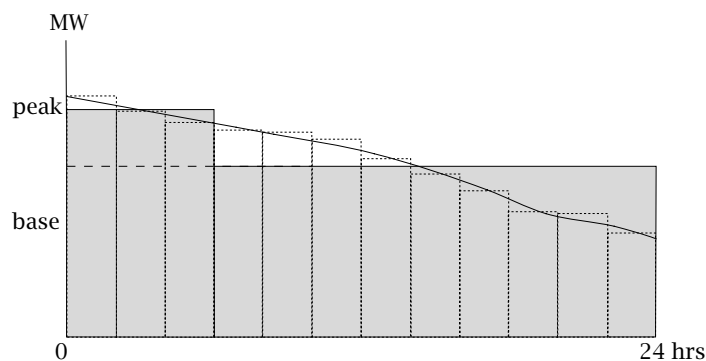


Figure 16.4: Approximating cumulative load duration curve

16.3 A deterministic expansion model

In this section a deterministic version of the power system expansion problem is presented. This model forms the basis for subsequent sections that discuss stochastic approaches to deal with the uncertain demand.

This section

The main decision to be made is the new design capacity (in [GW]) of each type of power plant. It is assumed that all levels of design capacity can be built, and that there are no decisions to be made about either the number of power plants or their size. Once capacity has been decided, it can be allocated to satisfy demand. This allocation decision can be supplemented with the decision to import electricity when available capacity is insufficient. The underlying objective is to minimize total daily cost consisting of (a fraction of) capital cost to build new design capacity, operating cost associated with the allocation decision, and the cost of importing electricity.

Decisions to be made

In the previous section the distinction has been made between base demand and peak demand. These two categories represent instantaneous demand in GW. The duration of base demand is 24 hours, while the duration of peak demand is 6 hours. On the basis of instantaneous demand and its duration you can compute the electricity requirement per category in GWh as follows. The base electricity demand is equal to base demand multiplied by 24 [h]. The peak electricity demand is the difference between peak demand and base demand multiplied by the proper duration expressed in hours. The formula is provided at the end of the 'Notation' paragraph stated below.

Deterministic daily demand requirements

Due to physical plant restrictions, nuclear power can only be used to satisfy base demand requirements. In AIMMS such a restriction can be enforced by using domain restrictions on the allocation variables introduced in the model below.

Restricting nuclear usage

The objective and the constraints that make up the simplified deterministic power expansion model are first summarized in the following qualitative model formulation.

Verbal model statement

Minimize: *total capital, import and operating costs,*

Subject to:

- *for all plant types: allocated capacity must be less than or equal to existing design capacity plus new design capacity, and*
- *for all modes: utilized capacity plus import must equal electricity demand.*

The following symbols will be used.

Notation

Indices:

p	<i>plant types</i>
k	<i>demand categories</i>

Parameters:

e_p	<i>existing capacity of plant type p [GW]</i>
cc_p	<i>daily fraction of capital cost of plant p [10^3 \$/GW]</i>
oc_p	<i>daily operating cost of plant p [10^3 \$/GWh]</i>
ic_k	<i>electricity import cost for category k [10^3 \$/GWh]</i>
d_k	<i>instantaneous electricity demand for category k [GW]</i>
du_k	<i>duration of demand for category k [h]</i>
r_k	<i>required electricity for category k [GWh]</i>

Variables:

x_p	<i>new design capacity of plant type p [GW]</i>
y_{pk}	<i>allocation of capacity to demand [GW]</i>
z_k	<i>import of electricity for category k [GWh]</i>

where the parameter r_k is defined as $r_k = (d_k - d_{k-1})du_k$.

The mathematical description of the model can be stated as follows.

*Mathematical
model
statement*

Minimize:

$$\sum_p cc_p(e_p + x_p) + \sum_k \left(ic_k z_k + du_k \sum_p oc_p y_{pk} \right)$$

Subject to:

$$\begin{aligned} \sum_k y_{pk} &\leq e_p + x_p && \forall p \\ z_k + du_k \sum_p y_{pk} &= r_k && \forall k \\ x_p &\geq 0 && \forall p \\ y_{pk} &\geq 0 && \forall (p, k) \\ \mathcal{Y}^{\text{nuclear, peak}} &= 0 \\ z_k &\geq 0 \end{aligned}$$

16.4 What-if approach

This section presents a first approach to deal with the uncertainty in electricity demand. In essence, you consider a set of demand figures, called *scenarios* and observe the proposed design capacity for each scenario. The underlying motivation is to discover manually, through trial and error, those design capacity values that seem to be a good choice when taking into account all possible scenarios.

This section

Consider a set of four scenarios reflecting different weather and economic conditions. Estimated figures for both base and peak load per scenario are presented in Table 16.1.

Model data

Demand [GW]	Scenario 1	Scenario 2	Scenario 3	Scenario 4
base load	8.25	10.00	7.50	9.00
peak load	10.75	12.00	10.00	10.50

Table 16.1: Estimated demand figures for different scenarios

Initial existing capacity, together with capital and operating cost figures for each plant type are summarized in Table 16.2. In addition, the cost of importing electricity is set at 200 [10^3 \$/GWh] for each demand category, and, as stated previously, the duration of the base period is 24 hours while the duration of the peak period is 6 hours.

Plant Type p	e_p [GW]	cc_p [10^3 \$/GW]	oc_p [10^3 \$/GWh]
coal	1.75	200	30.0
hydro	2.00	500	10.0
nuclear		300	20.0

Table 16.2: Existing capacity and cost figures per plant type

A set of design capacities will be referred to as a *plan*. A plan can be determined by computing the optimal design capacities for a particular demand scenario. By repeating the computation for each of the four demand scenarios, the four plans in Table 16.3 are obtained. Note that coal and hydro power plants seem to be the most attractive options in all four plans. No nuclear capacity is installed.

Constructing plans from scenarios

Eventually, only a single plan can be implemented in reality. An obvious first choice seems to be the cheapest plan from the table above, namely, plan III. However, what happens when plan III is implemented and a scenario other than scenario 3 occurs? The answer to this question is illustrated in Table 16.4. You can see that total cost increases dramatically for the other scenarios. This increase is due to the relatively expensive cost to import electricity to meet the increased demand. If all scenarios are equally likely, then the average cost is 8,116.25 [10^3].

Selecting the cheapest plan

Plan	Based on	Total Capacity [GW]			Total Cost [10^3 \$]
		coal	hydro	nuclear	
I	scenario 1	2.50	8.25		7055.0
II	scenario 2	2.00	10.00		8160.0
III	scenario 3	2.50	7.50		6500.0
IV	scenario 4	1.75	8.75		7275.0

Table 16.3: Optimal design capacities for individual scenarios only

As indicated in the previous paragraph, the cheapest plan may not be the best plan. That is why it is worthwhile to look at all other plans, to see how they perform under the other scenarios. An overview is provided in Table 16.5. In this comparison, plan III turns out to be the worst in terms of average cost, while plan I scores the best.

Comparing all plans

Each plan was produced on the basis of one of the selected demand scenarios. You can always dream up another (artificial) scenario just to produce yet another plan. Such a plan could then be evaluated in terms of the average cost over the four scenarios, and perhaps turn out to be a better plan. For instance, an obvious choice of artificial demand figures is the average demand over the four scenarios. Solving the model for these demand figures results in a plan with an average cost of 7685.35 [10^3 \$], slightly better than plan I. Even though what-if analysis has provided some insight into the selection of a better plan, there is a clear need for an approach that provides the best plan without having to search over artificial scenarios. Such an approach is explained in the next section.

Is there a better plan?

16.5 Stochastic programming approach

This section presents a second and more sophisticated approach to deal with uncertainty in electricity demand when compared to the what-if approach of the previous section. Essentially, all demand scenarios are included in the model simultaneously, and their average cost is minimized. In this way, the model solution presents those particular design capacity values that seem to be a good choice in the light of all scenarios weighted by their probability. The model results of the worked example are compared to the solutions presented in the previous section.

This section

In the mathematical model of this chapter, the electricity demand covers a single time period. Prior to this period, the exact demand values are not known, but a capacity design decision must be made. This is referred to as the *first stage*. As the realization of electricity demand becomes known, the allocation of the already decided design capacity is made, together with the decision of

Two-stage model

	Capacity Imported [GW]	Total Cost [10^3 \$]
scenario 1	0.75	7805.0
scenario 2	2.00	10250.0
scenario 3		6500.0
scenario 4	0.50	7910.0

Table 16.4: The consequences of plan III for all scenarios

how much to import. This is the *second stage*. Using the terminology introduced in Section 16.1, the capacity design decision variables form the *control variables*, while the allocation and import decision variables form the *state variables*.

Both the allocation and import decision variables form the state variables, which will assume separate values for each scenario. This implies that there are also separate values associated with total operating cost and total import cost. It does not make sense to add all these total cost values in the cost minimization objective function. A weighted average over scenarios, however, seems a better choice, because the objective function then reflects the expected daily cost of the entire power supply system over a planning period. The weights are then the probabilities associated with the scenarios.

Expected cost

The following symbols will be used. Note that most of them are similar to the ones already presented in the previous section, except that a scenario index has been added. As to be expected, the first stage (control) variable x does not have a scenario index, while all second stage (state) variables are naturally linked to a scenario.

Notation

Indices:

p	plant types
k	demand categories
s	scenarios

Parameters:

e_p	existing capacity of plant type p [GW]
cc_p	annualized capital cost of plant p [10^3 \$/GW]
oc_p	operation cost of plant p [10^3 \$/GWh]
ic_k	electricity import cost for category k [10^3 \$/GWh]
d_{ks}	instantaneous electricity demand for k and s [GW]
du_{ks}	duration of demand for category k and scenario s [h]
r_{ks}	required electricity for k and s [GWh]
pr_s	probability of scenario s [-]

Plan	Total Cost [10^3 \$] per Scenario				Expected Cost [10^3 \$]
	1	2	3	4	
I	7055.0	9500.0	6785.0	7415.0	7688.8
II	7620.0	8160.0	7350.0	7710.0	7710.0
III	7805.0	10250.0	6500.0	7910.0	8116.3
IV	7350.0	9615.0	6825.0	7275.0	7766.3

Table 16.5: Plans compared in terms of cost

Variables:

x_p	<i>new design capacity of plant type p [GW]</i>
y_{pks}	<i>allocation of capacity to each demand realization [GW]</i>
z_{ks}	<i>electricity import for scenario s and category k [GW]</i>
v_s	<i>total import and operating cost for scenario s [10^3 \$]</i>

where the parameter r_{ks} is defined as $r_{ks} = (d_{ks} - d_{k-1,s})du_{ks}$.

The mathematical description of the stochastic model below resembles the model description in the previous section. The main difference is the formulation of the objective function. The capital cost determination associated with existing and new design capacity remains unchanged. All other cost components are scenario-dependent, and a separate definition variable v_s is introduced to express the expected operating and importing cost in a convenient manner.

Mathematical model statement

Minimize:

$$\sum_p cc_p(e_p + x_p) + \sum_s pr_s v_s$$

Subject to:

$$\begin{aligned} \sum_k y_{pks} &\leq e_p + x_p && \forall (p, s) \\ z_{ks} + du_{ks} \sum_p y_{pks} &= r_{ks} && \forall (k, s) \\ \sum_k \left(ic_k z_{ks} + du_{ks} \sum_p oc_p y_{pks} \right) &= v_s && \forall s \\ x_p &\geq 0 && \forall p \\ y_{pks} &\geq 0 && \forall (p, k, s) \\ y_{\text{nuclear, peak}, s} &= 0 && \forall s \\ z_{ks} &\geq 0 && \forall s \\ v_s &\geq 0 && \forall s \end{aligned}$$

Plan	Total Capacity [GW]			Expected
	coal	hydro	nuclear	Cost [10^3 \$]
I	2.50	8.25		7688.75
II	2.00	10.00		7710.00
III	2.50	7.50		8116.25
IV	1.75	8.75		7766.25
V	3.00	8.25	0.75	7605.00

Table 16.6: Optimal design capacities and expected cost per plan

The solution of the stochastic expansion model produces an entire new plan, which will be referred to as plan V. The design capacity values (both existing and new) and the corresponding expected cost values associated with the original four plans plus plan V are listed in Table 16.6. By construction, the expected cost of plan V is the lowest of any plan.

Model results

	allocation [GW]			import [GW]	surplus capacity [GW]	deficit [GW]	total costs [10^3 \$]
	coal	hydro	nuclear				
Scenario 1 base peak	2.50	8.25			1.25		7380.0
Scenario 2 base peak	1.00 2.00	8.25	0.75				8370.0
Scenario 3 base peak	1.75	7.50 0.75			2.00		7110.0
Scenario 4 base peak	1.50	8.25	0.75		1.50		7560.0

Table 16.7: Optimal allocations for individual scenarios

In Plan V nuclear power is selected as a source of electricity. The reason is that nuclear power plants have lower capital costs than hydro-electric power plants. This fact helps to keep the costs down in scenarios other than the most restrictive one, namely Scenario 2. The optimal allocations corresponding to Plan V are given in Table 16.7.

*Nuclear power
plant selected*

16.6 Summary

In this chapter, two methods for dealing with uncertainty were illustrated using a power plant capacity expansion example. Both methods were based on

the use of scenarios designed to capture the uncertainty in data. The first method is referred to as *what-if analysis*, and essentially computes the consequences for each individual scenario by solving a sequence of deterministic models. The second method is referred to as *stochastic programming*, and considers all scenarios simultaneously while minimizing expected cost. The resulting model increases in size in proportion to the number of scenarios. Based on the example, the solution of the stochastic programming formulation was shown to be superior to any of the solutions derived from the manual what-if analysis.

Exercises

- 16.1 Implement the deterministic formulation of the Power System Expansion model described in Section 16.3, and perform the what-if experiments described in Section 16.4 using AIMMS.
- 16.2 Implement the stochastic formulation of the Power System Expansion model described in Section 16.5, and compare the optimal solution with the solutions of the what-if experiments.
- 16.3 Set up an experiment in AIMMS to investigate the sensitivity of the optimal stochastic programming solution to changes in the initial equal scenario probabilities of .25.

Chapter 17

An Inventory Control Problem

In this chapter you will encounter a multi-period inventory control problem with uncertain demand. At the beginning of each period the volume of production is decided prior to knowing the exact level of demand. During each period the demand becomes known, and as a result, the actual inventory can be determined. The objective in this problem is to minimize overall expected cost. The problem can be translated into a stochastic multi-stage optimization model. Such a model is a nontrivial extension of the two-stage model discussed in Chapter 16, and will be examined in detail. An alternative statement of the objective function is developed, and an instance of the stochastic inventory model is provided for illustrative purposes.

This chapter

As already stated in Chapter 16, there is a vast literature on stochastic programming, but most of it is only accessible to mathematically skilled readers. Two selected book references on stochastic programming are [In94] and [Ka94].

References

Linear Program, Stochastic Program, Multi-Stage, Control-State Variables, Mathematical Derivation, Worked Example.

Keywords

17.1 Introduction to multi-stage concepts

In this section the extension of two-stage stochastic programming to multi-stage programming is discussed. Tree-based terminology is used to characterize the underlying model structure.

This section

Stochastic models contain so-called event parameters that do not have a priori fixed values. An event is an act of nature that falls outside the control of a decision maker. A typical example of an event parameter is product demand inside a decision model that determines production levels. The exact demand values are not known when the production decision has to be made, but they become known afterwards. Prior to the production decision, it only makes sense to consider various data realizations of the event parameter demand, and somehow take these into account when making the production decision.

*Event
parameters*

A stochastic model contains two types of variables, namely control and state variables. *Control variables* refer to all those decision variables that must be decided at the beginning of a period prior to the outcome of the uncertain event parameters. *State variables*, on the other hand, refer to all those decision variables that are to be determined at the end of a period after the outcome of the uncertain event parameters are known.

Control and state variables

The term *two-stage* decision making is reserved for the sequence of control decisions (first stage), event realizations, and state determination (second stage) for a single time period. The term *multi-stage* decision making refers to sequential two-stage decision making, and is illustrated in Figure 17.1.

Two-stage versus multi-stage

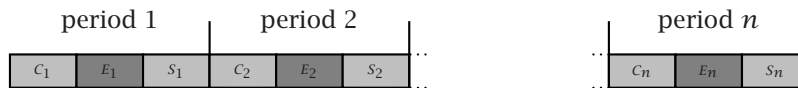


Figure 17.1: Multi-stage decision making

The number of possible data realizations of event parameters is usually very large. In theory, it is customary to relate the event parameters to continuous random variables with infinitely many outcomes. However, in practical applications it turns out better from a computational point of view to assume not only a finite number of data realizations, but also relatively few of them. This requires careful data modeling, so that the approximation with relatively few data realizations is still useful for decision making.

Assume few realizations

Whenever the underlying model is a multi-period model, the data realizations of event parameters can be expressed most conveniently in the form of a *tree*. An example is provided in Figure 17.2. Each level in the tree corresponds to a time slice, and each arc represents a particular realization of the event parameters. A node in the tree refers to a state of the system. The label associated with each arc is the *event description*. The fraction associated with each arc is the corresponding *event probability*. Note that the sum over all probabilities emanating from a single state equals one, reflecting the fact that all event parameter realizations are incorporated in the tree.

Tree with event probabilities

The event probabilities mentioned in the previous paragraph can be viewed as conditional probabilities. They describe the probability that a particular event realization will take place *given* the state from which this event emanates. In practical applications, these conditional probabilities form the most natural input to describe the occurrence of events.

Conditional probabilities ...

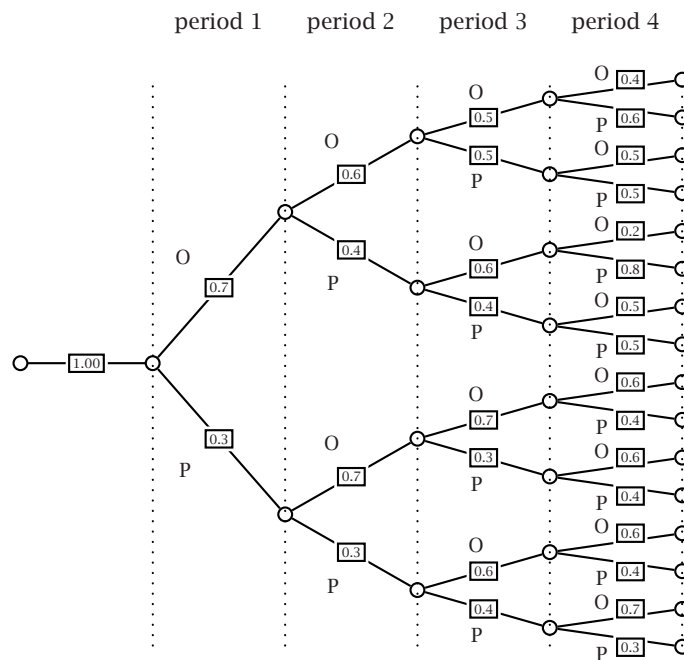


Figure 17.2: Tree with event labels and conditional event probabilities

By multiplying the event probabilities along the path from the root (the initial state) until a particular state, the unconditional probability to reach that particular state is computed. The tree with unconditional state probabilities is illustrated in Figure 17.3. Note that for all events in a single period, the sum over all unconditional probabilities add up to one. These probabilities will be used in the sequel to weight the outcome at each state to compute the overall expected outcome.

... versus
unconditional
probabilities

A *scenario* in the multi-stage programming framework is the collection of all event data instances along a path from the root to a particular leaf node in the event tree. Thus, the number of leaf nodes determines the number of scenarios. Note that the concepts of scenario and event coincide when there is only a single period. By definition, the probabilities associated with scenarios are the same as the unconditional probabilities associated with the leaf nodes (i.e. terminal states).

Scenarios

You may have noticed that there are two related terminologies that mingle naturally in the description of multi-stage stochastic programming models. One characterizes the concepts typically used in the stochastic programming literature, while the other one characterizes these same concepts in terms of tree structures. The tree terminology enables you to visualize concepts from stochastic programming in a convenient manner, and will be used throughout

Two related
terminologies

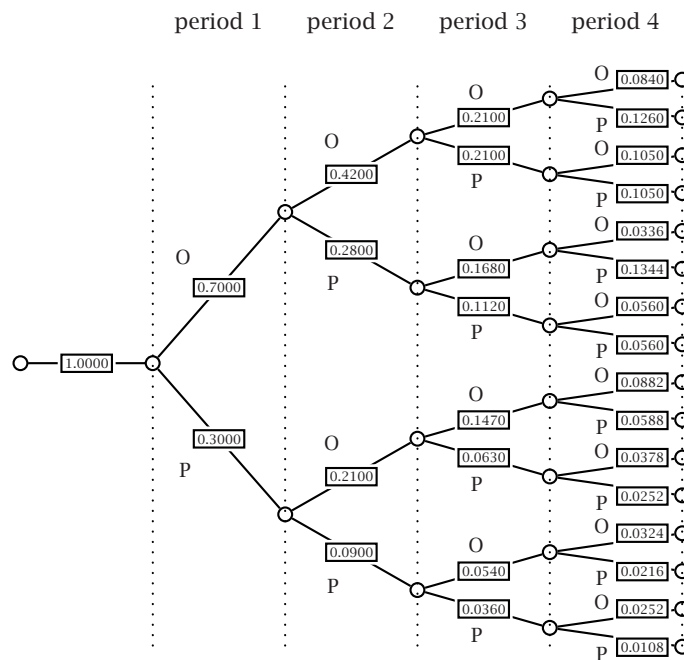


Figure 17.3: Tree with unconditional state probabilities

the chapter. The two related terminologies are summarized in Table 17.1.

stochastic terminology	tree-based terminology
event	arc
state	node
initial state	root node
final state	leaf node
scenario	path (root-to-leaf)

Table 17.1: Two related terminologies

17.2 An inventory control problem

In this section you will encounter a simplified example in which the volume of beer to be bottled is decided while minimizing bottling, inventory and external supply costs. This example is used to illustrate the multi-stage approach to stochastic modeling discussed in the previous section.

This section

Consider the decision of how much beer to bottle during a particular week. There are different beer types, and the available bottling machines can be used for all types. There is an overall bottling capacity limitation. Beer bottled in a particular week is available to satisfy demand of subsequent weeks. Bottling cost and storage cost are proportional to the amount of beer that is either bottled or stored. There is a minimum amount of storage required at the end of the last period.

Bottling of beer

Demand is assumed to be uncertain due to major fluctuations in the weather. The decision variable of how much beer to bottle in a particular week is taken prior to knowing the demand to be satisfied. Therefore, the decision variable is a control variable, and demand is the event parameter. Once weekly demand becomes known, the inventory can be determined. Therefore, the inventory variable is a state variable. The term ‘decide now and learn later’ is frequently used in the literature, and reflects the role of the control variables with respect to the event parameters.

Decide now and learn later

The uncertain demand over time can be characterized in terms of scenarios. For the sake of simplicity, assume that there are only a few of them. There will be pessimistic and optimistic events emanating from each state, and their conditional probabilities are known. All input data and model results are provided in Section 17.5.

Demand scenarios

17.3 A multi-stage programming model

In this section a multi-stage programming model formulation of the inventory control problem of the previous section is developed in a step-by-step fashion, and the entire model summary is presented at the end.

This section

The model identifiers need to be linked to the scenario tree in some uniform fashion. By definition, all variables are linked to nodes: state variables are defined for every node, while control variables are defined for emanating nodes. It is therefore natural to index variables with a state index. Event parameters, however, are linked to arcs and not to nodes. In order to obtain a uniform notation for all identifiers, event parameters will also be indexed with a state index in exactly the same way as state variables. This is a natural choice, as all arcs correspond to a reachable state.

Notation based on states

The decision to bottle beer of various types during a particular time period is restricted by the overall available bottling capacity c during that period expressed in [hl]. Let b denote beer types to be bottled, and s the states to be considered. In addition, let x_s^b denote the amount of beer of type b to be bottled at emanating state s in [hl]. Note that this decision is only defined for emanating states and thus not for terminating states. To enforce this re-

Bottling capacity constraint

striction, consider the element parameter α_s , which refers to the previous (or emanating) state of state s . When s refers to the initial state, the value of α_s is empty. For nonempty elements α_s the bottling capacity constraint can then be written as follows.

$$\sum_b x_{\alpha_s}^b \leq c \quad \forall \alpha_s$$

It is straightforward to implement element parameters, such as α_s , in the AIMMS language.

The inventory of each beer type at a particular reachable state, with the exception of the already known inventory at the initial state, is equal to the inventory at the emanating state plus the amount to be bottled decided at the emanating state plus externally supplied beer pertaining to the reachable state minus the demand pertaining to that reachable state. Note that externally supplied beer is used immediately to satisfy current demand, and will not exceed the demand due to cost minimization. Let y_s^b denote the amount of beer of type b that is stored at state s in [hl], and let z_s^b denote the external supply of beer of type b at state s in [hl]. Then, using d_s^b to denote the demand of beer of type b in state s in [hl], the inventory determination constraint can be written as follows.

*Inventory
determination
constraint*

$$y_s^b = y_{\alpha_s}^b + x_{\alpha_s}^b + z_s^b - d_s^b \quad \forall (s, b) \mid \alpha_s$$

Assume that the space taken up by the different types of bottled beer is proportional to the amount of [hl] bottled, and that total inventory space is limited. Let \bar{y} denote the maximum inventory of bottled beer expressed in [hl]. Then the inventory capacity constraint can be written as follows.

*Inventory
capacity
constraint*

$$\sum_b y_s^b \leq \bar{y} \quad \forall s$$

In the previous inventory determination constraint there is nothing to prevent currently bottled beer to be used to satisfy current demand. However, as indicated in the problem description, the amount bottled in a particular period is only available for use during subsequent periods. That is why an extra constraint is needed to make sure that at each reachable state, inventory of each type of beer at the corresponding emanating state, plus the external supply of beer pertaining to the reachable state, is greater than or equal to the corresponding demand of beer. This constraint can be written as follows.

*Demand
requirement
constraint*

$$y_{\alpha_s}^b + z_s^b \geq d_s^b \quad \forall (s, b)$$

During each period decisions are made to contribute to overall profit. For each state, the contribution to profit is determined by considering sales revenues attached to this state minus the sum of all costs associated with this state. These costs cover the corresponding bottling cost, the external supply cost, and the inventory cost. Let ps^b denote the selling price of beer of type b in [\$/hl], and let cp^b , ci^b , and ce^b denote the variable cost coefficients in [\$/hl] associated with bottling, inventory and external supply. The entire state-specific contribution to profit, denoted by v_s , can now be written as follows.

*Profit
determination
constraint*

$$v_s = \sum_b ps^b d_s^b - \sum_b (cp^b x_{\alpha_s}^b + ci^b y_s^b + ce^b z_s^b) \quad \forall s$$

In the presence of uncertain demand it does not make sense to talk about a deterministic overall profit determination. The approach in this section is to sum all state-specific profit contributions weighted with their unconditional probability of occurrence. It turns out that this computation expresses the expected profit level for the entire planning horizon, the length of which is equal to the number of time periods covered by the event tree. Such profit level can then be written as follows.

*Objective
function*

$$\sum_s p_s v_s$$

In the next section it is shown that the above expression coincides with the expected profit over all scenarios, where scenario-specific contributions to profit are weighted with their unconditional probability of occurrence. The number of terms in this alternative formulation of the objective function is then equal to the number of terminal states (i.e. the number of leaf nodes in the event tree). The two main ingredients are the scenario probabilities and the scenario profits. The scenario probabilities are the unconditional probabilities associated with the leaf nodes, and add up to one. The profit contribution per scenario is obtained by summing all corresponding state-specific profit contributions.

*Alternative
formulation*

The above objective and the constraints that make up the stochastic programming formulation of the simplified inventory control problem can now be summarized through the following qualitative model formulation.

*Verbal model
summary*

Maximize: total expected net profit,

Subject to:

- for all emanating states: total bottling volume must be less than or equal to overall bottling capacity,
- for all beer types and reachable states: inventory at reachable state is equal to inventory at emanating state plus bottling volume at emanating state plus external supply pertaining to reachable state minus demand pertaining to reachable state,

- for all reachable states: inventory of bottled beer at reachable state must be less than or equal to maximum inventory of bottled beer,
- for all beer types and reachable states: inventory at emanating state plus external supply pertaining to reachable state must be greater than or equal to demand pertaining to reachable state, and
- for all reachable states: total net profit is sales revenue minus total costs consisting of bottling, inventory and external supply costs.

The following symbols have been introduced to describe the objective function and the constraints. Notation

Indices:

b	beer types
s	states

Parameters:

ps^b	selling price of beer type b [\$/hl]
cp^b	production cost of bottling beer type b [\$/hl]
ci^b	inventory cost of storing beer type b [\$/hl]
ce^b	external supply cost of beer type b [\$/hl]
c	overall capacity to bottle beer [hl]
\bar{y}	maximum inventory of bottled beer [hl]
d_s^b	demand of beer type b in state s [hl]
p_s	probability of reaching state s [-]
α_s	previous state of state s in event tree

Variables:

x_s^b	beer type b bottled at emanating state s [hl]
y_s^b	beer type b stored at state s [hl]
z_s^b	external supply of beer b at state s [hl]
v_s	state-specific profit contribution at state s [\$]

The mathematical description of the model can now be summarized as follows.

*Mathematical
model summary*

Maximize:

$$\sum_s p_s v_s$$

Subject to:

$$\begin{aligned}
 \sum_b x_{\alpha_s}^b &\leq c && \forall \alpha_s \\
 y_{\alpha_s}^b + x_{\alpha_s}^b + z_s^b - d_s^b &= y_s^b && \forall (s, b) \mid \alpha_s \\
 \sum_b y_s^b &\leq \bar{y} && \forall s \\
 y_{\alpha_s}^b + z_s^b &\geq d_s^b && \forall (s, b) \\
 \sum_b p_s^b d_s^b - \sum_b (cp^b x_{\alpha_s}^b + ci^b y_s^b + ce^b z_s^b) &= v_s && \forall s \\
 x_s^b &\geq 0 && \forall (b, s) \\
 y_s^b &\geq 0 && \forall (b, s) \\
 z_s^b &\geq 0 && \forall (b, s)
 \end{aligned}$$

17.4 Equivalent alternative objective function

In this section you will find a proof of the fact that the expected profit function used in the previous section can be expressed in an alternative but equivalent form.

This section

Consider the following identifiers defined over the set of states.

Recursive profits and probabilities

π_s	<i>conditional probability of event prior to state s</i>
p_s	<i>unconditional probability of reaching state s</i>
v_s	<i>state-specific profit contribution at state s [\$]</i>
w_s	<i>cumulative profit contribution at state s [\$]</i>

Then, the recursive relationships between these identifiers are defined for each node in the tree (starting at the root), and will be used in the sequel.

$$\begin{aligned}
 p_s &= \pi_s p_{\alpha_s} \\
 w_s &= v_s + w_{\alpha_s}
 \end{aligned}$$

The recursive unconditional probabilities emanate from the initial state probability, which is equal to one. The recursive cumulative profit levels emanate from the initial state profit level, which is assumed to be zero. These recursive relationships can be implemented inside AIMMS using a FOR statement inside a procedure.

Initialization

Let $l = \{0, 1, \dots\}$ denote a level in the event tree, and let $L(l)$ denote the set of all nodes corresponding to this level. In addition, let \hat{l} denote the last level of the tree. Then the objective function expressing expected profit can be written in two seemingly different but equivalent forms.

Alternative formulation

$$\sum_s p_s v_s = \sum_{s \in L(\hat{l})} p_s w_s$$

This equality turns out to be a special instance of the following theorem.

Let the symbol \bar{l} denote any particular level of the event tree. Then the following equality holds for each \bar{l} . *Theorem*

$$\sum_{l=0}^{\bar{l}} \sum_{s \in L(l)} p_s v_s = \sum_{s \in L(\bar{l})} p_s w_s$$

Note that when \bar{l} is equal to \hat{l} , then the term on the left of the equal sign is nothing else but $\sum_s p_s v_s$, and the alternative formulation follows directly from the theorem. *Corollary*

The proof will be carried out by induction on the number of levels \bar{l} . For $\bar{l} = 0$, the theorem holds trivially. Consider any particular $\bar{l} > 0$, and assume that the theorem holds for $\bar{l} - 1$. Then to prove that the theorem also holds for \bar{l} , you need to rewrite summations, use the above recursive definitions of p_s and w_s , use the fact that $\sum_{k | \alpha_k = s} \pi_k = 1$, and, of course, use the statement of the theorem for $\bar{l} - 1$ during the last step. All this is done in the following statements. *Proof*

$$\begin{aligned} \sum_{s \in L(\bar{l})} p_s w_s &= \sum_{s \in L(\bar{l})} p_s (v_s + w_{\alpha_s}) \\ &= \sum_{s \in L(\bar{l})} p_s v_s + \sum_{s \in L(\bar{l})} p_s w_{\alpha_s} \\ &= \sum_{s \in L(\bar{l})} p_s v_s + \sum_{s \in L(\bar{l}-1)} \sum_{k | \alpha_k = s} \pi_k p_{\alpha_k} w_{\alpha_k} \\ &= \sum_{s \in L(\bar{l})} p_s v_s + \sum_{s \in L(\bar{l}-1)} p_s w_s \sum_{k | \alpha_k = s} \pi_k \\ &= \sum_{s \in L(\bar{l})} p_s v_s + \sum_{s \in L(\bar{l}-1)} p_s w_s \\ &= \sum_{s \in L(\bar{l})} p_s v_s + \sum_{l=0}^{\bar{l}-1} \sum_{s \in L(l)} p_s v_s \\ &= \sum_{l=0}^{\bar{l}} \sum_{s \in L(l)} p_s v_s \end{aligned}$$

□

17.5 A worked example

In this section an input data set is provided together with an overview of the results based on the multi-stage programming model developed in Section 17.3. The initial probabilities are the same as in Figures 17.2 and 17.3. The revenues

This section

and cost values are presented in Table 17.2 while the demand requirements for both beer types are listed in Figure 17.4.

b	ps^b [\$/hl]	cp^b [\$/hl]	ci^b [\$/hl]	ce^b [\$/hl]
light	300	12.0	5.0	195.0
regular	400	10.0	5.0	200.0

Table 17.2: Revenues and cost values

The overall bottling capacity c is 46.0 [hl] and the maximum inventory of bottled beer \bar{y} is 52.0 [hl]. Initial stock for light beer is 17 [hl], and initial stock for regular beer is 35 [hl].

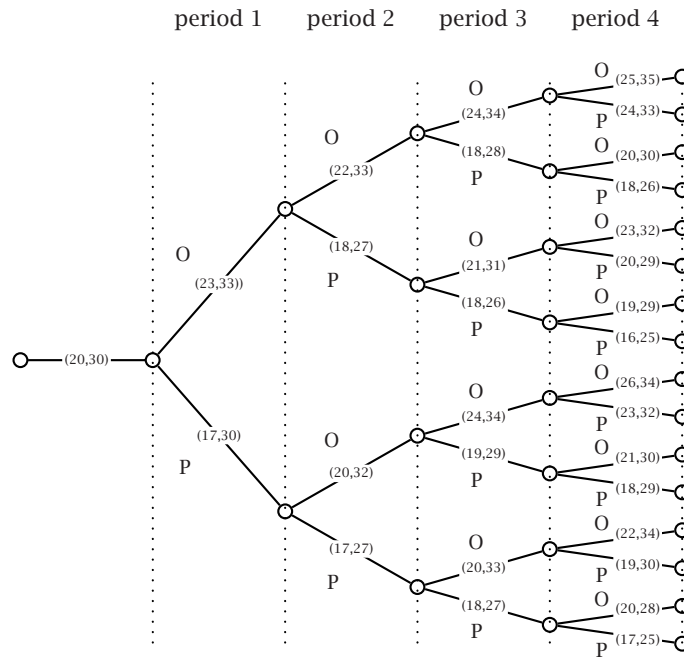


Figure 17.4: Demand requirements for both beer types (light,regular)

The optimal solution of the multi-stage programming model is presented in Table 17.3 with a total weighted profit of 76482.4 [\$]. Notice that the hierarchical structure of the scenarios is not only reflected in the order and description of their labels, but also in the zero-nonzero pattern of the control and state variables. Even though optimal decisions are determined for each period and all possible scenarios, in most practical applications only the initial decisions are implemented. The model is usually solved again in each subsequent period after new scenario information has become available.

*Optimal
solution*

Once you have implemented the model yourself, you may want to verify that the production capacity constraint is binding and has a positive shadow price for scenarios 'I', 'O', 'OO', 'OP' and 'PO'. Similarly, the storage capacity constraint is binding for scenarios 'PP' and 'PPP'. This indicates that both types of capacity play a vital role in the optimal use of the brewery.

*Binding
constraints*

s	b	light				regular				v_s
		x_s^b	y_s^b	z_s^b	d_s^b	x_s^b	y_s^b	z_s^b	d_s^b	
I		18.0	17.0	20.0	20.0	28.0	35.0	30.0	30.0	7840.0
O		18.0	18.0	6.0	23.0	28.0	30.0		33.0	18194.0
OO		18.0	18.0	4.0	22.0	28.0	28.0	3.0	33.0	17694.0
OOO			18.0	6.0	24.0		28.0	6.0	34.0	17704.0
OOOO				7.0	25.0			7.0	35.0	18735.0
OOOP				6.0	24.0			5.0	33.0	18230.0
OOP			18.0		18.0		28.0		28.0	15874.0
OOPO				2.0	20.0			2.0	30.0	17210.0
OOPP					18.0		2.0		26.0	15790.0
OP		19.0	18.0		18.0	27.0	31.0		27.0	15459.0
OPO			19.0	3.0	21.0		27.0		31.0	17387.0
OPOO				4.0	23.0			5.0	32.0	17920.0
OPOP				1.0	20.0			2.0	29.0	17005.0
OPP			19.0		18.0		32.0		26.0	15047.0
OPPO					19.0		3.0		29.0	17285.0
OPPP			3.0		16.0		7.0		25.0	14750.0
P		18.0	18.0		17.0	27.0	33.0		30.0	16349.0
PO		17.0	18.0	2.0	20.0	29.0	28.0		32.0	17694.0
POO			17.0	6.0	24.0		29.0	6.0	34.0	17706.0
POOO				9.0	26.0			5.0	34.0	18645.0
POOP				6.0	23.0			3.0	32.0	17930.0
POP			17.0	1.0	19.0		29.0	1.0	29.0	16181.0
POPO				4.0	21.0			1.0	30.0	17320.0
POPP				1.0	18.0				29.0	16805.0
PP		19.0	19.0		17.0	26.0	33.0		27.0	15154.0
PPO			19.0	1.0	20.0		26.0		33.0	18292.0
PPOO				3.0	22.0			8.0	34.0	18015.0
PPOP					19.0			4.0	30.0	16900.0
PPP			20.0		18.0		32.0		27.0	15452.0
PPPO					20.0		4.0		28.0	17180.0
PPPP			3.0		17.0		7.0		25.0	15050.0

Table 17.3: Optimal solution

17.6 Summary

In this chapter a multi-stage stochastic programming model was viewed as a sequence of two-stage stochastic programming models. A tree-based terminology was used to describe event probabilities and multi-stage scenarios. All concepts were illustrated on the basis of a simplified inventory control model. Two alternative and seemingly different objective functions were introduced, and were shown to be equivalent. A complete input data set was provided,

together with an overview of the model results.

Exercises

- 17.1 Implement the multi-stage mathematical program summarized at the end of Section 17.3 using the example data provided in Section 17.5. Verify that the optimal solution found with AIMMS coincides with the one presented in Table 17.3.
- 17.2 Implement the model with the alternative objective function described in Section 17.4, and verify whether the optimal solution remains the same.
- 17.3 Set up an experiment in AIMMS to investigate the sensitivity of the overall optimal stochastic programming objective function value to changes in the number of scenario's.

Part V

Advanced Optimization Modeling Applications

Chapter 18

A Portfolio Selection Problem

In this chapter you will find an extensive introduction to portfolio selection decision making. Decision making takes place at two distinct levels. At the strategic level, the total budget to be invested is divided among major investment categories. At the tactical level, the budget for a particular investment category is divided among individual securities. Both the strategic and the tactical portfolio selection problems are considered and subsequently translated into quadratic programming models using the variance of the portfolio as a measure of risk. The objective function of the relatively small strategic portfolio selection model minimizes added covariances, which are estimated outside the model. The objective function of the tactical portfolio selection model also minimizes added covariances, but their values are not explicit in the model. Instead, scenario data is used to represent covariances indirectly, thereby avoiding the explicit construction of a large matrix. The required mathematical derivations for both models are presented in separate sections. In the last part of the chapter you will find a section on one-sided variance as an improved measure of risk, a section on the introduction of logical constraints, and a section on the piecewise linear approximation of the quadratic objective function to keep the model with logical constraints within the framework of mixed-integer linear programming.

This chapter

The methodology for portfolio selection problems dates back to the work of Markowitz [Ma52] and is also discussed in [Re89].

References

Integer Program, Quadratic Program, Mathematical Derivation, Mathematical Reformulation, Logical Constraint, Piece-Wise Approximation, Worked Example.

Keywords

18.1 Introduction and background

The term investor is used for a person (or institution) who treasures a certain capital. There are many types of investors, such as private investors, institutional investors (pension funds, banks, insurance companies), governments, professional traders, speculators, etc. Each investor has a personal view of risk and reward. For instance, the point of view of a pension fund's portfo-

Investors

lio manager will differ from that of a private investor. This is due to several reasons. The pension fund manager invests for a large group of employees of a company (or group of companies) who expect to receive their pension payments in due time. In view of this obligation, the fund manager needs to be extremely risk averse. In contrast, a private investor is only responsible for his own actions and has full control over the investment policy. He will decide on the amount of risk that he is willing to consider on the basis of personal circumstances such as his age, family situation, future plans, his own peace of mind, etc.

There are many types of investment categories. Typical examples are deposits, saving accounts, bonds, stocks, real estate, commodities (gold, silver, oil, potatoes, pigs), foreign currencies, derivatives (options, futures, caps, floors), etc. Each investment category has its own risk-reward characteristics. For instance, saving accounts and bonds are examples of low risk investment categories, whereas stocks and real estate are relatively more risky. A typical example of an investment category with a high level of risk is the category of financial derivatives such as options and futures. These derivatives are frequently used for speculation, since they offer the possibility of a high rate of return in comparison to most other investment categories.

*Investment
categories*

The above statements need to be understood as general remarks concerning the average performance of the investment categories. Naturally, within each category, there are again discrepancies in risk-reward patterns. In the case of bonds, there are the so-called triple A bonds with a very small chance of default (for instance government loans of the United States of America). On the other end of the spectrum there are the high-yield or so-called junk bonds. These bonds are considered to have a relatively large chance of default. For instance, during the economic crises in South East Asia many of the government bonds issued by countries from that area are considered to be junk bonds. The interest paid on these type of bonds can amount to 30 percent. Similarly in the category of stocks there are many different types of stocks. The so-called blue chips are the stocks corresponding to the large international companies with a reliable track record ranging over a long period of time. On the other hand, stocks of relatively new and small companies with high potential but little or no profits thus far, often have a high expected return and an associated high degree of risk.

*Discrepancies
within
investment
categories*

The term security is used to denote one particular investment product within an investment category. For instance, the shares of companies like IBM or ABN AMRO are examples of securities within the category of stocks. Similarly a 2002 Oct Future contract on the Dutch Index of the Amsterdam Exchange (AEX) is an example of a security within the category of derivatives.

*Securities are
investment
products*

Investing in a portfolio requires funds and thus a budget. Funds are usually measured in some monetary unit such as dollars, yens or guilders. Using absolute quantities, such as \$-returns and \$-investments, has the drawback of currency influences and orders of magnitude. That is why the percentage rate of return (in the sequel referred to as ‘rate of return’),

Percentage rate of return is widely used

$$100 \cdot \frac{\text{new return} - \text{previous return}}{\text{previous return}}$$

is a widely accepted measure for the performance of an investment. It is dimensionless, and simplifies the weighting of one security against the other. As will be discussed later, the choice of the time step size between subsequent returns has its own effect on the particular rate of return values.

Naturally, investors would like to see high rates of return on their investments. However, holding securities is risky. The value of a security may appreciate or depreciate in the market, yielding a positive or negative return. In general, one can describe risk as the uncertainty regarding the actual rate of return on investment. Since most investors are risk averse, they are only willing to accept an additional degree of risk if the corresponding expected rate of return is relatively high.

Concept of risk

Instead of investing in one particular security, most investors will spread their funds over various investments. A collection of securities is known as a *portfolio*. The rationale for investing in a portfolio instead of a single security is that different securities perform differently over time. Losses on one security could be offset by gains on another. Hence, the construction of a portfolio enables an investor to reduce his overall risk, while maintaining a desired level of expected return. The concept of investing in a number of different securities is called *diversification*. This concept and its mathematical background was introduced by H. Markowitz in the early fifties ([Ma52]).

Portfolio diversification

Although diversification is a logical strategy to reduce the overall risk of a portfolio, there will be practical obstacles in realizing a well-diversified portfolio. For instance, the budget limitation of a small private investor will severely restrict the possibilities of portfolio diversification. This is not the case for the average pension fund manager, who manages a large amount of funds. He, on the other hand, may face other restrictions due to liquidity requirements over time by existing pension holders.

Practical limitations

The main focus in this chapter is on how to quantify the risk associated with a complete portfolio. What is needed, is a quantitative measure to reflect an investor’s notion of uncertainty with respect to performance of the portfolio. The approach presented in this chapter is based on tools from statistics, and is one that is frequently referred to in the literature. Nevertheless, it is just one of several possible approaches.

Quantification through statistical concepts

18.2 A strategic investment model

In this section a strategic portfolio selection model will be formulated. It models how top management could spread an overall budget over several investment categories. Once their budget allocation becomes available, tactical investment decisions at the next decision level must be made concerning individual securities within each investment category. Such a two-phase approach supports hierarchical decision making which is typical in large financial institutions.

This section

During the last decade there has been an enormous growth in investment possibilities. There are several logical explanations for this phenomenon. The globalization of financial markets has opened possibilities of geographical diversification. Investments in American, European or Asian stocks and bonds have completely different risk-reward patterns. The further professionalization of financial institutions has led to the introduction of all kinds of new financial products. In view of these developments, top management needs to concentrate on the global partitioning of funds into investment categories. This is referred to as the *strategic investment decision*.

The strategic investment decision

The allocation of the total budget over the various investment categories will be expressed in terms of budget fractions. These fractions need to be determined, and form the set of decision variables. Each budget fraction is associated with a particular investment category, and is defined as the amount invested in this category divided by the total budget.

Decision variables

The objective is to minimize the portfolio risk. In his paper Markowitz [Ma52] introduced the idea of using the variance of the total portfolio return as a measure of portfolio risk. His proposed risk measure has become a standard, and will also be used in this chapter.

Objective function

Each category has a known level of expected return. These levels, together with the budget fractions, determine the level of expected return for the entire portfolio. The investor will demand a minimal level of expected return for the entire portfolio. This requirement forms the main constraint in the portfolio model.

Minimal level of expected return

The overall approach is to choose budget fractions such that the expected return of the portfolio is greater than or equal to some desired target, and such that the level of risk is as small as possible. The model can be summarized as follows.

Verbal model statement

Minimize: *the total risk of the portfolio*

Subject to:

- *minimal level of desired return: the expected return of the portfolio must be larger than a given minimal desired level.*
- *definition of budget fractions: all budget fractions are nonnegative, and must add to 1.*

The following symbols will be used.

Notation

Index:

j *investment categories*

Parameters:

R_j *return of category j (random variable)*

m_j *expected value of random variable R_j*

M *desired (expected) portfolio return*

Variable:

x_j *fraction of the budget invested in category j*

The mathematical description of the model can be stated as follows.

*Mathematical
model
statement*

Minimize:

$$\text{Var}[\sum_j R_j x_j]$$

Subject to:

$$\sum_j m_j x_j \geq M$$

$$\sum_j x_j = 1$$

$$x_j \geq 0 \quad \forall j$$

Note that the objective function makes reference to random variables, and is therefore not a deterministic expression. To rewrite this expression, you need some mathematical concepts that are developed in the next section. It will be shown that the objective function is equivalent to minimizing added covariances. That is

*Deterministic
equivalent of
objective*

$$\text{Var}[\sum_j R_j x_j] = \sum_{jk} x_j \text{Cov}[R_j, R_k] x_k$$

Here, the new deterministic equivalent of the objective is a quadratic function in terms of the unknown x -variables. The coefficients $\text{Cov}[R_j, R_k]$ are known input data.

18.3 Required mathematical concepts

In this section the statistical concepts of expectation and variance are discussed, together with their role in the portfolio selection model presented in the previous section. The corresponding statistical functions are available in AIMMS.

This section

Consider for a moment the investment in one particular investment category, and define the random variable R that describes the rate of return on this investment category after one year. For simplicity, assume that R has a finite set I of values (outcomes), which are denoted by r_i with corresponding probabilities p_i for all $i \in I$ and such that $\sum_i p_i = 1$.

The random variable R

The concept of expectation (or expected value) corresponds to your intuitive notion of average. The expected value of the random variable R is denoted by $E[R]$ and is defined as follows.

Concept of expectation

$$E[R] = \sum_i r_i p_i$$

Note that whenever $r_i = c$ (i.e. constant) for all $i \in I$, then the expected value

$$E[R] = \sum_i r_i p_i = c \sum_i p_i = c$$

is also constant. The following result will be used to advantage in various derivations throughout the remainder of this chapter. Whenever f is a function of the random variable R , the expected value of the random variable $f(R)$ is equal to

$$E[f(R)] = \sum_i f(r_i) p_i$$

The concept of variance corresponds to the intuitive notion of variation around the expected value. The variance of a random variable R is denoted by $\text{Var}[R]$ and is defined as follows.

Concept of variance ...

$$\text{Var}[R] = E[(R - E[R])^2]$$

Using the result of the previous paragraph, this expression can also be written as

$$\text{Var}[R] = \sum_i (r_i - E[R])^2 p_i$$

Variance turns out to be a suitable measure of risk. The main reason is that variance, just like most other risk measures, is always nonnegative and zero variance is a reflection of no risk. Note that if R has only one possible value c , then the return is always constant and thus without risk. In that case, the expected value $E[R]$ is c and

... as measure of risk

$$\text{Var}[R] = \sum_i (R_i - c)^2 p_i = 0$$

A well-known inequality which is closely related to the concept of variance is the *Chebyshev inequality*:

Chebyshev inequality...

$$P(|R - E[R]| > \alpha\sigma) < \frac{1}{\alpha^2} \quad \forall \alpha > 0$$

In this inequality the term

$$\sigma = \sqrt{\text{Var}[R]}$$

is used, and is called the *standard deviation* of the random variable R .

The Chebyshev inequality states that the probability of an actual rate of return differing more than α times the standard deviation from its expected value, is less than 1 over α squared. For instance, the choice $\alpha = 5$ gives rise to a probability of at least 96 percent that the actual rate of return will be between $E[R] - 5\sigma$ and $E[R] + 5\sigma$. The smaller the variance, the smaller the standard deviation, and hence the smaller the distance between the upper and lower value of this confidence interval. This property of the Chebyshev inequality also supports the notion of variance as a measure of risk.

... supports variance as risk measure

By straightforward use of the definition of variance the following derivation leads to the Chebyshev inequality

Derivation of the Chebyshev inequality

$$\begin{aligned} \sigma^2 &= \text{Var}[R] = E[(R - E[R])^2] = \sum_i (r_i - E[R])^2 p_i \\ &= \sum_{i \mid |r_i - E[R]| \leq \alpha\sigma} (r_i - E[R])^2 p_i + \sum_{i \mid |r_i - E[R]| > \alpha\sigma} (r_i - E[R])^2 p_i \\ &\geq \sum_{i \mid |r_i - E[R]| > \alpha\sigma} (r_i - E[R])^2 p_i \\ &> \sum_{i \mid |r_i - E[R]| > \alpha\sigma} (\alpha\sigma)^2 p_i \\ &= \alpha^2 \sigma^2 \sum_{i \mid |r_i - E[R]| > \alpha\sigma} p_i \\ &= \alpha^2 \sigma^2 P(|R - E[R]| > \alpha\sigma) \end{aligned}$$

Thus, in summary, $\sigma^2 > \alpha^2 \sigma^2 P(|R - E[R]| > \alpha\sigma)$, which immediately leads to the Chebyshev inequality.

Suppose there are a finite number of investment categories j from which an investor can select. Let R_j be the random variable that denotes the rate of return on the j -th category with corresponding values (outcomes) r_{ij} and probabilities p_{ij} such that $\sum_i p_{ij} = 1 \forall j$. Let x_j be the fraction of the budget invested in category j . The return of the total portfolio is then equal to $\sum_j R_j x_j$, which, being a function of random variables, is also a random variable. This implies, using the previously described identity $E[f(R)] = \sum_i f(r_i) p_i$, that the expected value of the portfolio return equals

*Expected value
of portfolio
return*

$$\begin{aligned} E[\sum_j R_j x_j] &= \sum_i (\sum_j r_{ij} x_j p_{ij}) \\ &= \sum_j x_j (\sum_i r_{ij} p_{ij}) \\ &= \sum_j x_j E[R_j] \end{aligned}$$

This last expression is just the weighted sum of the expected values associated with the individual investment categories.

The variance of the portfolio return can now be determined as follows.

*Variance of
portfolio return*

$$\begin{aligned} \text{Var}[\sum_j R_j x_j] &= E[(\sum_j R_j x_j - E[\sum_j R_j x_j])^2] \\ &= E[(\sum_j R_j x_j - \sum_j x_j E[R_j])^2] \\ &= E[(\sum_j x_j (R_j - E[R_j]))^2] \\ &= E[(\sum_{jk} x_j (R_j - E[R_j]) x_k (R_k - E[R_k]))] \\ &= \sum_{jk} x_j x_k E[(R_j - E[R_j])(R_k - E[R_k])] \end{aligned}$$

Here, the term

$$E[(R_j - E[R_j])(R_k - E[R_k])]$$

is called the *covariance* of the random variables R_j and R_k , and will be denoted by $\text{Cov}[R_j, R_k]$. Note that $\text{Cov}[R_j, R_j] = \text{Var}[R_j]$ by definition.

The covariance of two random variables is a measure of the relation between above and below average values of these two random variables. When both positive and negative deviations tend to occur simultaneously, their covariance will be positive. When positive deviations of one of them tends to occur often with negative deviations of the second, their covariance will be negative. Only when positive and negative deviations occur randomly, their covariance will tend to be zero.

*Concept of
covariance*

By using the covariance terms, portfolio risk can be written as

$$\text{Var}[\sum_j R_j x_j] = \sum_{jk} x_j \text{Cov}[R_j, R_k] x_k$$

*The objective
minimizes
added
covariances*

Thus, the objective of the model can be formulated as the minimization of weighted covariances summed over all possible random pairs (R_j, R_k) .

From a mathematical point of view, the model shows that it is advisable to invest in categories with negative covariances. The logical explanation for this is that below average results of one investment category are likely to be offset by above average results of the other. Hence, the model formulation using covariances can be seen as the formalization of the intuitive concept of spreading risk by using various securities.

*Diversification
as a logical
strategy*

18.4 Properties of the strategic investment model

In this section several mathematical properties of the strategic investment model are investigated. In summary, it is shown that (a) any optimal solution of the nonlinear programming model developed in Section 18.2 is also a global optimum, (b) the risk-reward curve is nondecreasing and convex, and (c) multiple optimal portfolio returns are perfectly correlated.

This section

Any optimal solution of the presented portfolio selection model is globally optimal. This follows from the theory of optimization. The theory states that, whenever a model has linear constraints and the objective function $f(x)$ to be minimized is convex, i.e.

*Optimum is
global*

$$f(\alpha x_1 + (1 - \alpha)x_2) \leq \alpha f(x_1) + (1 - \alpha)f(x_2) \quad \forall \alpha \in [0, 1]$$

then any optimal solution of the model is also a globally optimal solution. In the portfolio selection model the objective function

$$f(x) = \sum_{jk} x_j \text{Cov}[R_j, R_k] x_k$$

is a quadratic function, and is convex if and only if the associated matrix of second-order derivatives is positive semi-definite.

Note that the matrix with second-order derivatives is precisely the covariance matrix with elements $\text{Cov}[R_j, R_k]$. Such a matrix is positive semi-definite if and only if

*Required matrix
condition is
satisfied*

$$\sum_{jk} x_j \text{Cov}[R_j, R_k] x_k \geq 0 \quad \forall x_j, x_k \in \mathbb{R}$$

This mathematical condition, however, happens to be equivalent to the inequality $\text{Var}[\sum_j R_j x_j] \geq 0$, which is true by definition.

The input parameter M characterizes the individual preference of the investor. Instead of solving the model for one particular value of M , it will be interesting for an investor to see what the changes in optimal risk value will be as the result of changes in M . The optimal risk value is unique for each value of M due to the global optimality of the solution. Thus the optimal risk value V can be considered as a function of the desired minimal level of expected return M . This defines a parametric curve $V(M)$.

Parametric risk-reward curve $V(M)$

The value $V(M)$ is defined for all values of M for which the model is solvable. It is straightforward to verify that $-\infty \leq M \leq \max_j m_j \equiv M_{\max}$. When M is set to $-\infty$, $V(M)$ obtains its lowest value. An investor will be interested in the largest feasible value of M that can be imposed such that $V(M)$ remains at its lowest level. Let M_{\min} be this level of M . Then, for all practical purposes, M can be restricted such that $M_{\min} \leq M \leq M_{\max}$. Note that the value of M_{\min} can be determined experimentally by solving the following penalized version of the portfolio selection model

Practical levels of desired return

Minimize:

$$\text{Var}[\sum_j R_j x_j] - \lambda (\sum_j m_j x_j)$$

Subject to:

$$\begin{aligned} \sum_j x_j &= 1 \\ x_j &\geq 0 \quad \forall j \end{aligned}$$

for a sufficiently small value of $\lambda > 0$.

The optimal value $V(M)$ is nondecreasing in M , because any feasible solution of the model for a particular value of M will also be a solution for smaller values of M . In addition, it will be shown in the paragraph below that $V(M)$ is convex. These two properties (nondecreasing and convex), coupled with the definition of M_{\min} from the previous paragraph, imply that $V(M)$ is strictly increasing on the interval $[M_{\min}, M_{\max}]$.

Properties of risk-reward curve $V(M)$

Let $M = \alpha M_1 + (1 - \alpha) M_2$ for $\alpha \in [0, 1]$, and let $M_{\min} \leq M_1 \leq M_2 \leq M_{\max}$. In addition, let x_M , x_1 and x_2 be the optimal solutions corresponding to M , M_1 and M_2 , respectively. Furthermore, let \mathbf{Q} denote the covariance matrix. Then, as explained further below,

Proof that $V(M)$ is convex

$$\begin{aligned} V(M) &= x_M^T \mathbf{Q} x_M \\ &\leq (\alpha x_1 + (1 - \alpha) x_2)^T \mathbf{Q} (\alpha x_1 + (1 - \alpha) x_2) \\ &\leq \alpha x_1^T \mathbf{Q} x_1 + (1 - \alpha) x_2^T \mathbf{Q} x_2 \\ &= \alpha V(M_1) + (1 - \alpha) V(M_2) \end{aligned}$$

The inequality on the second line of the proof follows from the fact that $\alpha x_1 + (1 - \alpha)x_2$ is a feasible but not necessarily optimal solution for $M = \alpha M_1 + (1 - \alpha)M_2$. The third line follows directly from the convexity of the quadratic objective function, which then establishes the desired result that

$$V(\alpha M_1 + (1 - \alpha)M_2) \leq \alpha V(M_1) + (1 - \alpha)V(M_2)$$

Thus, $V(M)$ is convex in M .

Consider a fixed value of M and two distinct optimal portfolios x_1^* and x_2^* with equal variance $V^*(M)$. Then any convex combination of these two portfolios will also be a feasible portfolio due to the linearity of the model constraints. From the convexity of the quadratic objective function the variance of each intermediate portfolio return can only be less than or equal to $V^*(M)$. However, it cannot be strictly less than $V^*(M)$, because this would contradict the optimality of $V^*(M)$. Hence the variance of the return of each intermediate portfolio is equal to $V^*(M)$ and thus also optimal.

Multiple optimal portfolios ...

As will be shown next, any two distinct optimal portfolios x_1^* and x_2^* for a fixed value of M have perfectly correlated returns. Let P_1 and P_2 be the corresponding portfolio returns, and consider the variance of the return of an intermediate portfolio. This variance can be written as a weighted sum of individual covariances as follows.

... have perfectly correlated returns

$$\begin{aligned} \text{Var}[\alpha P_1 + (1 - \alpha)P_2] &= \alpha^2 \text{Cov}[P_1, P_1] + 2\alpha(1 - \alpha) \text{Cov}[P_1, P_2] + (1 - \alpha)^2 \text{Cov}[P_2, P_2] \\ &= \alpha^2 \text{Var}[P_1] + (1 - \alpha)^2 \text{Var}[P_2] + 2\alpha(1 - \alpha) \text{Cov}[P_1, P_2] \end{aligned}$$

From the previous paragraph it is also known that

$$\text{Var}[\alpha P_1 + (1 - \alpha)P_2] = \text{Var}[P_1] = \text{Var}[P_2]$$

Therefore, by substituting the above identities, the term $\text{Cov}[P_1, P_2]$ can be determined as follows.

$$\text{Cov}[P_1, P_2] = \frac{1 - \alpha^2 - (1 - \alpha)^2}{2\alpha(1 - \alpha)} \text{Var}[P_1] = \text{Var}[P_1]$$

This implies that the correlation coefficient ρ between the portfolio returns P_1 and P_2 , defined as $\text{Cov}[P_1, P_2]/(\sqrt{\text{Var}[P_1]}\sqrt{\text{Var}[P_2]})$, is equal to 1. Hence, P_1 and P_2 are perfectly correlated.

Figure 18.1 gives an example of the feasible region and contours of the objective function for a portfolio of two securities. Notice that the feasible region is now restricted to that part of the budget line $x_1 + x_2 = 1$ for which the target return is at least achieved. It is intuitively clear that the optimal combination of securities is globally optimal, due to the shape of the contours of the objective.

Illustrating global optimality

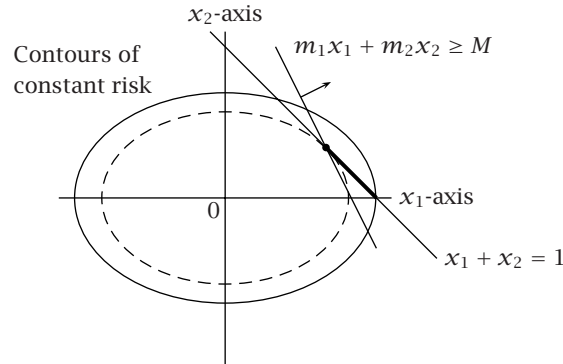


Figure 18.1: The feasible region and two objective function contours

18.5 Example of strategic investment model

In this section a small example of the strategic investment approach is presented. The required input data at this strategic level is usually not directly obtainable from public sources such as stock exchanges etc. Instead such input data is estimated from statistical data sources in a nontrivial manner, and is provided in this example without any further explanation.

This section

Consider three investment categories: stocks, bonds and real estate. The corresponding random variables will be denoted by X_1, X_2 and X_3 . The minimal level of expected return M will be set equal to 9.0. The expected return values, together with the covariances between investment categories, are displayed in Table 18.1.

Data

i	m_i j	Cov[X_i, X_j]		
		1	2	3
1	10.800	2.250	-0.120	0.450
2	7.600	-0.120	0.640	0.336
3	9.500	0.450	0.336	1.440

Table 18.1: Expected returns and covariances

After solving the portfolio model described in Section 18.2, the optimal portfolio fractions are $x_1 = 0.3233, x_2 = 0.4844, x_3 = 0.1923$. Therefore, approximately 32 percent will be invested in stocks, 48 percent in bonds and 19 percent in real estate. The corresponding optimal portfolio risk is equal to 0.5196.

The optimal solution ...

Note that the optimal portfolio risk (approximately 0.52) is even smaller than the variance of bonds (0.64), which is the lowest variance associated with any particular investment category. In addition, note that the expected portfolio return (9.00) is higher than if the entire investment had been in bonds only (namely 7.6). These results clearly illustrate the benefits of portfolio diversification.

... supports diversification

It is of interest to an investor to see what the change in optimal value V will be as a consequence of changing the value of desired return M . Below the function $V(M)$ is presented on the interval $[7.6, 10.8]$.

Risk-reward characteristic

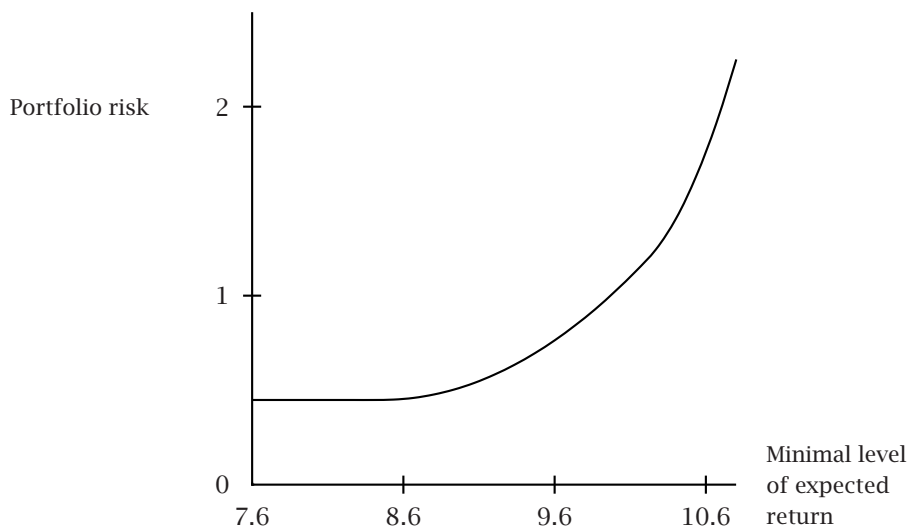


Figure 18.2: Risk-reward characteristic

The smallest level of expected return that is of interest to the investor is $M_{\min} = 8.4$, which can be derived by solving the model with the penalized objective function from the previous section. Note that this value is larger than $\min_i m_i = 7.6$. For values of M greater than M_{\min} , the curve is strictly increasing and the constraint regarding the minimal level of expected return is binding. Based on this curve an investor can make his trade-off between risk and reward.

As explained in the previous section, optimal portfolios need not be unique. In this example, however, they are unique, because there are no perfectly correlated portfolios. You may verify this observation by computing the correlation coefficients between returns on the basis of the covariances listed in Table 18.1.

Unique optimal portfolios

Similar to the parametric curve representing the optimal risk value as a function of the minimal level of expected return, you may also want to view the budget fractions of the optimal portfolio as functions of the minimal level of expected return. These parametric curves illustrate portfolio diversification, and are presented in Figure 18.3.

Portfolio diversification illustrated

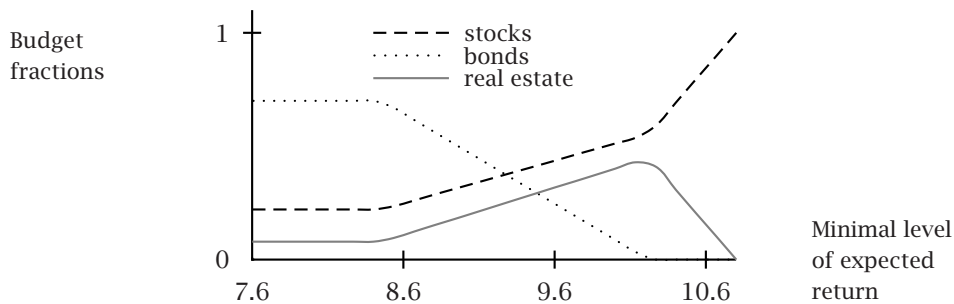


Figure 18.3: Portfolio diversification

18.6 A tactical investment model

At the tactical level, there are specialized fund managers to manage a particular investment category. Each manager receives a specific budget, which is based on the solution of the strategic investment model. In this section the tactical investment model is derived from the strategic investment model. The major difference between the two models is that the much larger covariance matrix in the tactical model is no longer modeled as an explicit matrix.

This section

The solution of the strategic investment model in the previous section suggested to invest approximately 32 percent of the overall budget in stocks. Such a result was based on aggregated data representing the entire stock investment category, and not on data regarding individual stocks. For individual stocks the question naturally arises which ones should be selected to spend the 32 percent of the total budget. This is considered to be a question at the tactical level.

From aggregated investment categories ...

The possibilities to select individual securities from within a particular investment category are enormous. If the underlying decision model at this level was the same as the strategic decision model, the corresponding covariance matrix would become very large. That is why an alternative modeling approach is proposed to deal with securities at the tactical level

... to numerous individual securities

In the paragraphs to follow, it is shown that the original portfolio variance as measure of risk can be estimated directly from real-world observed returns taken at distinct points in time. Each time observation consists of a vector of returns, where the size of the vector is equal to the number of individual securities. By considering two subsequent time periods, it is possible to compute the corresponding rate of returns. The resulting vector is referred to as a *scenario*.

*Observed
returns
determine
scenarios...*

By construction, there are as many scenarios as there are time observations minus one. It is to be expected that the time step size will vary the rate of return values associated with each scenario. The movements in returns between subsequent time observations are likely to be different when you consider hourly, daily or monthly changes in return values. The choice of time step in the computation of scenarios should be commensurable with the time unit associated with the investment decision.

*... based on
specific time
steps*

Consider a vector of random variables denoting rates of returns R_j for j . Every instance of this vector denotes a scenario. Assume that there is a finite set of scenarios. Let T denote this set with index $t \in T$. Let r_t denote a single scenario, and $p(r_t)$ its associated probability. By definition, the sum over all scenarios of the probabilities (i.e. $\sum_t p(r_t)$) equals 1.

Notation

Index:

t scenarios of size $|T|$

Parameters:

r_t vector of particular return rates for scenario t

$p(r_t)$ probability of scenario t

r_{tj} particular return rate of security j in scenario t

Note that the symbol r is overloaded in that it is used for both the vector of return rates (r_t) and the individual return rates per scenario (r_{tj}). This is done to resemble previous notation and is used throughout the remainder of this section for consistency.

Consider the following straightforward algebraic manipulations based on moving the \sum -operator, and using the properties of the E -operator.

*Reformulation
of the objective*

$$\begin{aligned}
\text{Var}[\sum_j R_j x_j] &= E[(\sum_j R_j x_j) - E[\sum_j R_j x_j]]^2] \\
&= E[(\sum_j R_j x_j) - \sum_j x_j E[R_j]]^2] \\
&= E[(\sum_j x_j (R_j - E[R_j]))^2] \\
&= \sum_t (\sum_j x_j (r_{tj} - E[R_j]))^2 p(r_t) \\
&= \sum_t p(r_t) y_t^2
\end{aligned}$$

where $y_t = \sum_j x_j (r_{tj} - E[R_j]) \quad \forall t \in T$. This results in a compact formula for the objective function in terms of the new variables y_t . These new decision variables plus their definition will be added to the model.

The repetitive calculation of the objective function and its derivatives, required by a nonlinear programming solver, can be carried out much faster in the above formulation than in the formulation of Section 18.2. This is because, in the tactical investment model, $|T|$ (the number of scenarios) is typically much smaller than $|J|$ (the number of individual securities). Therefore, the number of nonlinear terms y_t^2 is significantly smaller than the number of nonlinear terms $x_j x_k$.

Comparing the number of nonlinear terms

Let $m_j = E[R_j]$ be the expected value of security j , and let $d_{tj} = (r_{tj} - m_j)$ be the deviation from the expected value defined for each scenario. Then using the approach presented in the previous paragraph, a quadratic optimization model with x_j and y_t as the decision variables can be written as follows.

The model formulation

Minimize:

$$\sum_t p(r_t) y_t^2$$

Subject to:

$$\begin{aligned}
\sum_j d_{tj} x_j &= y_t \quad \forall t \\
\sum_j m_j x_j &\geq M \\
\sum_j x_j &= 1 \\
x_j &\geq 0 \quad \forall j
\end{aligned}$$

The properties of the above investment model are the same as the ones associated with the strategic investment model, because the above model is just an equivalent reformulation of the model presented in Section 18.2. Of course, it is still possible to derive the model properties directly from the mathematical formulation above. For instance, the verification that the new objective function is also convex, follows directly from the observation that the matrix with second-order derivatives is a $|T| \times |T|$ diagonal matrix with $2p(r_t) \geq 0$ as its diagonal elements. Such a matrix is always positive semi-definite.

*Model
properties
unchanged*

18.7 Example of tactical investment model

In this section you find a small example in terms of 5 individual stocks and 51 observations. In a realistic application, the number of observations is usually in the hundreds, while the number of candidate stocks is likely to be a multiple thereof.

This section

The stocks that can be part of the portfolio are: RD (Royal Dutch), AKZ (Akzo Nobel), KLM (Royal Dutch Airline Company), PHI (Philips) and UN (Unilever). The historical data are weekly closing values from August 1997 to August 1998, and are provided in Table 18.2. The corresponding weekly rates of return can be computed on the basis of these return values, and have all been given equal probability.

Data

As for the strategic investment model a risk-reward characteristic can be presented. The expected level of return for the various stocks is: RD -0.28, AKZ 0.33, KLM 0.40, PHI 0.30, UN 0.55. The parametric curve $V(\mu)$ is computed on the interval $[0, 0.55]$. Below both the risk-reward characteristic as well as the budget fractions of the optimal solutions are presented in Figure 18.4 and Figure 18.5.

The results

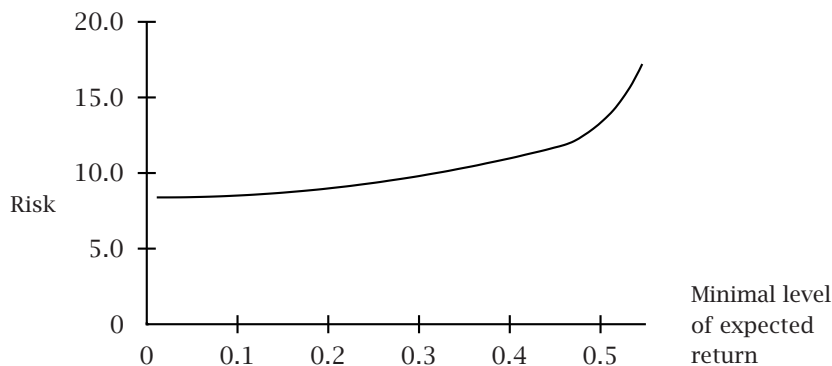


Figure 18.4: Risk-reward characteristic

Scena- rios	Valueofstocks					Scena- rios	Valueofstocks				
	RD	AKZ	KLM	PHI	UN		RD	AKZ	KLM	PHI	UN
$t-0$	111.0	82.5	70.0	154.6	110.8	$t-26$	112.8	107.0	76.3	155.9	134.2
$t-1$	108.1	81.6	73.7	152.4	108.0	$t-27$	109.7	110.4	86.0	155.0	140.9
$t-2$	107.9	80.1	72.3	146.1	103.7	$t-28$	111.7	109.7	88.9	149.9	138.2
$t-3$	108.5	83.1	69.7	157.5	106.6	$t-29$	120.4	105.9	83.5	153.5	141.6
$t-4$	111.4	85.0	69.5	168.4	107.3	$t-30$	118.0	105.9	83.4	153.0	140.6
$t-5$	115.5	92.6	74.8	166.9	109.5	$t-31$	119.7	103.0	84.9	149.9	158.2
$t-6$	113.2	91.6	73.8	164.1	108.7	$t-32$	116.7	102.4	84.9	153.7	149.6
$t-7$	111.9	88.3	70.2	169.0	111.1	$t-33$	115.8	107.2	86.1	167.0	152.8
$t-8$	99.7	80.8	64.3	143.8	101.0	$t-34$	113.7	104.5	78.9	181.0	144.3
$t-9$	105.1	86.1	71.8	151.3	105.4	$t-35$	115.7	105.8	79.5	189.4	155.5
$t-10$	100.9	81.8	71.7	148.3	109.6	$t-36$	114.4	104.8	79.1	197.9	154.2
$t-11$	105.0	85.6	69.5	140.6	112.8	$t-37$	113.8	103.8	79.9	201.7	154.5
$t-12$	105.2	84.6	70.5	131.5	113.6	$t-38$	114.0	107.0	82.0	196.3	158.0
$t-13$	107.0	90.3	74.9	138.0	117.4	$t-39$	114.1	107.4	77.2	188.0	163.8
$t-14$	109.0	88.3	78.5	135.4	123.1	$t-40$	111.5	112.0	78.5	189.9	164.3
$t-15$	111.4	85.6	73.0	114.0	124.5	$t-41$	109.2	106.8	77.1	172.1	163.9
$t-16$	107.2	81.6	74.5	116.1	118.7	$t-42$	110.1	105.3	76.1	178.0	165.6
$t-17$	111.3	87.4	75.0	121.6	125.0	$t-43$	112.8	113.1	82.6	171.0	161.4
$t-18$	108.6	87.5	77.0	127.6	127.7	$t-44$	111.0	116.6	91.4	179.5	165.4
$t-19$	105.6	86.6	72.4	116.2	121.2	$t-45$	105.6	126.7	94.5	180.6	160.9
$t-20$	105.9	90.0	71.4	128.6	125.1	$t-46$	107.3	123.1	90.0	173.5	162.0
$t-21$	104.7	91.4	68.9	129.4	119.9	$t-47$	103.2	112.3	88.5	164.4	153.0
$t-22$	107.7	95.3	69.7	137.1	117.9	$t-48$	102.8	103.1	81.8	164.2	141.2
$t-23$	107.4	92.9	68.6	134.5	124.6	$t-49$	93.9	95.0	80.7	153.0	133.4
$t-24$	108.0	97.0	70.2	156.0	124.3	$t-50$	93.6	92.7	80.5	164.0	139.3
$t-25$	104.7	102.6	74.3	159.5	128.8						

Table 18.2: Stock returns for 50 different scenarios

Although the stock RD has a negative expected return, it is a major component in the optimal portfolio for low values of minimal expected return. This unexpected result is an artifact of the problem formulation and is addressed in the next section. The stock is included due to its relative stable behavior which stabilizes the overall performance of the portfolio and is therefore used to reduce the overall risk of the portfolio. Only for large values of minimal expected rates of return (over the 0.02 on a weekly basis, so over the 10 percent on a yearly basis) the budget fraction of the stock UN will be larger than the fraction of RD.

Comment

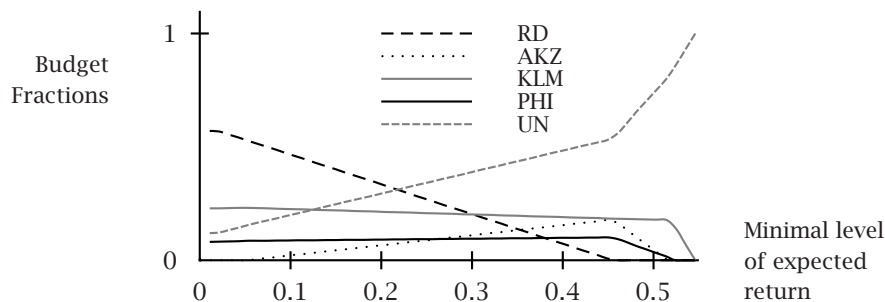


Figure 18.5: Portfolio diversification

18.8 One-sided variance as portfolio risk

In this section, the notion of one-sided variance will be introduced as an extension of regular variance. Based on this new notion a more realistic reformulation of the basic Markowitz model can be developed.

This section

A serious drawback of using variance as a measure of risk is that it penalizes both high and low portfolio returns. In this sense, it fails to capture an investor's basic preference for higher rather than lower portfolio returns.

Main fallacy of variance as portfolio risk

The concept of one-sided variance is similar to the concept of variance but considers only deviations either below (downside variance) or above (upside variance) of some specific target value. For each random variable R_j consider the following two definitions of one-sided variance with respect to the desired expected portfolio return M .

Concept of one-sided variance

$$\begin{aligned}\text{DownVar}[R_j, M] &= E[(\max[M - R_j, 0])^2] = \sum_{t | r_{tj} \leq M} (M - r_{tj})^2 p(r_t) \\ \text{UpVar}[R_j, M] &= E[(\max[R_j - M, 0])^2] = \sum_{t | r_{tj} \geq M} (r_{tj} - M)^2 p(r_t)\end{aligned}$$

Reflecting the investor's preference for higher rather than lower portfolio returns, the focus in this section will be on downside variance of a portfolio as the risk measure to be minimized.

Downside variance of a portfolio

$$\begin{aligned}\text{DownVar}[\sum_j R_j x_j, M] &= E[(\max[M - \sum_j R_j x_j, 0])^2] \\ &= \sum_{t | \sum_j r_{tj} x_j \leq M} [M - \sum_j r_{tj} x_j]^2 p(r_t)\end{aligned}$$

The above expression makes reference to the unknown budget fractions x_j inside the condition controlling the summation. Such expressions cannot be handled by current solution packages, as these require the structure of the constraints to be known and fixed prior to solving the model. That is why another representation must be found such that the special condition controlling the summation is no longer present.

Reformulation required

Whenever you are required to reference positive and/or negative values of an arbitrary expression, it is convenient to write the expression as the difference between two nonnegative variables. This reformulation trick was already introduced in Chapter 6.

Introduce new variables

As indicated before, the focus is on downside variance, and only one new variable needs to be introduced. Let the new variable $q_t \geq 0$ measure the below deviations of the target value M . Then,

*Equivalent
formulation*

$$\text{Minimize} \sum_{t | \sum_j r_{tj} x_j \leq M} [M - \sum_j r_{tj} x_j]^2 p(r_t)$$

can be rewritten as

Minimize:

$$\sum_t p(r_t) q_t^2$$

Subject to:

$$\begin{aligned} M - \sum_j r_{tj} x_j &\leq q_t & \forall t \\ q_t &\geq 0 & \forall t \end{aligned}$$

Note that this reformulation does not result in a simple computational formula, but in an optimization model with inequalities. The objective function will force the nonnegative q_t variables to be as small as possible. This results in $q_t = M - \sum_j r_{tj} x_j$ whenever M is greater than or equal to $\sum_j r_{tj} x_j$, and $q_t = 0$ otherwise.

Comment

Based on the above development concerning the downside risk of a portfolio, the tactical quadratic optimization model of Section 18.6 can be rewritten with x_j and q_t as the decision variables.

*Summary of
model using
downside
variance*

Minimize:

$$\sum_t p(r_t) q_t^2$$

Subject to:

$$\begin{aligned} \sum_j r_{tj} x_j + q_t &\geq M & \forall t \\ \sum_j m_j x_j &\geq M \\ \sum_j x_j &= 1 \\ x_j &\geq 0 & \forall j \\ q_t &\geq 0 & \forall t \end{aligned}$$

Global optimality of any optimal solution to the above model is guaranteed. As before, the quadratic and minimizing objective function possesses a positive semi-definite matrix of second-order derivatives, and all constraints are linear.

Global optimality guaranteed

To compare the computational results for the two notions of variance, Table 18.3 presents the optimal budget fractions for a minimal level of expected return of $M = 0.2$. Note that a lower total risk value associated with down-sided variance does not necessarily imply a lower risk, because there is no ordinal relationship between both risk measures.

Solution

	two-sided variance	down-sided variance
RD	0.335	0.337
AKZ	0.066	0.046
KLM	0.214	0.351
PHI	0.091	0.017
UN	0.294	0.250
total risk	8.982	5.189

Table 18.3: Optimal budget fractions for $M = 0.2$

18.9 Adding logical constraints

This section discusses several logical conditions that can be added to the portfolio selection models in this chapter. The resulting models are mixed-integer quadratic programming models. When linearized, these models can be successfully solved within AIMMS.

This section

Investing extremely small fractions of the budget in an investment category or individual security is unrealistic in real-life applications. A natural extension is to introduce an either-or condition. Such a condition specifies for each investment category or individual security to invest either at least a smallest positive fraction of the budget or nothing at all.

Imposing minimum fractions

Some financial institutions may charge a fixed fee each time they execute a transaction involving a particular type of security. Such a fee may have a limiting effect on the number of different securities in the optimal portfolio, and is significant enough in most real-world applications to be considered as a necessary part of the model.

Fixed fee for transaction costs

A portfolio manager may have specific preferences for various types of securities. Some of these preferences are of the form: if security of type A is to be included in the optimal portfolio, then also security of type B has to be included. Such conditional selections result from practical considerations, and therefore form a natural extension of the model.

Conditional selections

The logical conditions described in the previous paragraphs can be translated into new constraints and new variables to be added to the portfolio models developed thus far. None of the above logical conditions are worked out in detail in this chapter, as you have already encountered them in previous chapters. The formulation tricks involving binary decision variables are described in detail in Chapter 7 with additional illustrations thereof in Chapter 9.

Use logical variables

Adding binary variables to the quadratic programming model of the previous section requires the availability of a solver for quadratic mixed-integer programming. One way to circumvent the need for this class of algorithms is to approximate the quadratic terms in the objective by piecewise linear functions, thus obtaining a linear formulation. Adding binary variables to that formulation causes the entire model to become a mixed-integer linear program, for which solvers are readily available.

Motivation to linearize the objective

18.10 Piecewise linear approximation

In this section the piecewise approximation of the quadratic function $f(q_t) = q_t^2$ is explained in detail. Special attention is paid to the determination of the overall interval of approximation, the quality of the approximation, and the corresponding division into subintervals.

This section

Figure 18.6 illustrates how a simple quadratic function can be approximated through a piecewise linear function. The function domain is divided into equal-length subintervals. By construction, both the true function value and the approximated function value coincide at the endpoints of each subinterval. The slopes of the linear segments increase from left to right, which is what you would expect for a piecewise convex function. Through visual inspection you might already conclude that the approximation is worst at the midpoints of each subinterval. As will be shown, the size of the corresponding maximum approximation error is the same for each interval, as long as intervals are of equal length.

Illustration of piecewise approximation

Recall from the previous section that the quadratic objective function to be minimized is $\sum_t p(r_t)q_t^2$. The individual quadratic terms $f(q_t) = q_t^2$ can each be approximated independently over a finite portion of q_t -axis divided into subintervals indexed with t and l . The length of each subinterval is denoted

Components piecewise formulation

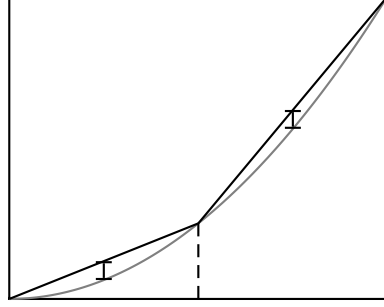


Figure 18.6: Piecewise linear approximation illustrated

with \bar{u}_{tl} . For each subinterval, a variable u_{tl} is introduced where

$$0 \leq u_{tl} \leq \bar{u}_{tl} \quad \forall (t, l)$$

In addition, the *slope* of the function $f(q_t)$ in each subinterval is defined as

$$s_{tl} = \frac{f(q_{tl}^e) - f(q_{tl}^b)}{q_{tl}^e - q_{tl}^b} \quad \forall (t, l)$$

where q_{tl}^b and q_{tl}^e denote the beginning and end values of the intervals, respectively. The following three expressions, defining the approximation of each individual term $f(q_t)$, can now be written.

$$\begin{aligned} f(q_t) &= \sum_l s_{tl} u_{tl} & \forall t \\ q_t &= \sum_l u_{tl} & \forall t \\ u_{tl} &\leq \bar{u}_{tl} & \forall (t, l) \end{aligned}$$

The above approximation only makes sense if the variable $u_{tl} = \bar{u}_{tl}$ whenever $u_{t,l+1} > 0$. That is, u_{tl} must be filled to their maximum in the left-to-right order of the intervals l , and no gaps are allowed. Fortunately, this condition is automatically guaranteed for convex functions to be minimized. The slope s_{tl} increases in value for increasing values of l , and any optimal solution in terms of the u_{tl} -variables will favor the variables with the lowest s_{tl} -values.

Correctness

Recall that q_t denotes downside variance, which is always greater than or equal to zero. The largest value that q_t can attain is when r_{tj} attains its smallest value over all investment categories or individual securities j , and the corresponding fraction x_j is equal to one. It is highly unlikely that x_j will be one, but this value establishes the proper interval size for q_t .

Function domain

$$0 \leq q_t \leq \bar{q}_t \equiv \min_j r_{tj} \quad \forall t$$

For the special case of quadratic terms $f(q_t) = q_t^2$, the general expression for the slope of the linear approximation

The value of slope s_{tl}

$$s_{tl} = \frac{f(q_{tl}^e) - f(q_{tl}^b)}{q_{tl}^e - q_{tl}^b} \quad \forall (t, l)$$

reduces to the following simple expression in terms of endpoints.

$$s_{tl} = \frac{(q_{tl}^e)^2 - (q_{tl}^b)^2}{q_{tl}^e - q_{tl}^b} = \frac{(q_{tl}^e + q_{tl}^b)(q_{tl}^e - q_{tl}^b)}{q_{tl}^e - q_{tl}^b} = q_{tl}^e + q_{tl}^b \quad \forall (t, l)$$

The function q_t^2 is convex, and the linear approximation on the interior of any subinterval of its domain overestimates the true function value. The point at which the approximation is the worst, turns out to be the midpoint of the subinterval. The steps required to prove this result are as follows. First write an error function that captures the difference between the approximated value and the actual value of q_t^2 on a particular subinterval. Then, find the point at which this error function attains its maximum by setting the first derivate equal to zero.

Approximation is worst at midpoint ...

Consider the error function to be maximized with respect to u_{tl}

... and can be derived as follows

$$((q_{tl}^b)^2 + s_{tl}u_{tl}) - (q_{tl}^b + u_{tl})^2$$

By taking the first derivative with respect to u_{tl} and equating this to zero, the following expression results.

$$s_{tl} - 2(q_{tl}^b + u_{tl}) = 0$$

Using the fact that $s_{tl} = q_{tl}^e + q_{tl}^b$, the value of u_{tl} for which the above error function is maximized, becomes

$$u_{tl} = \frac{q_{tl}^e - q_{tl}^b}{2}$$

Note that a *maximum* occurs at this value of u_{tl} , because the second derivative of the error function is negative (a necessary and sufficient condition). As a result, the maximum is attained at the midpoint of the subinterval.

$$q_{tl}^b + u_{tl} = q_{tl}^b + \frac{q_{tl}^e - q_{tl}^b}{2} = \frac{q_{tl}^e + q_{tl}^b}{2}$$

The size of the maximum approximation error ϵ_{tl} can be determined in a straightforward manner by substituting the optimal u_{tl} expression in the error function. This requires some symbolic manipulations, but finally results in the following simple compact formula.

Maximum approximation error

$$\epsilon_{tl} = \frac{(q_{tl}^e - q_{tl}^b)^2}{4}$$

Note that the above maximum approximation error is a function of the *length* of the subinterval, and is in no way dependent on the *position* of the interval. This implies that the choice of equal-length subintervals is an optimal one when you are interested in minimizing the maximum approximation error of the piecewise linear approximation of a quadratic function. In addition, the number of subintervals n_t dividing the overall interval $[0, \bar{q}_t]$ can be determined as soon as the desired value of an overall ϵ , say $\bar{\epsilon}$, is specified by the user of the model. The following formula for the number of subintervals n_t of equal size guarantees that the maximum approximation error of q_t will never be more than $\bar{\epsilon}$.

Number of subintervals

$$n_t = \left\lceil \frac{\bar{q}_t}{2\sqrt{\bar{\epsilon}}} \right\rceil$$

Using the notation developed in this section, the following piecewise linear programming formulation of the portfolio selection model from the previous section can be obtained.

The piecewise linear program

$$\begin{aligned}
 &\text{Minimize:} && \sum_t p(r_t) \sum_l s_{tl} u_{tl} \\
 &\text{Subject to:} && \sum_j r_{tj} x_j + \sum_l u_{tl} \geq M \quad \forall t \in T \\
 &&& \sum_j m_j x_j \geq M \\
 &&& \sum_j x_j = 1 \\
 &&& x_j \geq 0 \quad \forall j \\
 &&& 0 \leq u_{tl} \leq L_{tl} \quad \forall (t, l)
 \end{aligned}$$

18.11 Summary

In this chapter, both a strategic and a tactical portfolio selection problem have been translated into a quadratic programming model. The relatively small strategic model uses a covariance matrix as input, whereas the relatively large tactical model uses historic rates of return as scenarios to estimate the risk and expected return of a portfolio. Both models can be used to determine the particular combination of investment categories or securities that is the least risky for a given lower bound on expected return. Apart from single optimal solutions, parametric curves depicting the trade-off between risk and return were also provided. Several properties of the investment model were investigated. It was shown that (a) any optimal solution is also a global optimum, (b) the risk-reward curve is nondecreasing and convex, and (c) multiple optimal portfolio returns are perfectly correlated. An improvement to the model was introduced by minimizing only downside risk, thus making the model more realistic. Further extensions were suggested to take into account such real-world requirements as minimum investment fractions, transaction costs and

conditional security selections. Finally, a piecewise linear approximation of the quadratic objective function was introduced in order to keep the model with logical constraints within the framework of mixed-integer linear programming.

Exercises

- 18.1 Implement the strategic investment model presented in Section 18.2 using the example data provided in Table 18.1. Use AIMMS to reproduce the risk-reward curve illustrated in Figure 18.2.
- 18.2 Implement the tactical investment model presented in Section 18.6 using the example data presented in Table 18.2. Modify the objective function to represent downside variance as the measure of portfolio risk, and compare the result with the solution presented in Table 18.3.
- 18.3 Implement the piecewise linear formulation of the tactical investment model as described at the end of Section 18.10. Add the logical requirement that either at least 5% of the total budget is invested in any particular security or 0%. In addition, add the requirement that whenever the percentage invested in security 'RD' is greater than 20%, then the percentage invested in security 'KLM' has to be less than 30%. If the number of intervals becomes too large for a particular t and a particular ϵ , design a dynamic scheme to adjust the interval length based on a previous solution.

Chapter 19

A File Merge Problem

This chapter considers the merging of two statistical database files. The problem can be formulated as a transportation model and solved using a linear programming solver or a specialized network solver. However for large files, the number of variables in the underlying model is too large. To overcome this, a column evaluation approach is proposed. Specifically, a customized solution algorithm which controls the size of the network to be solved by systematically considering a subset of all columns each major iteration. The underlying theory is explained, and subsequently applied to the file merge model.

This chapter

The problem and its formulation have been adapted from Glover et al. ([G192]). The underlying theory of the simplex method and column generation can be found in [Ch83].

References

Linear Program, Network Program, Simplex Method, Column Generation, Mathematical Derivation, Customized Algorithm, Worked Example.

Keywords

19.1 Problem description

In this section the problem of merging an income data file and a population data file is described. The structure of these files is examined, and the file merge problem is viewed as a distance minimization problem.

This section

Statistical databases are typically developed and maintained in such government institutions as statistical offices, ministries and planning agencies. These databases are the result of extensive surveys involving (tens of) thousands of households or businesses, and contain information that can be used to analyze, for instance, the effect of government policy measures. Examples are the study of welfare measures, the effect of social security benefits or taxation on government income, etc.

*Statistical
database files
...*

A statistical database file consists of similar records. Each record has a fixed number of data fields. These data fields contain values that are applicable to a group of households or businesses with similar characteristics. As a result, data is not stored per individual household or business, but aggregated (in some way) for each group. That is why the number of similar households or businesses is always part of each record. The net effect is that the number of records in a file is much smaller than when records are maintained for each individual household or business. Nevertheless, the number of records may still be in the order of thousands. As you will see in later paragraphs, the data field containing the number of families or businesses in each record will play a central role in both the problem and model formulation.

... and their structure

One example of a statistical database file used in this chapter is a file referred to as the 'Income Data File' (see [G192]). Each record describes some specific income characteristics of families, and also contains the number of families sharing these characteristics. These families from one record are of course not identical in all respects. For instance, in Table 19.1, the 'Gross Family Income' is an average and thus not exact for an individual family, while the 'Source of Income' is identical for all families in the record.

Income data file

Record	No. of Families	Gross Family Income	No. of Family Members	Source of Income	Interest Income
1	20	10,000	3	Commerce	0
2	30	15,500	2	Commerce	1,000
3	25	20,000	5	Agriculture	1,000
4	18	25,000	4	Agriculture	3,000
5	32	15,000	3	Commerce	500
:	:	:	:	:	:

Table 19.1: Income Data File

Another example of a statistical database file used in this chapter is a file referred to as the 'Population Data File' (see [G192]). Each record describes some specific demographic characteristics of families. Again, the families within a record are not identical in all respects. For instance, in Table 19.2, the 'Number of Family Members' is just an indication for the group as a whole, while the 'Head of Household' characteristics may apply to all families in the record.

Population data file

Consider the evaluation of a tax reduction scheme. Such a scheme is usually based on a partitioning of households in terms of both income and demographic characteristics. Unfortunately, it is not clear how families in the 'Income Data File' are related to families in the 'Population Data File'. What is needed is a way to combine these two files, so that each new record describes

Need to merge

Record	No. of Families	Gross Family Income	No. of Family Members	Head of Household			No. of Family Members Under 18
				Age	Education	Sex	
1	25	15,000	4	40	12	M	2
2	30	15,000	2	25	16	M	0
3	18	20,000	1	30	18	F	0
4	27	25,000	2	35	16	F	1
5	25	20,000	4	25	12	M	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 19.2: Population Data File

both income and demographic characteristics for an entire group of similar families.

Records in both files contain common information such as ‘Number of Families’, ‘Gross Family Income’ and ‘Number of Family Members’. These data fields form the basis for merging the two files. In practical applications, however, there is not a one-to-one mapping between records in each file on the basis of common characteristics. The database files, presumably derived from the same population, are typically based on data from different samples of households. Merging records in such a case is no trivial exercise.

Common information

With database files from different sources, the merging of records becomes somewhat arbitrary. One could even wonder whether entire records should be merged. Perhaps the merging of two particular records should take place for just a subset of families in each record. In any case, some measure has to be developed to determine whether two records are similar enough to be merged. Such a measure can only be based on common information. When the common information is exact, only records with matching entries can be combined. In such a situation, quite a few combinations of records can be ignored. When the common information in a data field is an average or a typical value, then records with differing, but sufficiently close, entries can be combined. One could speak of “distance” between records, where pairs of records with low distance are regarded as similar, and the ones with large distance are regarded as dissimilar.

Distance between records

Assume that a distance measure has been determined. Then the goal is to combine records in such a way that the sum of distances for all combined records is minimal. Of course, when two records are combined in this process, it must be decided how many families from each of them share this new record. In addition, the new value of each income and demographic data field must be decided. A specific approach to deal with these questions is proposed in the next section.

Decisions to be made

19.2 Mathematical formulation

In this section the translation of the file merge problem into a network model is proposed and evaluated. The construction of an overall distance function is illustrated for the merging of the ‘Income’ and ‘Population’ data files. In addition, suggestions are made for the computation of data entries that make up the newly constructed records.

This section

As noted in the previous section the merging of records is arbitrary, and several approaches can be thought of. Without the use of an optimization model you might think of sorting each of the two files according to one or more major characteristics, and then select families from both sorted files in increasing order to make up new records. Other similar heuristic approaches can be thought of, but they all suffer from the fact that issues are not looked at simultaneously but only sequentially in an ad hoc manner. This is why a mathematical formulation based on simultaneous constraints is proposed in this section.

Several approaches

In several modeling exercises the translation of a problem into a model is not obvious at first, but almost trivial or self-evident afterwards. This is also the case with the file merge problem. Each record contains a data field for the number of families, and somehow families in records of the first file must be assigned to families in records of the second file. This sounds like an assignment problem of some sort, which may remind you of the network applications in Chapter 5. As it turns out, the key observation is to view each record as a node in a network.

Network approach

Consider the records of one file as supply nodes in a transportation network with the ‘Number of Families’ as the available supply. Similarly, consider the records of the second file as demand nodes with the ‘Number of Families’ as the required demand. Throughout this chapter it is assumed that the total supply of families is equal to the total demand of families. Next consider a decision variable for each pair of nodes (records) to express how many families from a particular supply node will be “shipped” (i.e. assigned) to a particular demand node. You now have the ingredients for describing a set of constraints which contains all permitted assignments of families.

View as transportation model

Indices:		<i>Transportation constraints</i>
i	<i>supply nodes (records i)</i>	
j	<i>demand nodes (records j)</i>	
Parameters:		
N_i	<i>number of families in record i</i>	
N_j	<i>number of families in record j</i>	

Variable:

x_{ij} number of families shipped from i to j

Constraints:

$$\begin{aligned}\sum_j x_{ij} &= N_i & \forall i \\ \sum_i x_{ij} &= N_j & \forall j \\ x_{ij} &\geq 0 & \forall (i, j)\end{aligned}$$

When the simplex method is applied to the above set of equalities, then the solution values x_{ij} are guaranteed to be integer as long as both N_i and N_j are integer. This (unimodularity) property has been referred to in Chapter 2.

Integer solutions

In the context of the file merge problem you may interpret any feasible solution of the above set of constraints as follows. Whenever the variable x_{ij} is positive, records i and j will be merged to form a new record, and the value of x_{ij} is the number of families sharing that new record. As the total number of families (summed over all records) in both files are identical, all families will be placed in some new record. Note that nothing has been said thus far concerning the contents of the income and demographic data fields of the new records. This will be discussed later.

Interpretation

When you add a total distance function as the objective function to the above set of constraints, you obtain a complete optimization model. Any optimal solution of this model states how existing records must be merged to form new records such that the total distance between merged records is minimal. Let d_{ij} be a parameter containing the distance between records i and j . Then the objective function to be added to the above constraints becomes

Objective function

Minimize:

$$\sum_{ij} d_{ij} x_{ij}$$

Similarity between records can be stated in a variety of ways. The following formulation of distance was developed and motivated in [G192].

Formulation of distance

Parameters:

G_i, G_j 'Gross Family Income' in record i or j
 M_i, M_j 'Number of Family Members' in record i or j
 s_G^2 estimated variance of all G_i and G_j values
 s_M^2 estimated variance of all M_i and M_j values

The proposed measure of distance d_{ij} is then as follows.

$$d_{ij} = \sqrt{\frac{(G_i - G_j)^2}{s_G^2} + \frac{(M_i - M_j)^2}{s_M^2}}$$

Note that by dividing the squared deviation of two data values by their variance, the computed values become comparable in size. Such normalization avoids the situation that deviations in one parameter strongly dominate deviations in another parameter.

Once the solution of the above optimization model is obtained, the number of families for each new record is known. However, the value of the other data fields must still be computed. As stated previously, there is no unique method to determine new data entries whenever the originating records show differing values. Choices have to be made by those who are involved in the construction process. The following constructs are merely two different suggestions. Let the index $n(i, j)$ refer to a new record derived from records i and j . Then the new values of ‘Gross Family Income’ and ‘Number of Family Members’ can be computed as follows.

*Constructing
data fields*

$$\begin{aligned} G_{n(i,j)} &= (G_i + G_j)/2 && \text{(average)} \\ M_{n(i,j)} &= \max\{M_i, M_j\} && \text{(largest)} \end{aligned}$$

In the file merge model the role of the two database files can be reversed, because the total number of families in each file are assumed to be identical. The question then arises whether such reversal has any effect on the optimal solution. The answer is negative. First of all, any feasible shipments from supply nodes to demand nodes are also feasible shipments from demand nodes to supply nodes. This implies that the set of feasible solutions is not affected by the role reversal of the two database files. Secondly, in the total distance function the coefficients are symmetric for each pair of records. As a result, the optimal solution value remains optimal when the shipping direction between the supply and demand nodes is reversed.

*Choice of origin
and destination*

An initial guess concerning the size of the newly formed merged file, might lead to the following bounds. Let $|I|$ and $|J|$ denote the number of records in file 1 and 2 respectively. Then $\max\{|I|, |J|\}$ seems to be the smallest number of records in the newly merged file, while $|I| \times |J|$ seems to be the largest such number. The lower bound is correct, but the upper bound is excessively over estimated. According to the theory of the simplex method (explained in Section 19.4), the maximum number of decision variables away from their bounds is equal to the number of constraints. For the above transportation model this implies that the maximum number of positive decision variables is at most $|I| + |J|$. This value is then a much improved upper bound on the number of records in the merged file.

*Size of merged
file*

Consider the first five records of both the Income Data File in Table 19.1 and the Population Data File in Table 19.2. The total number of families in each of these two portions is 125. Let the distance between records be determined by the differences between 'Gross Family Income'. Then the graph in Figure 19.1 displays a possible merging scheme for which total distance has been kept to a minimum. The number associated with a node is the number of families in the corresponding original record. The number associated with an arc is the number of families in the corresponding new record.

Example of network ...

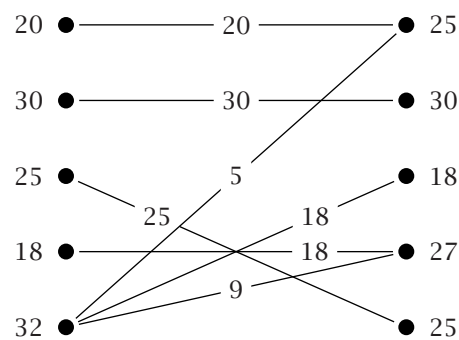


Figure 19.1: Network representation of a solution

On the basis of Figure 19.1 it is now straightforward to construct the merged file as displayed in Table 19.3. As expected, the total Number of Families has remain unchanged. Only the entries underneath the common headers 'Gross Family Income' and 'Number of Family Members' have to be reconciled. In this example, 'Gross Family Income' in the merged file is the average of the entries in the originating files. The 'Number of Family Members' in the merged file is determined differently, and has been set to the maximum of the originating entries.

... and resulting merged file

Record	Gross		No. of	Source of	Interest	Head of Household				No. of
	No. of	Family				Age	Education	Sex	Under 18	
1	20	12,500	4	Commerce	0	40	12	M	2	
2	30	15,250	2	Commerce	1,000	25	16	M	0	
3	25	20,000	5	Agriculture	1,000	25	12	M	1	
4	18	25,000	4	Agriculture	3,000	35	16	F	1	
5	5	15,000	4	Commerce	500	40	12	M	2	
6	18	17,500	3	Commerce	500	30	18	F	0	
7	9	20,000	3	Commerce	500	35	16	F	1	

Table 19.3: Merged Data File

19.3 Solving large instances

This section presents an overview of a method to solve large instances of the file merge model. The key idea is to use an iterative approach that systematically adds and removes variables so that an overall optimized model is found.

This section

The number of decision variables and objective coefficients in the file merge model is the product of the two file sizes and consequently the model size can become unmanageable very quickly. When the number of records in each file is of the order $O(10^2)$, then the number of decision variables (reflecting all possible pairs of records) is of the order $O(10^4)$. Such a model is not considered to be large by today's standards. However, when the number of records in each file is of the order $O(10^4)$, then the number of variables is of the order $O(10^8)$. In general, models of this size cannot be solved in its entirety using currently available technology. For standard hardware and software configurations, either a special solution approach is required, or the model must be reduced in size prior to being solved.

Model size

One approach is to consider whether all possible combinations of records must be included. The number of variables can be reduced significantly if you only consider those variables with a distance value d_{ij} less than or equal to some sufficiently low cutoff value. Alternatively, a controlled number of variables can be generated if only the k smallest d_{ij} values are considered for each i . There are several other such schemes, all aimed at reducing the number of variables to a manageable level.

A priori reduction ...

Using these plausible suggestions many variables can be eliminated, but the question remains whether such a priori reduction will result in unwanted side effects. The answer is surprisingly yes. The main reason is that these reduction schemes are based on the values of d_{ij} alone, and do not take into account the values of both N_i and N_j . In many applications, the reduction schemes discussed above lead directly to infeasible solutions.

... may not always work

A better approach to reduce the number of variables would be to carry out the following three steps.

A better approach

1. Apply a heuristic to determine at least one feasible solution.
2. Consider some form of a priori reduction.
3. Extend the set of variables iteratively and selectively until the optimal solution is found.

Note that feasibility is guaranteed by construction. The quality of the optimal solution for the reduced model following the second step should be quite good, as several variables with low d_{ij} values are already included in

the model. The key to a successful implementation of the last step is the identification of variables that will improve the objective function. It turns out that the simplex method of linear programming provides a basis to implement this step. The underlying theory is explained in the next section, and is subsequently applied to the file merge problem in Section 19.5.

As already highlighted, in practical applications the number of distance coefficients d_{ij} 's may be so large that it is impractical to store them. However, the coefficients do not need to be stored since it is possible to calculate d_{ij} from its definition during runtime. The value of all d_{ij} 's can be calculated from just $|I| + |J|$ records. Clearly, calculating d_{ij} will consume significant runtime and therefore care should be given when specifying the expression to reduce calculation overhead. In AIMMS it is possible to specify the objective coefficients (like all parameters) using expressions. Consequently, the solution method presented above can be implemented in AIMMS such that the distance coefficients are calculated during runtime as required.

Calculating d_{ij}

19.4 The simplex method

This section describes the simplex algorithm using matrix-vector notation for the underlying linear algebra. The algorithm forms the basis for the column evaluation technique used in this chapter, and the column generation technique used in Chapters 20 and 21.

This section

Without loss of generality all linear programming constraints can be written as equalities. Specifically, an inequality can be transformed to an equality by introducing a slack or surplus variable. In the Simplex method, the variables in the model are partitioned into two groups: the basic variables x_B and the nonbasic variables x_N . By definition, nonbasic variables are at one of their bounds (upper or lower) while basic variables are between their bounds. The matrices associated with the basic and nonbasic variables are denoted with B and N , respectively.

Basic and nonbasic variables

It is important to note that the choice of x here follows standard notation and it is not related to the x_{ij} used in the file merge model. Similarly, the matrix N is not related to N_i or N_j . The scope of this notation is limited to the current section.

Notation

Minimize:

$$c_B^t x_B + c_N^t x_N$$

Partitioned linear program

Subject to:

$$\begin{aligned} Bx_B + Nx_N &= b \\ x_B, x_N &\geq 0 \end{aligned}$$

After rewriting the equality constraint, and using the fact that the optimal basis is invertible, the basic variables x_B can be written in terms of the nonbasic variables x_N .

*Solution
rewritten*

$$x_B = B^{-1}b - B^{-1}N x_N \geq 0$$

Next, this expression for x_B is substituted in the objective function to obtain the following form.

*Objective
function
rewritten*

$$c_B^t B^{-1}b + (c_N^t - c_B^t B^{-1}N)x_N$$

Taking the vector derivative of the last expression with respect to b , gives $\lambda^t = c_B^t B^{-1}$. This defines the *shadow price* of the associated constraint. As explained in Chapter 4, it is the rate of change of the objective function for a unit increase in the right-hand side of the constraint.

Shadow prices

Similarly, taking the vector derivative with respect to x_N gives the term $(c_N^t - c_B^t B^{-1}N) = (c_N^t - \lambda^t N)$. This defines the *reduced cost* of a variable. The reduced cost of a variable gives the rate of change of the objective function for a one unit increase in the bound of the variable. As discussed in Chapter 4, the reduced cost of a basic variable is zero. Reduced costs can be considered to be the sensitivity of the objective function value with respect to bounds associated with the nonbasic variables.

Reduced costs

As previously discussed, nonbasic variables x_N in the simplex method are at one of their bounds. During a simplex iteration, one of these variables is introduced into the basis, and a basic variable leaves the basis to become nonbasic. For the case, as in the file merge problem, where all variables are positive and nonbasic variables are at their lower bound, such an exchange is only of interest (for a minimization problem) when the corresponding component of the reduced cost vector $(c_N^t - \lambda^t N)$ is negative. In this particular case, the objective function value will decrease when the value of the corresponding component of x_N is increased (away from its lower bound of zero). As soon as all components of the reduced cost vector $(c_N^t - \lambda^t N)$ are nonnegative, no improvement in the objective function value can be made, and the current basic solution $x_B = B^{-1}b$ is optimal. Note that, by definition, the reduced costs associated with basic variables are always zero.

*Simplex
iteration*

19.5 Algorithmic approach

This section describes an algorithmic approach to solve the overall file merge model as a sequence of smaller submodels. The construction of each submodel is based on evaluating the reduced cost values of all variables as given by the simplex method. The inter-record distance d_{ij} are computed during runtime, and the corresponding variable is either put into a candidate list or ignored.

This section

Assume that the file merge model has been solved for a subset S of the variables x_{ij} , $(i, j) \in S$. The resulting vector of shadow prices λ can be partitioned into λ^s for the supply constraints and λ^d for the demand constraints with components λ_i^s and λ_j^d respectively. Consider a particular combination of records i and j , $(i, j) \notin S$. After computing d_{ij} directly from the two records, the quantity $d_{ij} - (\lambda_i^s + \lambda_j^d)$ can be evaluated. Whenever this quantity is negative (i.e. may lower the objective function), the corresponding variable x_{ij} is a candidate variable.

*Candidate
variables*

The first step of the algorithm is to find an initial feasible solution using an heuristic approach. Specifically, the records in each file are sorted with respect to 'Gross Family Income'. Next, in each file the first record with a positive value of 'Number of Families' is found. These two records are then merged to form a new record. The 'Number of Families' of the new record is equal to the minimum of the 'Number of Families' associated with the two input file records. The 'Number of Families' associated with each input file are adjusted by subtracting the smallest value of the two. The process is repeated until all records in both files have been considered, and the total number of families has been divided over the new records. All pairs of originating records i and j considered in this process, result in a basic variable $x_{ij} > 0$.

*Initial solution
in words*

In addition to the variables identified in the previous paragraph a further selection can be made on the basis of small d_{ij} values for each i . The number of such additional variables can be as large as you desire. A typical value is between 5 to 25 extra variables for each i . Experience has shown that such a set is quite a good selection, and that for this selection the solution, the objective function value and the shadow prices of the submodel are close to optimal for the much larger model with all $|I| \times |J|$ variables.

*Additional
selection of
variables*

Once an initial optimal solution for the current selection of variables has been computed, the algorithm visits all $|I| \times |J|$ pairs of records. During this process there is an active search for candidate variables which, together with the variables from the previous submodel, will determine the next submodel to be solved. The entire process is repeated until there are no new candidates after visiting all possible pairs of records.

*Overall solution
in words*

The flowchart in Figure 19.2 presents the computational steps to determine the initial values of S and x_{ij} . The set S contains all pairs (i, j) for which the corresponding variable x_{ij} is greater than zero. The element parameters i^* and j^* refer to records. Eventually, the x_{ij} values satisfy the equality constraints, and form a basic solution. At most $|I| + |J|$ values of x_{ij} will be positive, because each iteration the (remaining) number of families from at least one record is assigned. The symbol \wedge represents the logical AND.

*Flowchart initial
solution*

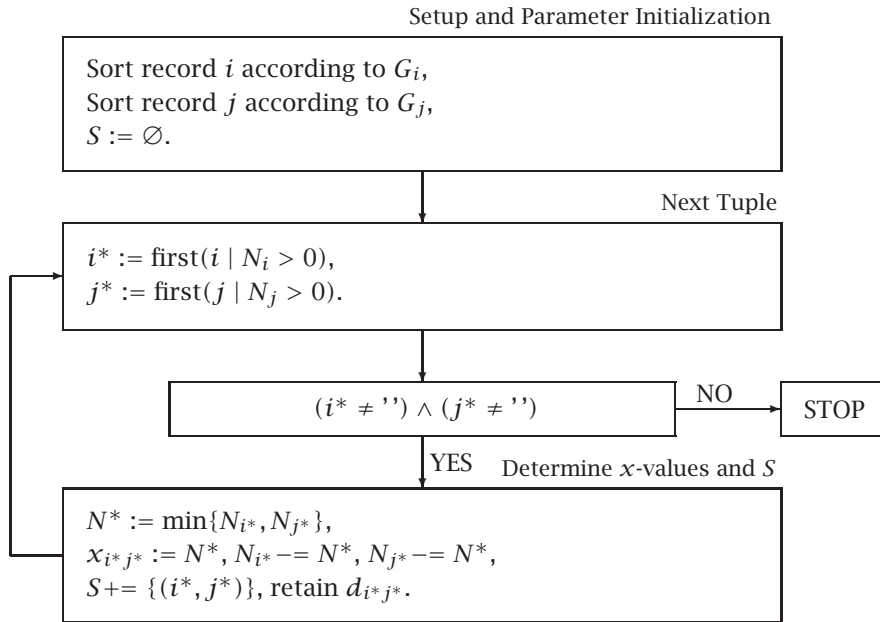


Figure 19.2: Flowchart initial solution

The flowchart in Figure 19.3 presents the computational steps to determine the optimal solution of the overall file merge model. Most of the notation has been introduced previously. New is that the element parameters i^* and j^* can be increased in value. That is, the assignment $i^* += 1$ states that i^* refers to the next record in the sorted set I of records. Any reference beyond the last element is empty (i.e. $''$).

*Flowchart
overall solution*

In the algorithm presented above there is no control over the size of the set S (the set of considered variables x_{ij}). Control could be implemented by either deleting already considered variables or by limiting the number of candidate variables to be added. Deletion could be based on examining the (already considered) variables with the highest (instead of the lowest) $d_{ij} - (\lambda_i^s + \lambda_j^d)$ value. Addition could be based on restricting the maximum number of candidates for each i .

*Computational
considerations*

19.6 Summary

In this chapter the problem of merging of two files has been introduced as an application of the classical transportation problem. However, in practical applications the number of decision variable is extremely large and the resulting LP can not be solved in its entirety. To overcome this problem, a customized solution algorithm has been proposed. The proposal consists of a heuristic

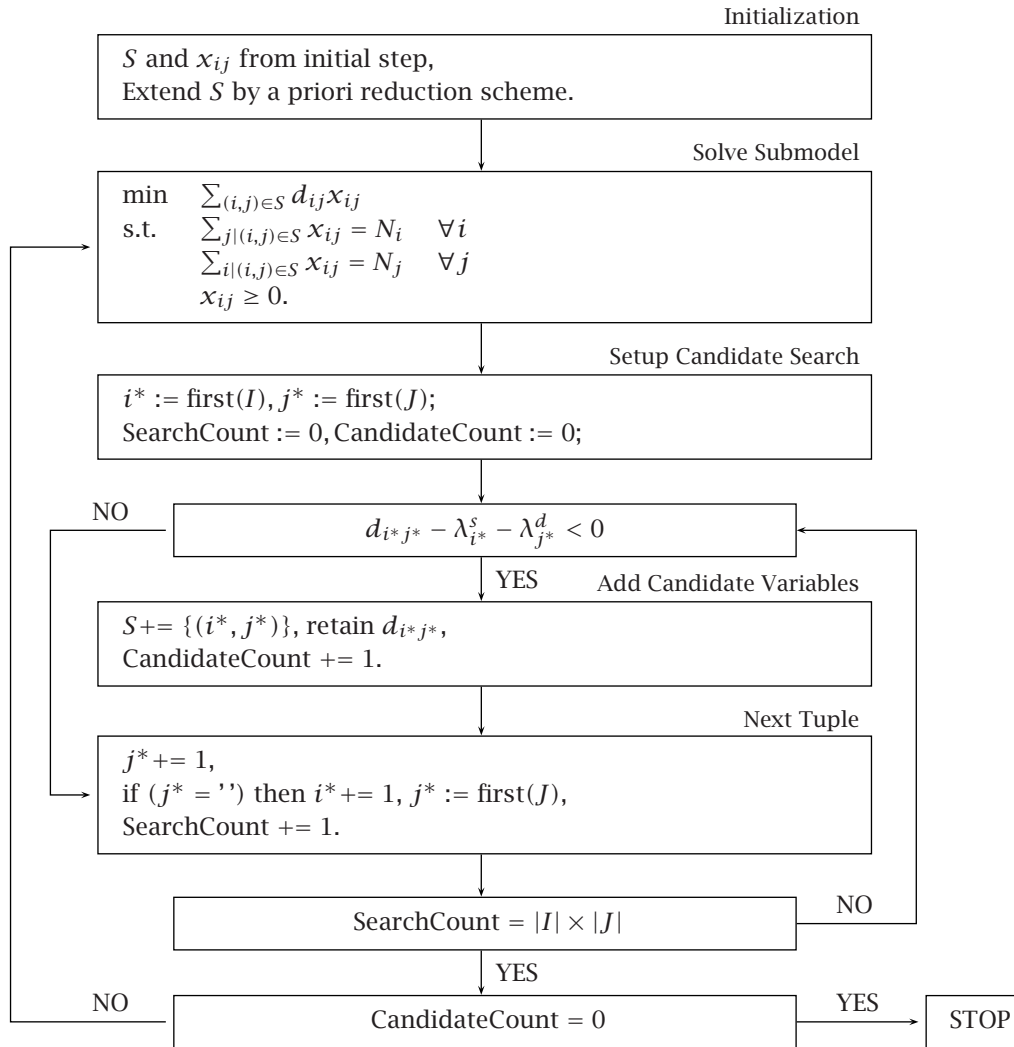


Figure 19.3: Flowchart algorithmic approach

approach to find initial solution values and shadow prices, followed by an algorithm to find the optimal solution of the model through systematically solving a sequence of smaller submodels. The underlying theory and detailed flowcharts have been presented.

Exercises

- 19.1 Implement the file merge model presented in Section 19.2 using the first five records of the Income Data file and the Population Data File contained in Tables 19.1 and 19.2. Verify for yourself whether the

optimal solution found with AIMMS is the same as the one presented in Table 19.3.

- 19.2 How would you adjust your formulation of the model if the number of families in the Income Data File of Table 19.1 were 5 less for each of the five records?
- 19.3 Implement the algorithmic approach presented in Section 19.5 for the model and data referred to in the first exercise. Verify for yourself whether the optimal solution found is the same as the one found previously.

Chapter 20

A Cutting Stock Problem

This chapter applies a *delayed column generation* technique to find a set of optimum cutting patterns for a class of cutting stock problems. Each pattern is essentially a column of the underlying linear program. In practical applications, the number of cutting patterns can be extremely large. However, instead of considering the millions of possible cutting patterns, a submodel of the cutting stock problem is systematically built up with new patterns until it contains the optimum solution. The new patterns are added to the submodel by solving an auxiliary integer program, referred to as the *column pattern generation* model. The chapter begins with a basic model which is then extended.

This chapter

The methodology for cutting stock problems dates back to work of Gilmore and Gomory ([Gi61, Gi63]). A good exposition on this subject and its underlying theory can also be found in [Ch83].

References

Linear Program, Integer Program, Simplex Method, Column Generation, Mathematical Derivation, Customized Algorithm, Auxiliary Model, Worked Example.

Keywords

20.1 Problem description

This section introduces a class of *cutting stock problems*, which are typically encountered in the paper and textile industries.

This section

Materials such as paper and textiles are often produced in long rolls, the length of a truck trailer for instance. These long rolls are referred to as *raws*. These raws are subsequently cut into smaller portions called *finals*, with their sizes specified by customers.

Raws and finals

A raw can be sliced all the way through so that the diameter of each final has the same diameter as the original raw. This is usually what happens when rolls of paper are cut. The slicing of a raw is illustrated in Figure 20.1.

Slicing raws

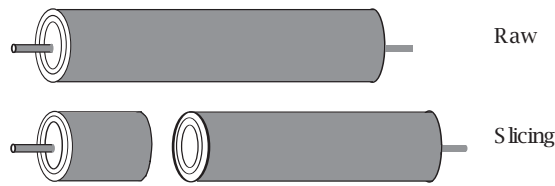


Figure 20.1: Slicing raws of paper and textile

Assume that a production scheduler has a list which specifies the required number of each final size. He must then develop a production schedule detailing the number of rolls and how they should be cut to meet demand.

Demand requirements

The objective of the scheduler is to determine the most economical way to meet the demand. It is assumed that there are no storage space constraints. The objective becomes to minimize the total number of rolls required to make the finals. In this chapter, the more general multi-period inventory case is not addressed. When time periods are considered, the objective is not just to minimize the number of raws used, but also to consider the storage costs involved.

Objective

20.2 The initial model formulation

A natural inclination when initially constructing a model for the cutting stock problem is to consider two sets, namely 'Raws' and 'Finals'. However, on closer inspection, you will note there is only one kind of raw in the problem description. The question then arises: does the set 'Raws' contain only one element (reflecting that only one kind exists), or does this set contain an undetermined number of elements (one for each raw to be cut)? Similarly, should the set 'Finals' contain each final or just the possible sizes of finals? The answer to these questions is not immediately apparent, and further analysis is required.

Investigating the structure

If you have to write down a model for a small example, it is likely you will develop the concept of a cutting pattern. A *cutting pattern* is a specific recipe stating for each size of final how many finals are cut from a single raw. Of course, there are several such recipes. For small examples the size of the set of cutting patterns is not exorbitant, but in most real problems the number of possible cutting patterns could be in the millions.

Cutting patterns

When you have adopted the concept of a cutting pattern as a building block for the model, it is also clear that the set 'Finals' should contain the possible sizes of finals (and not each individual final that is demanded by the customers). This is because the cutting pattern is defined in terms of possible sizes of finals.

The set 'Finals'

Assume for the duration of this section that the size of the set of cutting patterns is workable. A verbal statement of the model is then as follows.

Verbal model description

Minimize: *the number of raws to be used*

Subject to:

for all possible sizes of finals: the number of finals produced from cutting raws according to the set of allowable cutting patterns must meet the demand.

The following integer program is a mathematical representation of the verbal model in the previous paragraph.

Mathematical description

Indices:

p *cutting patterns*
 f *finals*

Parameters:

d_f *demand for final f*
 a_{fp} *number of finals f in cutting pattern p*

Variable:

x_p *number of raws cut with cutting pattern p*

Minimize:

$$\sum_p x_p$$

Subject to:

$$\sum_p a_{fp} x_p \geq d_f \quad \forall f$$

$$x_p \geq 0 \text{ integer} \quad \forall p$$

As the number of cutting patterns increases, the solution time of the underlying integer programming solver will also increase. In which case, an attractive alternative may be to drop the requirement that x_p is integer, and just solve the corresponding linear programming model. A straightforward rounding scheme may then be quite sufficient for all practical purposes. The rounded integer solution may not be optimal, but it can easily be made to satisfy the demand requirements.

Relax integer requirement

A simple and useful heuristic is 'Largest In Least Empty' (LILE). This heuristic is loosely described in the following four steps.

Largest In Least Empty

1. Round the fractional solution values downwards, and determine the unmet demand.
2. Sort the finals in the unmet demand from largest to smallest.
3. Place the largest final from the unmet demand in the least empty raw that can contain this final. If this is not possible, an extra raw must be added.

4. Continue this process until the sorted list of finals from the unmet demand is completely allocated.

It is possible that a pattern generated by this algorithm is one of the patterns used in the relaxed integer programming solution (see Table 20.2 in which pattern 12 is generated again). The LILE algorithm tends to minimize the number of extra rows required, and turns out to work quite well in practice.

Consider an example where the raws are ten meters long, and there are four sizes of finals, namely, 450 cm, 360 cm, 310 cm and 140 cm. The raws can be cut using thirty-seven cutting patterns as shown in Table 20.1.

*Example data
and ...*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
450 cm	2	1	1	1	1	1	1	1	1										
360 cm		1	1							2	2	2	1	1	1	1	1	1	1
310 cm				1	1								2	1	1	1			
140 cm		1		1		3	2	1		2	1			2	1		4	3	2
	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	
450 cm																			
360 cm	1	1																	
310 cm			3	2	2	2	1	1	1	1	1								
140 cm	1			2	1		4	3	2	1		7	6	5	4	3	2	1	

Table 20.1: Thirty-seven cutting patterns to cut a raw of size 1000

With the use of all thirty-seven cutting patterns, the minimum number of raws required according to the linear programming solution is $452\frac{1}{4}$. As expected, the number of times a cutting pattern is used in the optimal solution is fractional. One of the optimal solutions is listed in Table 20.2. Rounding this optimal fractional linear programming solution, using the LILE heuristic, gives an integer objective function value of 453 required raws. This number of 453 is optimal, because the linear programming objective function value of $452\frac{1}{4}$ is a lower bound for the integer objective function value, and 453 is the first integer in line.

... the solution

20.3 Delayed cutting pattern generation

Problems commonly encountered in the paper industry involve raws and finals of arbitrary sizes. The number of possible cutting patterns can then grow into the millions and the approach of the previous section is no longer workable. This section describes an algorithm that makes the problem workable by limiting the number of possible cutting patterns. Instead of going explicitly through millions of cutting patterns, a *cutting stock submodel* is systematically

*Cutting stock
submodel
approach*

Finals	Optimal Patterns				LILE Patterns			Demand
	01	10	12	13	02	12	30	
450 cm	2				1			97
360 cm		2	2	1	1	2		610
310 cm				2			1	395
140 cm		2			1			211
Fractional Solution	$48\frac{1}{2}$	$105\frac{1}{2}$	$100\frac{3}{4}$	$197\frac{1}{2}$				
LILE Solution	48	105	100	197	1	1	1	

Table 20.2: The solutions: linear programming versus LILE

built up to contain the optimum solution by adding patterns identified by solving an auxiliary integer program, referred to as the *column pattern generation model*.

The first step of the algorithm is to create a submodel of the cutting stock problem which contains a set of cutting patterns which will satisfy the requirements. Clearly, this initial set will not (necessarily) be optimal. This submodel is then solved. Using the resulting shadow prices in conjunction with simplex method theory, it is possible to formulate an auxiliary model (cutting pattern generation model) integer program. Solving this model identifies one cutting pattern which is then added to the cutting stock submodel to improve its objective (i.e. to reduce the number of rows). The cutting stock submodel with this extra pattern, is then solved. The process is repeated (using updated shadow prices) until the submodel contains the set of optimum cutting patterns. In practice, the total number of new cutting patterns generated by the cutting pattern generation model is quite small and so the overall algorithm is very efficient.

*Algorithm
description*

Like in Chapter 19, this algorithm is based on the simplex method. However, the approach differs because it takes advantage of the fact that all objective function coefficients in the cutting stock model are identical, while the ones in the file merge model are not. You are referred to Chapter 19 for a discussion of the simplex method and the application of shadow prices and reduced costs.

New approach

The iterative solving of the cutting stock submodel and the cutting pattern generation program is summarized below.

*Iteration
between two
models*

```

Initialize cutting stock submodel
WHILE progress is being made DO
    Solve cutting stock submodel
    Solve cutting pattern generation model
    IF new cutting pattern will lead to improvement
        THEN add it to cutting stock submodel
ENDWHILE

```

The cutting stock submodel can be initialized in many ways. The simplest option is to include one pattern for each final size. With each pattern consisting of the maximum number of finals that can be cut from the raw. For example, for the model in the previous section, you could use patterns 1, 12, 22 and 31. By selecting patterns which include all final sizes, then the first solution will be feasible (but not necessarily optimal).

*Cutting stock
model
initialization*

In addition to the four initial patterns, patterns 10 and 13 were generated by the cutting pattern generation program. These six patterns were sufficient to determine an optimal solution of the cutting stock model.

*Additional
patterns*

Assume there is some cutting pattern y which is not part of the cutting stock submodel. Let y_f be a component of this vector. Each such component corresponds to the number of finals of size f used in the cutting pattern. In addition to y_f , let λ_f denote the shadow price associated with each demand requirement f in the cutting stock submodel. Then cutting pattern y should be added to the submodel whenever

*Mathematical
formulation*

$$1 - \sum_f \lambda_f y_f < 0$$

This condition is the reduced cost criterion in the simplex method when applied to the cutting stock model.

An auxiliary cutting pattern generation model can now be proposed based on the following three observations. First of all, the numbers y_f making up a cutting pattern must be nonnegative integers. Secondly, their values must be such that the cutting pattern does not require more than the total width of a raw. Thirdly, the new cutting pattern values should offer the opportunity to improve the objective function value of the reduced cutting stock problem as discussed in the previous paragraphs. Let w_f denote the required width of final f , and let W denote the total width of a raw. Then the three observations can be translated into the following model constraints.

*An auxiliary
model*

$$\begin{aligned} y_f &\geq 0, \text{ integer } \forall f & (1) \\ \sum_f w_f y_f &\leq W & (2) \\ 1 - \sum_f \lambda_f y_f &< 0 & (3) \end{aligned}$$

The above model formulation contains a strict inequality and it must be manipulated before using a solver which is based on inequalities. There is one observation that makes it possible to rewrite the above system as a mixed-integer linear programming model. Whenever the term $\sum_f \lambda_f y_f$ is greater than one, the last inequality is satisfied. You could write this term as an objective function to be maximized subject to the first two constraints. Whenever the optimal value of this mathematical program is greater than one, you have found an interesting cutting pattern. Whenever this optimal value is less than

*Transformation
into a MIP
model*

or equal to one, you know that there does not exist a cutting pattern that can improve the objective value of the reduced cutting stock problem expressed as $\sum_f \lambda_f y_f$. This observation results in the following cutting pattern generation model.

Maximize:

$$\sum_f \lambda_f y_f$$

*Cutting pattern
generation
model*

Subject to:

$$\sum_f w_f y_f \leq W$$

$$y_f \geq 0, \text{ integer} \quad \forall f$$

The implementation of this model in AIMMS is straightforward since the λ_f 's are calculated during each solve iteration and can be directly accessed. It is important to allow numerical inaccuracies in the computed shadow prices. For this reason, it is generally advisable to use a small tolerance $\delta > 0$ when verifying whether a new patterns will lead to improvement. The mathematical condition to be verified for progress then becomes

*Allowing
inaccuracies*

$$\sum_f \lambda_f y_f \geq 1 + \delta$$

The value of δ is typically in the order of 10^{-4} . When δ is too small, the overall algorithm may not converge. In that case the cutting pattern generation model produces the same new pattern every time it is solved.

The transformation of the initial auxiliary model into a MIP model is strongly dependent on the property that all objective function coefficients are identical. Without this property, it is not meaningful to translate a strict inequality of the form $c_y - \sum_f \lambda_f y_f < 0$ into an objective function of the form $\sum_f \lambda_f y_f$ as has been done. Without identical coefficients, the stopping criterion in the delayed column generation algorithm is no longer correct. The reason is that when $c_{y^*} - \sum_f \lambda_f y_f^* \geq 0$ holds for an optimal solution y^* (indicating termination), it is still possible that $c_{\hat{y}} - \sum_f \lambda_f \hat{y}_f < 0$ holds for some other solution \hat{y} due to a smaller value of $c_{\hat{y}}$.

*Identical
objective
coefficients*

20.4 Extending the original cutting stock problem

In this section three possible extensions to the original cutting stock model are introduced and subsequently incorporated into a single new cutting stock model.

This section

One extension is to include several types of raws. Each type with its own length. This will result in a large increase in the overall number of cutting patterns to be considered when solving the underlying model.

Multiple types of raws

Another extension is to include the purchase cost of each type of raw. This changes the objective function of the underlying model. Rather than minimizing the number of raws to be used, the objective is to minimize the cost of raws.

Purchase cost

The third extension is to introduce machine capacity restrictions. It is assumed that there is an upper bound on the number of raws of each type that can be cut on the available machines during a fixed period of time. It is assumed that these upper bounds are not dependent on each other.

Capacity limitation

The resulting extended cutting stock problem can be translated into the following mathematical model.

Model formulation

Indices:

r	<i>types of raws</i>
p	<i>cutting patterns</i>
f	<i>finals</i>

Parameters:

c_r	<i>unit cost of raws of type r</i>
d_f	<i>required demand for final f</i>
a_{fpr}	<i>number of finals f in pattern p for raws of type r</i>
k_r	<i>available capacity for raws of type r</i>

Variable:

x_{pr}	<i>number of raws of type r cut with pattern p</i>
----------	--

Minimize:

$$\sum_{p,r} c_r x_{pr}$$

Subject to:

$$\begin{aligned} \sum_{pr} a_{fpr} x_{pr} &\geq d_f && \forall f \\ \sum_p x_{pr} &\leq k_r && \forall r \\ x_{pr} &\geq 0, \text{ integer} && \forall (p, r) \end{aligned}$$

With the extension of multiple raws and varying cost coefficients, it is no longer clear whether the delayed column generation algorithm of the previous section is applicable. The previous auxiliary model, finds a cutting pattern for just a single size of raw, and the contribution of a new cutting pattern is compared to the constant value of 1.

Delayed pattern generation ...

Observe, however, that the cost coefficients are constant for all patterns belonging to a single type of raw. This implies that the idea of cutting pattern generation can still be applied as long as each type of raw is considered separately. The resulting generalized delayed pattern generation algorithm is summarized below.

... can still be applied

```

WHILE progress is being made DO
  Solve cutting stock submodel
  FOR each type of raw DO
    Solve cutting pattern generation model
    IF new cutting pattern will lead to improvement
      THEN add it to cutting stock submodel
  ENDFOR
ENDWHILE

```

As a result of the extra capacity constraints, the condition to check whether a new pattern will lead to improvement needs to be modified. Let π_r denote the shadow price associated with the capacity constraint for raws of type r obtained after solving the cutting stock submodel. Then any cutting pattern y^r produced by the auxiliary pattern generation model for raws of type r will lead to an improvement only if

Capacity constraint modification

$$c_r - \pi_r - \sum_f \lambda_f y_f^r < 0$$

This condition is the reduced cost criterion in the simplex method applied to the extended cutting stock model developed in this section.

Recall from the previous section that the inaccuracies in the shadow price computation need to be taken into account. By again introducing a $\delta > 0$, the above condition can be rewritten as

Allow for inaccuracies

$$\sum_f \lambda_f y_f^r \geq c_r - \pi_r + \delta$$

In the above delayed pattern generation algorithm summary, the auxiliary model is solved for every type of raw r before solving the next cutting stock submodel. An alternative approach is to solve the cutting stock model as soon as one new interesting pattern has been found. You might want to investigate this alternative when the time required to solve the cutting pattern generation model is large relative to the time required to solve the cutting stock submodel.

Alternative solution sequence

Consider three types of raws (600 cm, 800 cm and 1000 cm) and the same four final sizes as in Section 20.2 (140 cm, 310 cm, 360 cm and 450 cm). The corresponding demand for these finals is 100, 300, 500 and 200 respectively. The unit cost and the available capacity associated with each raw type is presented in Table 20.3.

A worked example

Raw	c_r	k_r
600 cm	25	200
800 cm	30	200
1000 cm	40	300

Table 20.3: Raw type data

20.5 Summary

In this chapter a cutting stock problem was translated into a mathematical formulation based on the concept of cutting patterns. Due to the large number of cutting patterns in practical applications, a delayed column generation approach using a cutting stock submodel was introduced. This approach solves an auxiliary integer programming model to produce a single new cutting pattern which is then added to the cutting stock submodel. The auxiliary model has been developed in detail, and the overall solution approach has been outlined. The algorithm can easily be implemented in AIMMS.

Exercises

- 20.1 Implement the cutting stock model described in Section 20.2 using the example data presented in Table 20.1. Write a small procedure in AIMMS to round the optimal linear programming solution using the Largest-In-Least-Empty heuristic.
- 20.2 Implement the delayed cutting pattern generation approach described in Section 20.3 in AIMMS as an iteration between two models. Check whether the optimal solution found is the same as the one found previously.
- 20.3 Implement the extension of the initial cutting stock model, which is described in Section 20.4. Verify that the optimal objective function value equals 15,600 using the example data from Section 20.4.

Chapter 21

A Telecommunication Network Problem

In this chapter you will encounter a capacity utilization problem in a telecommunication network. Traffic in such a network is expressed in terms of calls, and calls are made between a large number of origin-destination pairs during a particular period of time. Calls between origins and destinations can be routed along any path through the network subject to capacity limitations. The objective is to identify bottleneck capacity in the network. In practical applications, the model turns out to be quite large due to the many possible paths that exist between origins and destinations. For that reason a path generating technique is introduced to control the size of the model that is passed to a solver during a sequence of iterations.

This chapter

The telecommunication network model discussed in this chapter can be found in various sources. Two references, spanning a period of almost 30 years, are [Hu69] and [Me98]. The required theory of linear programming and column generation can be found in [Ch83] and in Chapter 19 of this book. In addition, Chapter 20, 'A Cutting Stock Problem', also provides an application in which column generation plays a central role.

References

Linear Program, Network Program, Simplex Method, Column Generation, Auxiliary Model, Customized Algorithm, Mathematical Derivation, Worked Example.

Keywords

21.1 Problem description

This section provides a brief introduction to the terminology and concepts used in the telecommunication network problem described in this chapter. The problem itself is summarized towards the end of this section.

This section

In a telecommunication network, transmission lines are used to carry traffic in the form of calls. These lines form the link between switch-stations. Traffic is routed from one switch-station to the next until it reaches its destination. For the sake of simplicity both the origin and destination of a call are assumed to be switch-stations.

*Network
configuration*

Each switch-station is represented as a *node* in the network, and each link between any two nodes is represented as an *arc*. The maximum amount of traffic that can go through a switch-station during a specific period of time will be referred to as *node capacity*. A similar definition holds for *arc capacity*. A route from origin to destination through the network is a *path*.

Nodes, arcs and paths

Traffic for a particular origin-destination pair can be split and subsequently recombined at any node in the network. This flexibility in routing traffic allows for efficient use of the entire network. The implication of flexible routing for the model to be developed, is that all possible paths between an origin and a destination will need to be considered.

Flexible routing

Assume that the amount of traffic between all origin-destination pairs for a particular period is known, and that the capacity for all switch-stations and transmission lines is provided. The problem that will be addressed is the bottleneck identification problem. In this problem traffic is routed along paths in the network so that traffic requirements are met. In addition, the bottleneck in the network is identified by finding that arc or node with the largest percentage use of the available capacity.

Problem summary

The bottleneck identification problem can be viewed as a strategic network design problem. In a network there are often bottlenecks that must be alleviated through redesign either by adding new switch-stations or new transmission lines, or by adding capacity to any of the existing facilities. The problem in this chapter represents a simplification, because it does not consider such practical matters as network robustness and reliability under (uncertain) traffic regimes. Nevertheless, the model developed next is of interest, as it forms a basis for several extensions.

Network design

21.2 Bottleneck identification model

In this section you will encounter a compact arc-path formulation of the bottleneck identification problem described in the previous section. Initially, it is assumed that all possible paths between origin-destination pairs are enumerated explicitly. This assumption will be relaxed in the next section where paths are generated one-at-a-time as needed.

This section

Figure 21.1 depicts a simplified Dutch telecommunication network containing 6 nodes and 12 (bi-directional) arcs. In this example, it is assumed that there is traffic between all possible pairs of (origin-destination) nodes, and that each node and arc has a limited capacity. Even in this small example, the number of undirected paths is quite large (namely 377), and only a few of them are listed in Table 21.1.

Example

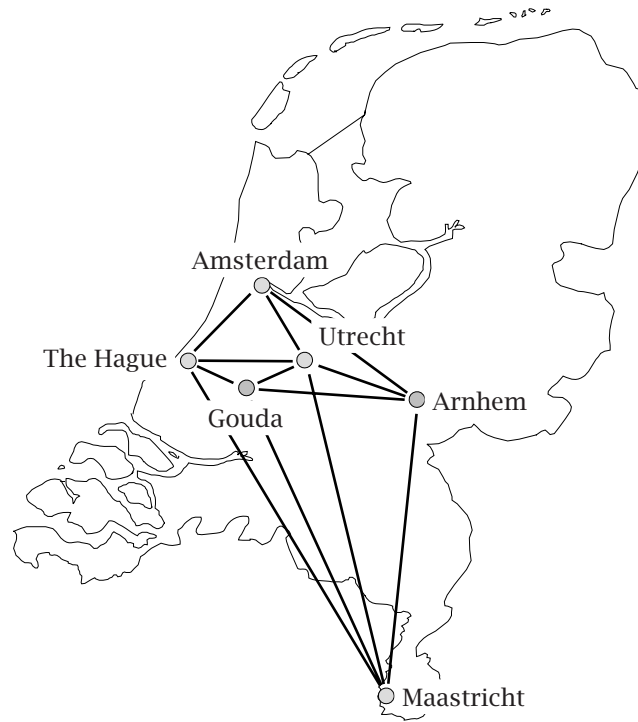


Figure 21.1: A Dutch Telecommunication Network

The verbal model is expressed in terms of network terminology for reasons of conciseness and ease of recall. The interpretation in terms of transmission lines and switch-stations is straightforward. Note that the use of any arc or node (as referred to in the problem summary) is expressed as a fraction of the available capacity.

Verbal model description

Minimize: *maximum fraction of either arc use or node use,*

Subject to:

- *for each origin-destination pair: total traffic along paths connecting this pair is equal to the required amount of traffic,*
- *for each arc in the network: total traffic that uses that arc is equal to a fraction of the available arc capacity,*
- *for each node in the network: total traffic that uses that node is equal to a fraction of the available node capacity,*
- *for each arc in the network: capacity use is less than or equal to the maximum fraction,*
- *for each node in the network: capacity use is less than or equal to the maximum fraction.*

The mathematical description is slightly more complicated due to the various indices that play a role.

origin	destination	path
Amsterdam	Maastricht	Amsterdam - The Hague - Maastricht
-	-	Amsterdam - The Hague - Utrecht - Maastricht
-	-	Amsterdam - The Hague - Utrecht - Gouda - Maastricht
-	-	Amsterdam - The Hague - Utrecht - Gouda - Arnhem - Maastricht
-	-	Amsterdam - The Hague - Utrecht - Arnhem - Maastricht
-	-	Amsterdam - The Hague - Utrecht - Arnhem - Gouda - Maastricht
-	-	Amsterdam - The Hague - Gouda - Maastricht
-	-	Amsterdam - The Hague - Gouda - Utrecht - Maastricht
-	-	Amsterdam - The Hague - Gouda - Utrecht - Arnhem - Maastricht
-	-	Amsterdam - The Hague - Gouda - Arnhem - Maastricht
-	-	Amsterdam - The Hague - Gouda - Arnhem - Utrecht - Maastricht
-	-	Amsterdam - Utrecht - Maastricht
-	-	Amsterdam - Utrecht - The Hague - Maastricht
...		

Table 21.1: Paths between Amsterdam and Maastricht

The following symbols will be used for the mathematical description of the bottleneck identification model.

Mathematical description

Indices:

n	<i>nodes</i>
a	<i>arcs</i>
o, d	<i>origin and destination nodes</i>
p	<i>paths</i>

Set:

S_{od}	<i>all paths between origin o and destination d</i>
----------	---

Parameters:

A_{ap}	<i>incidence: arc a is on path p</i>
B_{np}	<i>incidence: node n is on path p</i>
C_a	<i>capacity of arc a</i>
C_n	<i>capacity of node n</i>
D_{od}	<i>required traffic between origin o and destination d</i>

Variables:

x_p	<i>traffic along path p</i>
f_a	<i>fraction of available capacity of arc a</i>
f_n	<i>fraction of available capacity of node n</i>
M	<i>maximum fraction of either arc or node capacity</i>

Traffic requirements are typically specified for only a subset of all origin-destination pairs. Such traffic may be routed along any path. As the set S_{od} contains all paths for each (o, d) pair, specifying the corresponding traffic requirement is straightforward.

Traffic requirement

$$\sum_{p \in S_{od}} x_p = D_{od} \quad \forall (o, d) \mid D_{od} > 0$$

Note that the above traffic requirement constraint is only defined when the required amount of traffic D_{od} is greater than zero. Enforcing this condition is one way to reduce the number of constraints. A second and practically more effective reduction in the number of constraints is to add the requirements D_{od} and D_{do} , and to consider only traffic requirement constraints for $o < d$.

Reducing the number of constraints

An arc a can be on several paths that are selected to meet the traffic requirement for various (o, d) pairs. The total (bi-directional) amount of traffic along such an arc, however, is limited by the arc's capacity. Rather than specifying a hard capacity constraint for each arc, an additional nonnegative variable f_a is introduced to indicate the fraction of capacity used for that arc. Such a fraction should, of course, be less than or equal to one. By leaving it unrestricted from above, however, it measures the fraction of capacity sufficient to meet traffic requirements.

Arc capacity utilization

$$\sum_p A_{ap} x_p = f_a C_a \quad \forall a \mid C_a > 0$$

Note that the above arc capacity utilization constraint is only defined when the corresponding arc capacity C_a is greater than zero.

A node n can also be on several paths selected to meet traffic requirements for various (o, d) pairs. Just like arcs, nodes also have limited capacity. An additional nonnegative variable f_n is introduced to measure the fraction of capacity sufficient to meet traffic requirements.

Node capacity utilization

$$\sum_p B_{np} x_p = f_n C_n \quad \forall n \mid C_n > 0$$

Note that the above node capacity utilization constraint is only defined when the corresponding node capacity C_n is greater than zero.

Identifying the bottleneck capacity is now straightforward to model. Consider the maximum capacity utilization fraction M , and let all fractions of capacity utilization in the network be less than or equal to this fraction. By minimizing M , the optimal solution will identify one or more critical capacities. Note that the underlying linear program will always be feasible, because there is no effective capacity limitation (i.e. limit on M).

Identifying bottleneck capacity

$$\begin{aligned} \text{Minimize:} \quad & M \\ \text{Subject to:} \quad & f_a \leq M \quad \forall a \mid C_a > 0 \\ & f_n \leq M \quad \forall n \mid C_n > 0 \end{aligned}$$

An optimal $M \leq 1$ indicates that existing capacities can be utilized to meet all traffic requirements. An optimal $M > 1$ implies that capacity expansion is needed.

The following mathematical statement summarizes the model.

Model summary

Minimize:

M

Subject to:

$$\begin{aligned}
 \sum_{p \in S_{od}} x_p &= D_{od} & \forall (o, d) \mid o < d, D_{od} > 0 \\
 \sum_p A_{ap} x_p &= f_a C_a & \forall a \mid C_a > 0 \\
 \sum_p B_{np} x_p &= f_n C_n & \forall n \mid C_n > 0 \\
 0 &\leq x_p & \forall p \\
 0 &\leq f_a \leq M & \forall a \mid C_a > 0 \\
 0 &\leq f_n \leq M & \forall n \mid C_n > 0
 \end{aligned}$$

21.3 Path generation technique

The number of paths in practical applications is too large to enumerate, and thus the corresponding linear program cannot be generated in its entirety. This section develops a path generating scheme which avoids the complete enumeration of all paths. The approach resembles the column generating technique described in Chapter 20, except that the columns now correspond to paths.

This section

If all columns of a linear program cannot be generated prior to solving, they need to be generated 'as needed' during the solution phase. This requires insight into the construction of a column together with knowledge of the relevant shadow prices to make sure that any generated column will improve the objective function value of the underlying linear program. In the case of the bottleneck identification model, you need to consider the path-related variable x_p and see whether there is a new column p that may lead to a better solution of the overall model.

*Dynamic
column
generation*

Each new column with coefficients corresponding to a new x_p variable has zero-one entries in the first three symbolic constraints of the above bottleneck identification model. The entries associated with the first constraint are all 0 except for a single 1 corresponding to a particular (o, d) pair. The entries of the new column associated with the second constraint are 1 or 0 dependent on whether an arc is on the new path or not. Let z_a be 1 if arc a is on the path, and 0 otherwise. Similarly, the entries associated with the third constraint are 1 or 0 dependent on whether a node is on the new path or not. Let h_n be 1 if node n is on the path, and 0 otherwise.

*A typical
column*

The symbols z_a and h_n can be viewed as variables characterizing the path-related coefficients of each new column to be determined. The shadow prices corresponding to the first three constraints are needed to decide on a new column to be added to the bottleneck identification model. They are:

Column entry condition ...

$$\begin{aligned}\lambda_{od} & \text{ for the traffic requirement constraint} \\ \mu_a & \text{ for the arc capacity utilization constraint} \\ \theta_n & \text{ for the node capacity utilization constraint}\end{aligned}$$

You may want to verify that the condition below describes the situation in which a path p may contribute to the optimal solution value of the underlying bottleneck identification model. This condition, which is equivalent to the column entry condition in the simplex method for a minimization model, is similar to the reduced cost criterion explained in Chapters 19 and 20.

$$0 - \lambda_{od} - \sum_a \mu_a z_a - \sum_n \theta_n h_n < 0$$

By considering only those values of z_a and h_n that together determine a path, and by minimizing the left side of the above inequality over all particular (o, d) pairs, you obtain the best path in terms of the reduced cost criterion. If the minimum value is strictly less than zero, you have found a path between an o and a d that will improve the underlying linear program. If this minimum is greater than or equal to zero, then there does not exist a new path that will improve the linear program. This observation leads to the idea to employ a path-finding model as an auxiliary model to identify a new path p satisfying the above column entry condition. In this chapter a path-finding linear programming formulation has been selected. A shortest-path approach based on Dijkstra's algorithm could also have been employed.

... leads to minimization

The following auxiliary model is selected to play a role in the path generating approach proposed in this chapter. The notation in this auxiliary path-finding model is based on the concept of directed arcs for reason of convenience. Its relationship to the column entry condition above will be discussed in subsequent paragraphs.

Auxiliary path-finding model

Sets:

$$\begin{aligned}N & \text{ nodes} \\ I \subset N & \text{ intermediate nodes}\end{aligned}$$

Indices:

$$i, j \quad \text{nodes}$$

Element Parameters:

$$\begin{aligned}\text{orig} & \text{ originating node} \\ \text{dest} & \text{ destination node}\end{aligned}$$

Numerical Parameter:

$$k_{ij} \quad \text{objective coefficient for arc } (i, j)$$

Variable:

$$y_{ij} \quad 1 \text{ if arc } (i, j) \text{ is on optimal path, } 0 \text{ otherwise}$$

When $k_{ij} > 0$ and the set I contains all nodes except the originating node *orig* and the destination node *dest*, you may verify that the following model with network constraints determines a best path from *orig* to *dest* expressed in terms of indices i and j .

Minimize:

$$\sum_{(ij)} k_{ij} y_{ij}$$

Subject to:

$$\sum_j y_{ji} - \sum_j y_{ij} = \begin{cases} -1 & \text{if } i = \text{orig} \\ 0 & \text{if } i \in I \\ 1 & \text{if } i = \text{dest} \end{cases}$$

The above model can be solved using a specialized algorithm. It can also be solved as a linear program with constraints $0 \leq y_{ij} \leq 1$, which will result in a zero-one optimal solution due to the network property of the model (see e.g. Chapter 5).

Not all y_{ij} variables need to be considered in the above model formulation. First of all, all variables with $i = j$ are superfluous, as no such variable will be in the optimal solution with $k_{ii} > 0$. Similarly, all variables y_{ij} with $i = \text{dest}$ or with $j = \text{orig}$ are also superfluous. As a result, the number of relevant y_{ij} variables is equal to $|N|(|N| - 3) + 3$ for $|N| \geq 2$. Throughout the remainder of this section the restricted domain of y_{ij} is implied.

Restricting the y_{ij} domain

It is not immediately obvious how the above auxiliary path-finding model can be used to minimize the expression

Required translation

$$-\lambda_{od} - \sum_a \mu_a z_a - \sum_n \theta_n h_n$$

The required translation turns out to be fairly straightforward, but demands some attention. In essence, the z and h terms must be translated into the y_{ij} decision variables, while the other terms must be translated into the objective function coefficients k_{ij} .

The term λ_{od} is a constant once a particular (o, d) pair is selected. Therefore, this term can be ignored when constructing the objective function coefficients of the auxiliary model for a particular pair.

The λ terms

An arc a corresponds to a tuple (i, j) as well as a tuple (j, i) . Therefore, a possible translation is to write $z_a = y_{ij} + y_{ji}$ with $\mu_{ij} = \mu_{ji} = \mu_a$. Such a translation is permitted, because at most one of the y_{ij} and y_{ji} values can be equal to 1 in an optimal solution of the auxiliary model when $k_{ij} > 0$.

The μ and z terms

Let $(o, d) = (\text{orig}, \text{dest})$. You can express the relationship between h_n and y_{ij} as follows. First of all, $h_o = \sum_j y_{oj}$ and $h_d = \sum_i y_{id}$. Then for all intermediate nodes, either $h_{i \in I} = \sum_j y_{ij}$ or $h_{i \in I} = \sum_j y_{ji}$. The term θ_n needs not be modified, and can be used directly in the construction of the objective function coefficients k_{ij} .

The θ and h terms

The column entry condition without the constant term λ_{od} is

$$-\sum_a \mu_a z_a - \sum_n \theta_n h_n$$

Rewriting the column entry condition

and can now be rewritten in terms of the y_{ij} variables of the auxiliary model in one of two ways depending on the expression for $h_{i \in I}$. Either

$$-\sum_{(ij)} \mu_{ij} y_{ij} - \theta_o \sum_j y_{oj} - \sum_{i \in I} \theta_i \sum_j y_{ij} - \theta_d \sum_i y_{id}$$

or

$$-\sum_{(ij)} \mu_{ij} y_{ij} - \theta_o \sum_j y_{oj} - \sum_{i \in I} \theta_i \sum_j y_{ji} - \theta_d \sum_i y_{id}$$

By carefully combining terms in the above two expressions and considering only the restricted (i, j) domain, the corresponding values of k_{ij} can be written either as

Determining the coefficients k_{ij}

$$\begin{aligned} k_{oj} &:= -\mu_{oj} - \theta_o & \forall j \neq d \\ k_{ij} &:= -\mu_{ij} - \theta_i & \forall (i, j), i \neq o, j \neq d \\ k_{id} &:= -\mu_{id} - \theta_i - \theta_d & \forall i \end{aligned}$$

or as

$$\begin{aligned} k_{oj} &:= -\mu_{oj} - \theta_j - \theta_o & \forall j \\ k_{ij} &:= -\mu_{ij} - \theta_i & \forall (i, j), i \neq o, j \neq d \\ k_{id} &:= -\mu_{id} - \theta_d & \forall i \neq o \end{aligned}$$

The values μ and θ are typically zero when the corresponding capacity constraints in the bottleneck identification model are not critical. Once a capacity constraint has an associated capacity fraction value equal to the critical value M , then the corresponding μ or θ value will be strictly less than 0, causing the corresponding k_{ij} to be greater than 0. You may verify this by applying the definition of a shadow price as described in Section 4.2 to the capacity constraints after moving all variable terms to the left-hand side. By adding a sufficiently small $\epsilon > 0$ to all permitted values of k_{ij} , the requirement of $k_{ij} > 0$ is automatically satisfied and the optimal solution of the auxiliary path finding model is guaranteed to be a proper path between o and d without any

Forcing $k_{ij} > 0$

zero-valued subtours. A recommended choice for ϵ is the smallest positive initial k_{ij} value divided by the total number of positive initial k_{ij} values.

Once the optimal solution of the auxiliary path finding model has been determined, a check must be made to verify whether the newly generated path should be added to the bottleneck identification model. This check is nothing more than verifying whether the column entry condition, described in terms of the y_{ij} values, is satisfied. You may verify that the following expression forms the correct check.

*Correcting k_{ij}
afterwards*

$$-\lambda_{od} + \sum_{ij} (k_{ij} - \epsilon) y_{ij} < 0$$

Recall that the values of k_{ij} could have been determined in one of two ways, but both sets are appropriate when checking the above condition.

The translation of the column entry condition into the terminology of the auxiliary path-finding model is now complete. The following algorithmic skeleton loosely summarizes the approach to solve the bottleneck identification model with path generation. Initialization (without initial shadow prices) is accomplished by setting the objective function coefficients k_{ij} equal to 1, leading to the computation of the shortest path with the fewest number of intermediate nodes between every possible (o, d) pair. These paths determine the initial entries of the parameters A_{ap} and B_{np} in the bottleneck identification model (referred to as 'main model' below). In addition, each such shortest path is also a single initial element of the set S_{od} .

*Path generation
algorithmic
skeleton*

```

FOR all origin-destination pairs DO
  Solve path-finding model
  Add shortest path to parameters in main model
ENDFOR

WHILE at least one new path has been added DO
  Solve current bottleneck model
  FOR all origin-destination pairs DO
    Solve path-finding model
    IF new path contributes
      THEN add path to parameters in main model
    ENDIF
  ENDFOR
ENDWHILE

```

An implementation of this algorithm in AIMMS is not entirely straightforward, and does require some carefully constructed set and parameter manipulations to update the input of both the path-finding model and the bottleneck identification model. Special care is also required when implementing the column entry condition in order to avoid the repeated generation of a single path. The powerful language features of AIMMS, however, allow for a one-to-one translation of the notation used in this chapter.

In the above proposed algorithm a new path between all possible (o, d) pairs is computed prior to solving the next bottleneck capacity model. By restricting yourself to only those (o, d) pairs with paths along critical nodes and/or arcs (i.e. nodes and/or arcs with maximum capacity utilization fractions), the computational effort to find new paths prior to solving the next bottleneck identification model can be significantly reduced. The suggestion in this paragraph is only one of several possibilities to reduce computational effort.

*Reducing
computational
efforts*

21.4 A worked example

The traffic requirements (in terms of calls) between all possible origins and destination pairs are presented in Table 21.2.

Traffic data

d o	Amsterdam	Utrecht	The Hague	Gouda	Arnhem	Maastricht
Amsterdam		55	95	20	30	45
Utrecht	90		50	10	15	20
The Hague	85	45		15	10	30
Gouda	35	25	35		10	15
Arnhem	45	15	20	5		35
Maastricht	60	25	40	10	30	

Table 21.2: Traffic requirements between all origin-destination pairs

The node and arc capacities for the network in this example are provided in Table 21.3.

Capacity data

	Arc capacities						Node capacities
	Amsterdam	Utrecht	The Hague	Gouda	Arnhem	Maastricht	
Amsterdam		360	300		240		490
Utrecht			360	60	90	120	340
The Hague				90		180	400
Gouda					40	120	220
Arnhem						210	280
Maastricht							340

Table 21.3: Arc and node capacities in the network

The initial number of shortest paths between all possible (o, d) pairs is 15 ($= \binom{6}{2}$). These paths were generated by the shortest-path procedure, and are summarized in Table 21.4. Note that only the arc-path incidences are provided. An explicit path description in terms of nodes can always be derived from the arc names.

Initial paths

Arcs	Initial paths														
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
(Amsterdam,Utrecht)	×		×		×										
(Amsterdam,The Hague)		×										×			
(Amsterdam,Arnhem)				×								×			
(Utrecht,The Hague)						×									
(Utrecht,Gouda)			×				×								
(Utrecht,Arnhem)								×							
(Utrecht,Maastricht)					×				×						
(The Hague,Gouda)										×					
(The Hague,Maastricht)												×			
(Gouda,Arnhem)													×		
(Gouda,Maastricht)														×	
(Arnhem,Maastricht)															×

Table 21.4: Initial paths generated by shortest-path algorithm

After continuing the path generating procedure, a total of 9 additional paths were generated. The observed bottleneck fraction was reduced from 1.500 (based on the initial paths only) to a value of 1.143 (based on both the initial and additional paths). The 9 additional paths are summarized in Table 21.5. It is of interest to note that the bottleneck identification model was solved 5 times in total to obtain these results. The number of times the auxiliary path-finding model was solved, amounted to 90.

*Additional
paths*

Arcs	Additional paths								
	16	17	18	19	20	21	22	23	24
(Amsterdam,Utrecht)		×							
(Amsterdam,The Hague)		×	×	×					
(Amsterdam,Arnhem)	×							×	×
(Utrecht,The Hague)									
(Utrecht,Gouda)						×			
(Utrecht,Arnhem)						×			
(Utrecht,Maastricht)									
(The Hague,Gouda)		×							×
(The Hague,Maastricht)			×	×		×			
(Gouda,Arnhem)	×								
(Gouda,Maastricht)			×		×	×	×		×
(Arnhem,Maastricht)					×		×	×	×

Table 21.5: Additional paths generated by path-finding algorithm

When the algorithm was modified and restricted to path generation for critical (o, d) pairs only, 4 instead of 9 additional new paths were generated. In this case, the bottleneck identification model was solved 4 instead of 5 times, and the auxiliary path-finding model was solved 35 instead of 90 times. As expected, the bottleneck fraction was again reduced to 1.143.

*Restricting to
critical paths*

21.5 Summary

In this chapter you have encountered a bottleneck identification problem in a telecommunication network, together with two model formulations. The first formulation assumes that the input to the model is based on all possible paths between origins and destinations. The second (more practical) formulation does not explicitly enumerate all possible paths, but generates them only when they can contribute to the identification of the bottleneck capacity. An auxiliary model is used to generate these paths. The overall model can be used to decide how to modify existing capacity to meet changing traffic requirements, or how existing traffic is to be routed through the network.

Exercises

- 21.1 Consider the path generation technique presented in Section 21.3 and implement the bottleneck identification model of Section 21.2 using the example data contained in Tables 21.2 and 21.3. Verify whether the optimal solution found with AIMMS coincides with the one presented in Tables 21.4 and 21.5.
- 21.2 Investigate whether there is any difference in the optimal solution due to the choice of the k_{ij} coefficient values developed in Section 21.3.
- 21.3 Adjust your search algorithm to examine only those (o, d) pairs with paths along critical nodes and/or arcs. Verify for yourself how much the amount of computational work has been reduced due to this modification.

Chapter 22

A Facility Location Problem

This chapter considers the problem of selecting *distribution centers* along with their associated customer zones. For small and medium-sized data sets, the mathematical model is a straightforward mixed-integer programming formulation and can easily be solved with standard solvers. However for large data sets, a decomposition approach is proposed. This chapter explains the Benders' decomposition technique and applies it to the facility location problem.

This chapter

The example in this chapter is based on "Multicommodity Distribution System Design by Benders Decomposition" ([Ge74]) by Geoffrion and Graves.

References

Integer Program, Mathematical Reformulation, Mathematical Derivation, Customized Algorithm, Auxiliary Model, Constraint Generation, Worked Example.

Keywords

22.1 Problem description

A commonly occurring problem in distribution system design is the optimal location of intermediate distribution centers between production plants and customer zones. These intermediate facilities (temporarily) store a large variety of commodities that are later shipped to designated customer zones.

*Distribution
system design*

Consider the situation where several commodities are produced at a number of plants with known production capacities. The demands for each commodity at a number of customer zones are also known. This demand is satisfied by shipping via *intermediate distribution centers*, and for reasons of administration and efficiency, each customer zone is assigned exclusively to a single distribution center. For each center there is a lower as well as an upper limit on the total throughput (of all commodities). There is also a fixed rental charge and a per unit throughput charge associated with each distribution center. In addition, there is a variable unit cost of shipping a commodity from a plant to a customer zone through a distribution center. This cost usually includes the unit production cost.

*Basic problem
in words*

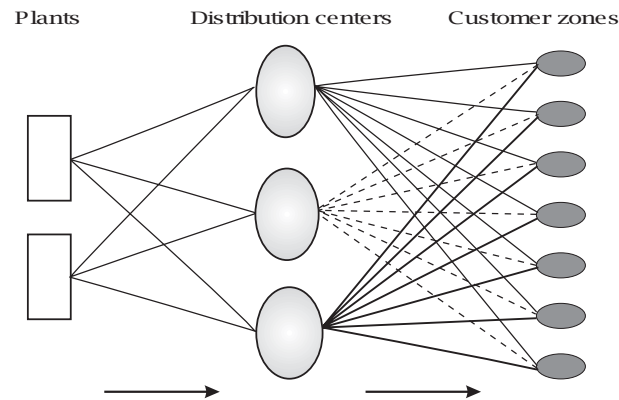


Figure 22.1: Commodity distribution scheme

The facility location problem is shown schematically in Figure 22.1. It has the property that the main decisions are of type yes/no. The problem is to determine which distribution centers should be selected, and what customer zones should be served by the selected distribution centers. The optimum solution is clearly dependent on the pattern of transportation flows for all commodities. It is assumed that the time frame under consideration is sufficiently long to motivate good decision making.

Decisions to be made

The decisions described in the previous paragraphs are to be made with the objective to meet the given demands at minimum total distribution and production cost, subject to plant capacities and distribution center throughput requirements.

Cost minimization

This chapter formulates and solves the above problem description. However in real-world applications, there may be additional constraints which require some specialized formulation. Some possibilities are mentioned below.

Problem extensions

- The throughput capacity in a particular distribution center can be treated as a decision variable with an associated cost.
- Top management could impose an a priori limit on the number of distribution centers, or express preferences for particular logical combinations of such centers (not A unless B, not C and D, etc.).
- Similarly, there could be an a priori preference for certain logical combinations of customer zones and distribution centers (if Center A is open, then Zone 2 must be assigned, etc.).

- If distribution centers happen to share common resources or facilities, there could be joint capacity constraints.

You are referred to Chapter 7 for ideas on how to model these special logical conditions.

22.2 Mathematical formulation

This section presents the mathematical description of the facility location problem discussed in the previous section.

This section

The objective and the constraints are described in the following qualitative model formulation.

*Qualitative
model
description*

Minimize: *total production and transport costs,*

Subject to:

- *for all commodities and production plants: transport must be less than or equal to available supply,*
- *for all commodities, distribution centers and customer zones: transport must be greater than or equal to required demand,*
- *for all distribution centers: throughput must be between specific bounds, and*
- *for all customer zones: supply must come from exactly one distribution center.*

The following notation will be used in this chapter:

Notation

Indices:

c	<i>commodities</i>
p	<i>production plants</i>
d	<i>distribution centers</i>
z	<i>customer zones</i>

Parameters:

S_{cp}	<i>supply (production capacity) of commodity c at plant p</i>
D_{cz}	<i>demand for commodity c in customer zone z</i>
\overline{M}_d	<i>maximum throughput at distribution center d</i>
\underline{M}_d	<i>minimum throughput at distribution center d</i>
R_d	<i>per unit throughput charge at distribution center d</i>
F_d	<i>fixed cost for distribution center d</i>
K_{cpdz}	<i>variable cost for production and shipping of commodity c, from plant p via distribution center d to customer zone z</i>

Variables:

x_{cpdz}	<i>nonnegative amount of commodity c shipped from plant p via distribution center d to customer zone z</i>
------------	--

v_d	<i>binary to indicate selection of distribution center d</i>
y_{dz}	<i>binary to indicate that customer zone z is served by distribution center d</i>

The supply constraint specifies that for each commodity c and each production plant p , the total amount shipped to customer zones via distribution centers cannot be more than the available production capacity,

Supply constraint

$$\sum_{dz} x_{cpdz} \leq S_{cp} \quad \forall c, p$$

The demand constraint specifies that the demand for each commodity c in each zone z should be supplied by all plants, but only through the chosen distribution center y_{dz} ,

Demand constraint

$$\sum_p x_{cpdz} \geq D_{cz} y_{dz} \quad \forall c, d, z$$

The throughput constraints make sure that for each distribution center d the total volume of commodities to be delivered to its customer zones remains between the minimum and maximum allowed throughput,

Throughput constraints

$$\underline{M}_d v_d \leq \sum_{cpz} x_{cpdz} = \sum_{cz} D_{cz} y_{dz} \leq \overline{M}_d v_d \quad \forall d$$

The allocation constraint ensures that each customer zone z is allocated to exactly one distribution center d .

Allocation constraint

$$\sum_d y_{dz} = 1 \quad \forall z$$

The objective function that is to be minimized is essentially the addition of production and transportation costs augmented with the fixed and variable charges for distribution centers and the throughput of commodities through these centers.

Objective function

$$\text{Minimize: } \sum_{cpdz} K_{cpdz} x_{cpdz} + \sum_d [F_d v_d + R_d \sum_{cz} D_{cz} y_{dz}]$$

22.3 Solve large instances through decomposition

The facility location problem can be solved for small to medium sized data sets using any of the mixed integer programming solvers that are available through AIMMS. However, its solution process is based on a branch-and-bound approach and this can sometimes be improved if you add some constraints.

Black box approach

These constraints are redundant for the integer formulation but tighten the associated relaxed linear program solved at each node of the underlying branch-and-bound tree.

Two examples of such redundant constraints are:

Redundant constraints

$$\begin{aligned} y_{dz} &\leq v_d & \forall d, z, \text{ and} \\ \sum_d v_d &\leq L \end{aligned}$$

where L is a heuristically determined upper limit on the number of distribution centers to be opened (based on total demand). For your application, you may want to test if adding these constraints does indeed improve the solution process. In general, the benefit increases as the data set becomes larger.

In some practical applications, it is not unusual for the number of commodities and customer zones to be in the order of 100's to 1000's. Under these conditions, it is possible that the internal memory required by the solver to hold the initial data set is insufficient. If there is enough memory for the solver to start the underlying branch-and-bound solution process, the number of nodes to be searched can be extremely large, and inefficient search strategies (such as depth-first search) may be required to keep the entire search tree in memory.

Large instances

When your model uses an extremely large data set, you may consider re-examining your approach to the problem. One option is to decompose the problem into several smaller *subproblems* that are solved sequentially rather than simultaneously. The next section explains one such approach, namely Benders' decomposition. The technique is a powerful algorithmic-based approach and its application to solve large instances of the facility location problem will be detailed.

Decomposition

22.4 Benders' decomposition with feasible subproblems

This section presents the mathematical description of Benders' decomposition for the case with feasible subproblems. It is based on an abstract model that has been partitioned into an *easy* linear portion and a *difficult* nonlinear/integer portion. Once you understand the underlying decomposition theory plus the basic rules for writing dual linear programs described in the next section, you will be able to apply the Benders' decomposition approach to the facility location problem.

This section

Consider the following minimization problem, which is referred to as $P(x, y)$:

Initial problem
 $P(x, y)$

Minimize:

$$c^T x + f(y)$$

Subject to:

$$Ax + F(y) = b$$

$$x \geq 0$$

$$y \in Y$$

with $A \in \mathbb{R}^{m \times n}$, x and $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $y \in Y \subset \mathbb{R}^p$. Here, $f(y)$ and $F(y)$ may be nonlinear. Y can be a discrete or a continuous range.

First, it is important to observe that for a fixed value of $y \in Y$ the problem becomes a linear program in terms of x . This is represented mathematically as $P(x|y)$. Next, it is assumed that $P(x|y)$ has a finite optimal solution x for every $y \in Y$. This may seem to be a rather restrictive assumption, but in most real-world applications this assumption is already met or else you can modify Y in such a way that the assumption becomes valid.

Feasible subproblems
 $P(x|y)$

The expression for $P(x, y)$ can be written in terms of an equivalent nested minimization statement, $P_1(x, y)$:

Equivalent reformulation
 $P_1(x, y)$

$$\min_{y \in Y} \{f(y) + \min_x \{c^T x \mid Ax = b - F(y), \quad x \geq 0\}\}$$

This statement can be rewritten by substituting the dual formulation of the inner optimization problem (see Section 22.6), to get an equivalent formulation, $P_2(u, y)$:

Equivalent reformulation
 $P_2(u, y)$

$$\min_{y \in Y} \{f(y) + \max_u \{[b - F(y)]^T u \mid A^T u \leq c\}\}$$

The main advantage of the latter formulation is that the constraint set of the inner problem is independent of y . Furthermore, the optimal solution of the inner maximization problem is finite because of the explicit assumption that $P(x|y)$ has a finite optimal solution for every $y \in Y$. Such an optimal solution will always be at one of the *extreme points* $u \in U$. Therefore, the following equivalent formulation, $P_3(u, y)$, may be obtained:

Extreme point reformulation
 $P_3(u, y)$

$$\min_{y \in Y} \{f(y) + \max_{u \in U} [b - F(y)]^T u\}$$

This nested formulation of $P_3(x, y)$ can finally be re-written as a single minimization problem which is referred to as the *full master problem*, $P_4(y, m)$:

*Full master
problem
 $P_4(y, m)$*

Minimize:

$$f(y) + m$$

Subject to:

$$\begin{aligned} [b - F(y)]^T u &\leq m & u \in U \\ y &\in Y \end{aligned}$$

In the full master problem it is important to observe that there is one constraint for each extreme point. It is true that there may be an enormous number in a problem of even moderate size. However, only a small fraction of the constraints will be binding in the optimal solution. This presents a natural setting for applying an iterative scheme in which a master problem begins with only a few (or no) constraints while new constraints are added as needed. This constraint generation technique is dual to the column generation scheme described in Chapter 20.

*Solve $P(x, y)$
iteratively*

From the full master problem, it is possible to define a *relaxed master problem* $M(y, m)$ which considers a subset B of the constraints U .

*The relaxed
master problem
 $M(y, m)$*

Minimize:

$$f(y) + m$$

Subject to:

$$\begin{aligned} [b - F(y)]^T u &\leq m & u \in B \\ y &\in Y \end{aligned}$$

where B is initially empty and m is initially 0.

The Benders subproblem is $S(u|y)$, which solves for an extreme point u given a fixed value of $y \in Y$, can be written as the following maximization problem:

*The subproblem
 $S(u|y)$*

Maximize:

$$[b - F(y)]^T u$$

Subject to:

$$A^T u \leq c$$

with $u \in \mathbb{R}^m$. $S(u|y)$ has a finite optimal solution, because of the assumption that $P(x|y)$ has a finite optimal solution for every $y \in Y$.

Figure 22.2 presents a flowchart of Benders' decomposition algorithm for the case when all subproblems are feasible.

A flowchart of Benders' decomposition

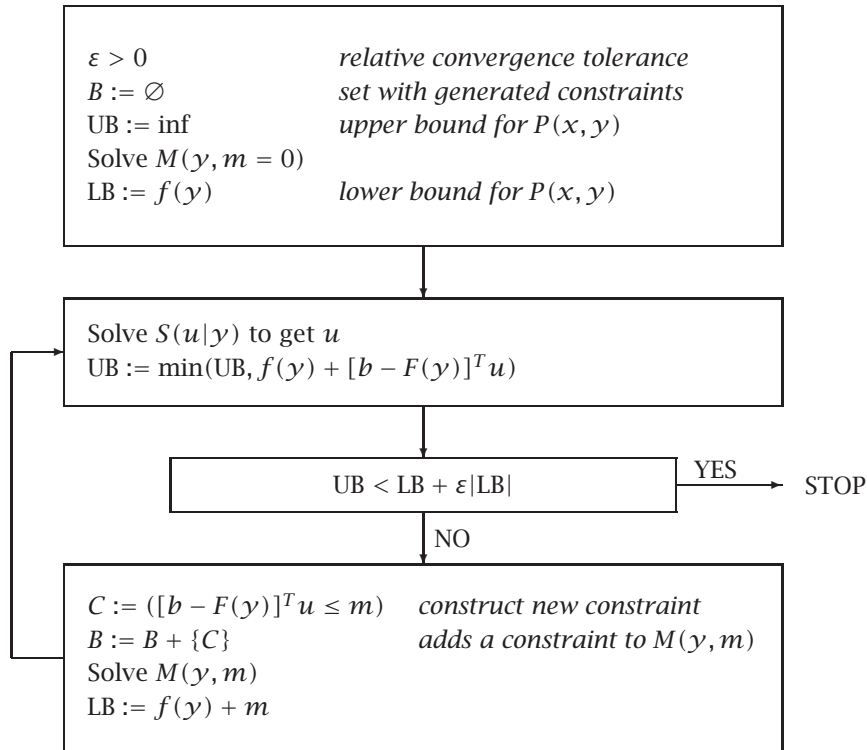


Figure 22.2: Benders' decomposition algorithm flowchart

Summarizing *Benders' decomposition* algorithm in words, the subproblem is solved for u given some initial $y \in Y$ determined by the master. Next, there is a simple test to determine whether a constraint involving u must be added to the master. If so, the master is solved to produce a new y as input to the subproblem which is solved again. This process continues until optimality (within a tolerance level) can be concluded.

The iterative process in words

Since B is a subset of U , the optimal value of the objective function of the relaxed master problem $M(y, m)$ is a lower bound on the optimal value of the objective function of the full master problem $P_4(y, m)$ and thus of $P(x, y)$. Each time a new constraint is added to the master, the optimal value of its objective function can only increase or stay the same.

Increasing lower bounds

The optimal solution u plus the corresponding value of y of the subproblem $S(u|y)$, when substituted in the constraints of $P_3(u, y)$, produces an upper bound on the optimal value of $P_3(u, y)$ and thus of $P(x, y)$. The best upper bound found during the iterative process, can only decrease or stay the same.

*Decreasing
upper bounds*

As soon as the lower and upper bounds of $P(x, y)$ are sufficiently close, the iterative process can be terminated. In practice, you cannot expect the two bounds to be identical due to numerical differences when computing the lower and upper bounds. It is customary to set a relative sufficiently small tolerance typically of the order 1 to $\frac{1}{100}$ of a percent.

Termination

When the iterative process is terminated prematurely for whatever reason, the latest y -value is still feasible for $P(x|y)$. The current lower and upper bounds on $P(x, y)$ provide an indication on how far the latest solution (x, y) is removed from optimality.

*Premature
termination*

22.5 Convergence of Benders' decomposition

For the moment, assume that for every iteration the extreme point u produced by solving the subproblem $S(u|y)$ is unique. Each such point will then result in the addition of a new constraint to the relaxed master problem.

*Uniqueness
assumption
results in ...*

As a result of the uniqueness assumption the iterative process will terminate in a finite number of steps. After all, there are only a finite number of extreme points. In the event that they have all been generated by solving $S(u|y)$ repeatedly, the resulting relaxed master problem $M(y, m)$ becomes equivalent to the full master problem $P_4(y, m)$ and thus the original problem $P(x, y)$.

*... finite
number of steps*

The sequence of relaxed master problems $M(y, m)$ produces a monotone sequence of lower bounds. In the event that after a finite number of steps the relaxed master problem $M(y, m)$ becomes the full master problem $P_4(y, m)$, the set B becomes equal to the set U . The corresponding lower bound is then equal to the original objective function value of $P_4(y, m)$ and thus of the original problem $P(x, y)$. At that moment both the optimal value of $f(y)$ and the optimal u -value of the subproblem make up the original solution of $P_3(u, y)$. As a result, the upper bound is then equal to the optimal objective function value of $P_3(u, y)$ and thus of the original problem $P(x, y)$.

*... converging
bounds*

Based on the uniqueness assumption and the resulting convergence of lower and upper bounds in a finite number of steps, the overall convergence of Benders' decomposition is guaranteed. Termination takes place whenever the lower bound on the objective function value of $P(x, y)$ is equal to its upper bound.

*... overall
convergence*

The entire convergence argument of the Benders' decomposition algorithm so far hinges on the uniqueness assumption, i.e. the assumption that all extreme points u produced by solving the subproblems $S(u|\gamma)$ during the iterative process are unique as long as termination has not been reached. Assume that at some point during the iteration process *prior to termination*, the u -value produced by solving the subproblem is not unique. In this case, the lower bound is still strictly less than the upper bound, but the relaxed master problem will produce the same γ value as in the previous iteration. The Benders' algorithm then cycles from here on and produces the same solution tuple $(\hat{u}, \hat{\gamma})$ each solution. This tuple has the property that the current lower bound LB (obtained from the relaxed master problem) is

$$f(\hat{\gamma}) + m$$

and that the current upper bound UB (with \hat{u} obtained from the subproblem and substituted in the objective function of $P_3(\hat{u}, \hat{\gamma})$) is at least as large as

$$f(\hat{\gamma}) + [b - F(\hat{\gamma})]^T \hat{u}$$

Note that

$$m \geq [b - F(\hat{\gamma})]^T u, \quad \text{for } u \in B$$

by construction. Note also that \hat{u} is already in B due to cycling, which implies that

$$m \geq [b - F(\hat{\gamma})]^T \hat{u}$$

Combining the above leads to

$$\text{LB} = f(\hat{\gamma}) + m \geq f(\hat{\gamma}) + [b - F(\hat{\gamma})]^T \hat{u} \geq \text{UB}$$

which is a contradiction to the fact that prior to termination the lower bound is strictly less than the upper bound. This shows that the uniqueness assumption is true and that Benders' decomposition with feasible subproblems as described in this chapter will always converge.

*Is uniqueness
assumption
true?*

22.6 Formulating dual models

In order to apply the Benders' decomposition scheme, it is necessary to formulate the dual of $P(x|\gamma)$. The rules for this step can be found in books on linear programming. For purposes of completeness and later reference these rules are summarized in this section in the form of typical examples.

This section

If a primal problem is stated as:

Minimize:

$$c_1 x_1 + c_2 x_2$$

*Dual of a
minimization
problem*

Subject to:

$$a_{11}x_1 + a_{12}x_2 \geq b_1$$

$$a_{21}x_1 + a_{22}x_2 = b_2$$

$$a_{31}x_1 + a_{32}x_2 \leq b_3$$

$$x_1 \geq 0, x_2 \geq 0$$

then its dual problem is:

Maximize:

$$u_1b_1 + u_2b_2 + u_3b_3$$

Subject to:

$$a_{11}u_1 + a_{21}u_2 + a_{31}u_3 \leq c_1$$

$$a_{12}u_1 + a_{22}u_2 + a_{32}u_3 \leq c_2$$

$$u_1 \geq 0, u_2 \text{ free}, u_3 \leq 0$$

If a primal problem is stated as:

*Dual of a
maximization
problem*

Maximize:

$$c_1x_1 + c_2x_2$$

Subject to:

$$a_{11}x_1 + a_{12}x_2 \geq b_1$$

$$a_{21}x_1 + a_{22}x_2 = b_2$$

$$a_{31}x_1 + a_{32}x_2 \leq b_3$$

$$x_1 \geq 0, x_2 \geq 0$$

then its dual problem is:

Minimize:

$$u_1b_1 + u_2b_2 + u_3b_3$$

Subject to:

$$a_{11}u_1 + a_{21}u_2 + a_{31}u_3 \geq c_1$$

$$a_{12}u_1 + a_{22}u_2 + a_{32}u_3 \geq c_2$$

$$u_1 \leq 0, u_2 \text{ free}, u_3 \geq 0$$

22.7 Application of Benders' decomposition

Using the decomposition and duality theory of the previous sections, the facility location example can now be divided into a master problem and a dual subproblem.

This section

The original facility location problem can be summarized as follows.

Facility location problem
 $P(x, v, y)$

Minimize:

$$\sum_{cpdz} K_{cpdz} x_{cpdz} + \sum_d \{F_d v_d + R_d \sum_{cz} D_{cz} y_{dz}\}$$

Subject to:

$$\sum_{dz} x_{cpdz} \leq S_{cp} \quad \forall (c, p) \quad (1)$$

$$\sum_p x_{cpdz} = D_{cz} y_{dz} \quad \forall (c, d, z) \quad (2)$$

$$\sum_d y_{dz} = 1 \quad \forall z \quad (3)$$

$$\underline{M}_d v_d \leq \sum_{cz} D_{cz} y_{dz} \leq \overline{M}_d v_d \quad \forall d \quad (4)$$

$$v_d, y_{dz} \in \{0, 1\} \quad (5)$$

$$x_{cpdz} \geq 0 \quad (6)$$

To conduct a Benders' decomposition, it is first necessary to divide the variables and constraints into two groups. The binary variables v_d and y_{dz} , together with the constraints (3), (4) and (5) represent the set Y . The continuous variable x_{cpdz} , together with the constraints (1), (2) and (6) represent the linear part to be dualized. As detailed soon, σ and π are two dual variables introduced for constraints (1) and (2).

How to decompose

In the description of the Benders' decomposition algorithm, the various models are indicated by $P(x, y)$, $M(y, m)$ and $S(u|y)$. The correspondence between the variables used here and those defined for the facility location problem is as follows.

Model notation

- x used previously is equivalent to x used above,
- y used previously is equivalent to v and y used above, and
- u used previously is equivalent to σ and π used above.

As a result, the equivalent model indicators become $P(x, v, y)$, $M(v, y, m)$ and $S(\sigma, \pi|v, y)$, respectively.

The initial master model $M(v, \gamma, m = 0)$ can be stated as follows.

*Initial master
model*
 $M(v, \gamma, m = 0)$

Minimize:

$$\sum_d \{F_d v_d + R_d \sum_{cz} D_{cz} \gamma_{dz}\}$$

Subject to:

$$\sum_d \gamma_{dz} = 1 \quad \forall z \quad (3)$$

$$\underline{M}_d v_d \leq \sum_{cz} D_{cz} \gamma_{dz} \leq \overline{M}_d v_d \quad \forall d \quad (4)$$

$$v_d, \gamma_{dz} \in \{0, 1\} \quad (5)$$

Note that the initial master model does not yet contain any Benders' cuts (i.e. $m = 0$) and that it corresponds to solving:

$$\min_{\gamma \in Y} f(\gamma)$$

previously introduced in the original Benders' decomposition algorithm.

The problem to be dualized, namely the equivalent of the inner optimization problem in $P_1(x, \gamma)$ of Section 22.4, can now be stated as follows.

*Problem to be
dualized*

Minimize:

$$\sum_{cpdz} K_{cpdz} x_{cpdz}$$

Subject to:

$$\sum_{dz} x_{cpdz} \leq S_{cp} \quad \forall (c, p) \mid S_{cp} > 0 \quad (1)$$

$$\sum_p x_{cpdz} = D_{cz} \gamma_{dz} \quad \forall (c, d, z) \mid \gamma_{dz} = 1 \quad (2)$$

$$x_{cpdz} \geq 0 \quad (6)$$

By introducing two dual variables σ_{cp} and π_{cdz} corresponding to the two constraints (1) and (2) respectively, the dual formulation of the problem from the previous paragraph can be written in accordance with the rules mentioned in Section 22.6. Note that the dual variable σ_{cp} is only defined when $S_{cp} > 0$, and the dual variable π_{cdz} is only defined when $\gamma_{dz} = 1$.

*Resulting
subproblem*
 $S(\sigma, \pi \mid v, \gamma)$

Maximize:

$$\sum_{cp} \sigma_{cp} S_{cp} + \sum_{cdz} \pi_{cdz} D_{cz} y_{dz}$$

Subject to:

$$\sigma_{cp} + \pi_{cdz} \leq K_{cpdz} \quad \forall (c, p, d, z)$$

$$\sigma_{cp} \leq 0$$

$$\pi_{cdz} \text{ free}$$

The question arises whether the above subproblem $S(\sigma, \pi | v, y)$ is always feasible for any solution of the initial master problem $M(v, y, m = 0)$. If this is the case, then the Benders' decomposition algorithm described in this chapter is applicable to the original facility location problem. Note that,

Subproblem is always feasible

$$\sum_p S_{cp} \geq \sum_z D_{cz} \quad \forall c$$

is a natural necessary requirement, and that

$$\sum_z D_{cz} \equiv \sum_{dz} D_{cz} y_{dz} \quad \forall c$$

is an identity, because $\sum_d y_{dz} = 1$. These together imply that there is enough supply in the system to meet the demand no matter which distribution center d is used to serve a particular customer zone z .

The Benders' cut to be added each iteration is directly derived from the objective function of the above subproblem $S(\sigma, \pi | v, y)$ evaluated at the original solution (σ, π) . This new constraint is of the form

Benders' cut to be added

$$\sum_{cp} \sigma_{cp} S_{cp} + \sum_{cdz} \pi_{cdz} D_{cz} y_{dz} \leq m$$

where (σ_{cp}, π_{cdz}) are parameters and y_{dz} and m are unknowns.

By adding the Bender's cuts to the initial master problem $M(v, y, m = 0)$, the following regular master problem $M(v, y, m)$ can be obtained after introducing the set B of Benders' cuts generated so far. Note that the optimal dual variables σ and π have been given an extra index $b \in B$ for each Benders' cut.

Resulting master problem $M(v, y, m)$

Minimize:

$$\sum_d \{F_d v_d + R_d \sum_{cz} D_{cz} y_{dz}\} + m$$

Subject to:

$$\sum_d y_{dz} = 1 \quad \forall z$$

$$\underline{M}_d v_d \leq \sum_{cz} D_{cz} y_{dz} \leq \overline{M} v_d \quad \forall d$$

$$\sum_{cp} \sigma_{bcp} S_{cp} + \sum_{cdz} \pi_{bcdz} D_{cz} y_{dz} \leq m \quad \forall b$$

$$v_d, y_{dz} \in \{0, 1\}$$

At this point all relevant components of the original facility location problem to be used inside the Benders' decomposition algorithm have been presented. These components, together with the flowchart in Section 22.4, form all necessary ingredients to implement the decomposition algorithm in AIMMS.

All ingredients available

22.8 Computational considerations

The presentation thus far has mainly focussed on the theory of Benders' decomposition and its application to the particular facility location problem. Implementation issues have barely been considered. This section touches on two of these issues, namely *subproblem splitting* aimed at preserving primary memory, and *use of first-found integer solution* aimed at diminishing computational time. Whether or not these aims are reached, depends strongly on the data associated with each particular model instance.

This section

The dual subproblem can be broken up and solved independently for each commodity c . This gives the advantage that the LP model is divided into $|c|$ smaller models which can all be solved independently. This can reduce memory usage, which is especially true when K_{cpdz} is stored on disk or tape. In this case, it is sufficient to read data into primary memory for only one c at the time. For each fixed commodity \bar{c} the problem then becomes

Splitting subproblem
 $S(\sigma, \pi | v, y)$

Maximize:

$$\sum_p \sigma_{\bar{c}p} S_{\bar{c}p} + \sum_{dz} \pi_{\bar{c}dz} D_{\bar{c}z} \gamma_{dz}$$

Subject to:

$$\sigma_{\bar{c}p} + \pi_{\bar{c}dz} \leq K_{\bar{c}pdz} \quad \forall (p, d, z)$$

$$\sigma_{\bar{c}p} \leq 0$$

$$\pi_{\bar{c}dz} \text{ free}$$

After solving the above problem for each fixed commodity \bar{c} , the objective function value of the overall problem $S(\sigma, \pi | v, \gamma)$, is then the sum over all commodities of the individual objective function values.

Joining solutions

The Benders' cut is derived after finding the optimal integer solution to the master problem. In practice, finding optimal integer solutions can be extremely time consuming as most solution algorithms have difficulty proving that a perceived optimal solution is indeed the optimal solution. Finding a first integer solution is in general easier than finding an optimal integer solution. That is why an alternative implementation of Benders' decomposition can be proposed to take advantage of such a first integer solution.

Use first integer solution

Consider the objective function value of the relaxed master problem for the first integer solution found. This value is not necessarily optimal and therefore cannot be considered as a valid lower bound for the original problem. Nevertheless, it will be treated as such. Now, the Benders' algorithm can terminate prematurely whenever this fake lower bound exceeds the current upper bound. In order to avoid premature termination in the presence of these fake lower bounds, the following constraint should be added:

Extra cut required

$$f(\gamma) + m \leq \text{UB} - \epsilon, \quad \epsilon \geq 0, \text{ small}$$

This constraint makes sure that any fake lower bound resulting from the use of a first integer solution of the master problem cannot be greater than or equal to the current upper bound, and thus will never cause unwanted premature termination.

New Benders' cuts are added every iteration. Their presence together with the above constraint on the fake lower bound will eventually result in an empty integer solution space. This happens when the generated Benders' cuts are such that the true lower bound is greater than or equal to the upper bound minus ϵ . From original Benders' algorithm it follows that convergence has occurred and that the current upper bound provides equals the optimal objective function value. Thus, the alternative approach based on first integer solutions terminates when the modified master problem becomes infeasible and no longer produces a first integer solution.

Convergence with first integer solutions

22.9 A worked example

In this section you will find a small and somewhat artificial example to illustrate the computational results of applying the Benders' decomposition approach to the facility location problem described in this chapter.

This section

In this example there are two production plants, three customer zones, and seven potential sites for distribution centers. Their coordinates are presented in Table 22.1, and are used to determine the transport cost figures as well as the map in Figure 22.3.

Network layout

City	Type	X-coord.	Y-coord.
Arnhem	Production plant	191	444
Rotterdam	Production plant	92	436
Amsterdam	Distribution center	121	488
The Hague	Distribution center	79	454
Utrecht	Distribution center	136	455
Gouda	Distribution center	108	447
Amersfoort	Distribution center	155	464
Zwolle	Distribution center	203	503
Nijmegen	Distribution center	187	427
Maastricht	Customer zone	175	318
Haarlem	Customer zone	103	489
Groningen	Customer zone	233	582

Table 22.1: Considered cities and their coordinates

A total of two commodities are considered. The corresponding supply and demand data for the production plants and the customer zones are provided in Table 22.2, and are specified without units.

Commodities

City	Product A		Product B	
	S_{cp}	D_{cz}	S_{cp}	D_{cz}
Arnhem	18		18	
Rotterdam	15		40	
Maastricht		8		9
Haarlem		9		10
Groningen		7		11

Table 22.2: Supply and demand data

For each distribution center the minimal and maximal throughput data, together with the associated throughput cost figures, are displayed in Table 22.3

Throughput data

d	\underline{M}_d	\overline{M}_d	R_d	F_d
Amsterdam	2	20	5.0	180
The Hague		20	7.0	130
Utrecht		14	3.0	60
Gouda		20	5.5	150
Amersfoort		21	6.0	140
Zwolle		17	7.0	150
Nijmegen		16	3.5	100

Table 22.3: Distribution throughput data

The transport cost values K_{cpdz} are based on distance according to the following formula:

Cost determination

$$K_{cpdz} = \left(\sqrt{(X_p - X_d)^2 + (Y_p - Y_d)^2} + \sqrt{(X_z - X_d)^2 + (Y_z - Y_d)^2} \right) / 100$$

Note that these cost values are the same for both products, and could have been written as K_{pdz} .

In the optimal solution ‘The Hague’, ‘Gouda’ and ‘Amersfoort’ are selected as distribution centers. ‘Haarlem’ is served from ‘The Hague’, ‘Maastricht’ is served from ‘Gouda’, and ‘Groningen’ is served from ‘Amersfoort’. The optimal flows through the network are presented in Table 22.4. The graphical representation of the optimal flows is displayed in Figure 22.3. The corresponding total production and transport costs amount to 828.9408. This optimal solution was obtained with an optimality tolerance of $\varepsilon = 0.0001$ and a total of 15 Benders’ cuts.

Solution

c	p	d	z	x_{cpdz}
product A	Arnhem	Gouda	Maastricht	2
product A	Arnhem	Amersfoort	Groningen	7
product A	Rotterdam	The Hague	Haarlem	9
product A	Rotterdam	Gouda	Maastricht	6
product B	Arnhem	Amersfoort	Groningen	11
product B	Rotterdam	The Hague	Haarlem	10
product B	Rotterdam	Gouda	Maastricht	9

Table 22.4: Optimal flows

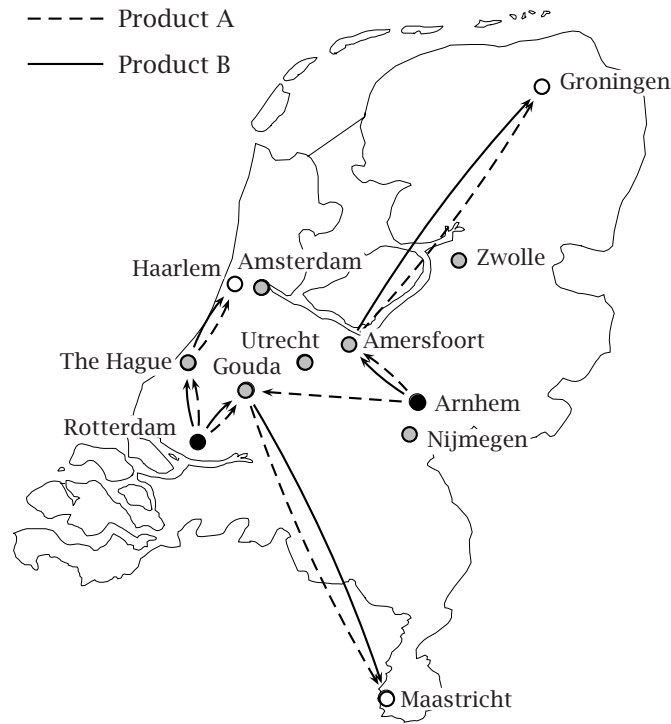


Figure 22.3: Optimal commodity flows

The computational performance of the Benders' decomposition method in terms of solution times is inferior when compared to solving the model as a single mathematical program. Nevertheless, the decomposition method provides a solution approach for extremely large model instances, with the added advantage that a feasible solution is available at any iteration. A premature termination of the algorithm (for instance, when the upper bound remains nearly constant) may very well lead to a good near-optimal solution. This observation applies to the data instance provided in this section.

Final comments

22.10 Summary

In this chapter a facility location problem was translated into a mixed-integer mathematical program. A Benders' decomposition approach was introduced to support the solution of large model instances. The theory underlying the decomposition method with feasible subproblems was first introduced, and subsequently applied to the facility location model. A flowchart illustrating the general Benders' decomposition algorithm was presented as the basis for an implementation in AIMMS. A small data set was provided for computational purposes.

Exercises

- 22.1 Implement the facility location model described in Section 22.2 using the example data presented in Tables 22.1, 22.2 and 22.3.
- 22.2 Implement the same model by using the Benders' decomposition approach described in Section 22.4 and further applied in Section 22.7. Verify whether the solution found with AIMMS is the same as the one found without applying Benders' decomposition.
- 22.3 Implement the Benders' decomposition approach based on using the first integer solution found during the solution of the relaxed master model as described in Section 22.8. In AIMMS you need to set the option `Maximal_number_of_integer_solutions` to 1 in order for the MIP solver to stop after it has found the first feasible integer solution.

Part VI

Appendices

Keyword Table

Chapters	Basic	Intermediate	Advanced
Keywords	An Employee Training Problem A Media Selection Problem A Diet Problem A Farm Planning Problem A Pooling Problem	A Performance Assessment Problem A Two-Level Decision Problem A Bandwidth Allocation Problem A Power System Expansion Problem An Inventory Control Problem	A Portfolio Selection Problem A File Merge Problem A Cutting Stock Problem A Telecommunication Network Problem A Facility Location Problem
Auxiliary Model		■	■ ■ ■
Column Generation			■ ■ ■
Constraint Generation			■
Control-State Variables	■	■ ■	
Customized Algorithm		■	■ ■ ■ ■
Integer Program	■ ■ ■	■	■ ■ ■
Linear Program	■ ■ ■	■ ■ ■	■ ■ ■ ■
Logical Constraint	■		■
Mathematical Derivation		■ ■	■ ■ ■ ■ ■
Mathematical Reformulation		■ ■	■ ■ ■ ■
Measurement Units	■ ■ ■		
Multiple Optima	■		
Multi-Stage		■	
Network Program			■ ■ ■
Nonlinear Program	■	■	
Piece-Wise Approximation			■
Probabilistic Constraint	■		
Quadratic Program			■
Rounding Heuristic	■		
Sensitivity Analysis	■		
Simplex Method			■ ■ ■
Stochastic Program		■ ■	
Two-Stage		■	
What-If Analysis	■	■ ■ ■	
Worked Example	■ ■ ■ ■ ■	■ ■ ■ ■ ■	■ ■ ■ ■ ■

Index

Symbols

λ -formulation, 82

A

a priori reduction, 228
absolute values, 63
abstract model, 2
agricultural model, 11, 113
AIMMS, v
 deployment documentation, xv
 example projects, xv
 help files, xv
 Language Reference, xiv
 Optimization Modeling, xiv
 tutorials, xv
 User's Guide, xiii
algebraic, 29
 notation, 32
approximation
 piecewise linear, 81, 216
arc, 55, 246
assignment problem, 59
authors
 Bisschop, J.J., xviii

B

balance
 constraint, 88
 tax-subsidy, 147
bandwidth allocation problem, 158
base unit, 109
basic variable, 229
basis
 constant, 49
Benders' cut, 271
Benders' decomposition, 262
 convergence, 266
 flowchart, 265
 termination, 266
binary variable, 26
binding constraint, 20
bisection phase, 150
bisection search, 149
blending
 model, 123
 proportional, 126

bottleneck identification model, 246
branch and bound, 26, 261

C

candidate variable, 231
chance constraint, 71
Chebyshev inequality, 201
column
 entry condition, 251
 generation, 239, 250
concavity, 83
concrete model, 2
conditional probability, 182
constant basis, 49
constant objective, 48
constraint
 balance, 55, 88
 binding, 20
 chance, 71
 conditional, 79
 covering, 98
 either-or, 77
 flow, 128
 logical, 99, 215
 non-binding, 20
 nonnegativity, 16
 probabilistic, 71, 74, 93
 range, 68, 107
 redundant, 262
 shadow price, 42
 transportation, 224
contour, 18, 29
control variable, 88, 169, 182
convex combination, 118
convexity, 83, 203
 strict, 147
corner solution, 20
correlated data, 140
correlation
 perfect, 205
covariance, 202
cover, *see* set, covering
critical value, 72
cumulative distribution, 72
curve fitting, 64
 least absolute values, 65
 least maximum deviation, 65

least-squares, 65
cutting stock problem, 235

D

data
 correlated, 140
 scaling, 142
data envelopment analysis, 134
data initialization, 39
DEA, *see* data envelopment analysis
decision problem
 picturing, 17
decision variable, 15
decomposition, 262
degeneracy, 44
delayed
 column generation, 239
 pattern generation, 238
derived unit, 109
deterministic model, 168
diet problem, 105
directed network, 52
discontinuous variable, 75
distribution
 cumulative, 72
documentation
 deployment features, xv
domain restriction, 129
downside variance, 213
dual model
 overview, 267
dual of maximization problem, 268
dual of minimization problem, 267
dynamic column generation, 250

E

efficiency
 absolute, 135
 relative, 135
either-or constraint, 77
element parameter, 136, 186, 231
employee training problem, 87
equilibrium model, 27
event
 parameter, 168, 181
 probability, 182
example projects, xv
expectation, 200
expected cost, 177
explicit formulation, 32
extreme point, 263

F

facility location problem, 258
farm planning problem, 113

feasible region, 18, 25, 29
 picturing, 18
file merge problem, 221
fixed cost, 76
formulation
 λ , 82
 AIMMS, 37
 dual overview, 267
 explicit, 32
 piecewise linear, 81, 216
 symbolic, 33
 symbolic-indexed, 34
fractional objective, 67
full master problem, 264

G

general pure network flow model, 59
generalized network problem, 60
global optimum, 30, 203, 205, 215

H

horizon, 90
hunt phase, 149

I

index notation, 35
indicator variable, 76
inequality
 linear, 13
infeasibility, 22, 27
initialization
 data, 39
integer programming, 26
inventory control problem, 181
inverse cumulative distribution, 72
investment model
 strategic, 198
 tactical, 208
IP, *see* integer programming

L

lag and lead notation, 89
Largest In Least Empty, 237
LILE, *see* Largest In Least Empty
linear
 equation, 13
 inequality, 13
 piecewise, 81, 216
 program, 63
 partitioned, 229
 programming, 13
local optimum, 30
logical constraint, 99, 215
 conditional, 79

either-or, 77
LP, *see* linear, programming

M

marginal values, 41
master problem, 264
mathematical
 algorithm, 5
 language, 5
 model, 3
 programming, 13
 theory, 5
maximum flow problem, 60
measurement units, 108
 base unit, 109
 derived unit, 109
 quantities, 108
 SI unit, 109
 unit consistency, 109
 unit convention, 111
 unit conversion, 110
 unit-valued parameter, 110
media selection problem, 96
minimax objective, 66
MIP, *see* mixed-integer programming
mixed integer programming, 26
model, 2
 abstract, 2
 agricultural, 11, 113
 assignment, 59
 blending, 123
 bottleneck identification, 246
 concrete, 2
 deterministic, 168
 dual overview, 267
 equilibrium, 27
 general pure network flow, 59
 generalized network, 60
 linear, 63
 mathematical, 3
 maximum flow, 60
 multi-commodity network, 61
 multi-period, 87
 network, 27
 network flow, 52
 nonlinear, 29
 optimization, 3
 path-finding, 251
 personnel planning, 10, 87
 portfolio, 9
 refinery, 7, 123
 relaxed, 26, 92
 set covering, 102
 set packing, 103
 set partitioning, 103
 shortest path, 60
 strategic investment, 198

tactical investment, 208
transportation, 59
transshipment, 59
validation, 7
waste water, 144

Model Explorer, 37
modeling process, 6
multi-commodity, 258
multi-commodity network problem, 61
multi-period model, 87
multi-stage, 169, 182

N

network, 52
 approach, 224
 arc, 55
 assignment problem, 59
 design, 246
 directed, 52
 flow model, 52
 general pure network flow, 59
 generalized network problem, 60
 maximum flow problem, 60
 multi-commodity network problem, 61
 node, 55
 shortest path problem, 60
 telecommunication problem, 245
 transportation problem, 59
 transshipment problem, 59
network model, 27
NLP, *see* nonlinear, programming
node, 55, 246
non binding constraint, 20
nonbasic variable, 229
nonlinear
 model, 29
 programming, 27

O

objective
 coefficient range, 50
 concave, 83
 constant, 48
 convex, 83
 fractional, 67
 minimax, 66
one-sided variance, 213
optimal solution, 20, 21
optimization
 model, 3
optimum
 global, 30, 203, 205, 215
 local, 30
 multiple, 21

P

Paragon Decision Technology B.V., v
 parameter
 element, 136, 186, 231
 event, 168, 181
 unit-valued, 110
 partitioned linear program, 229
 path, 246
 path-finding model, 251
 pattern generation, 238
 performance assessment problem, 134
 performance measure, 135
 personnel planning model, 10, 87
 piecewise linear formulation, 81, 216
 planning interval, 90
 policy making, 145
 policy receiving, 145
 pooling problem, 123
 portfolio, 197
 diversification, 197
 model, 9
 problem, 195
 risk, 197
 positive semi-definite, 211
 power system expansion problem, 168
 priority, 91
 probabilistic constraint, 71, 74, 93
 probability
 conditional, 182
 event, 182
 recursive, 189
 unconditional, 183
 problem
 bandwidth allocation, 158
 cutting stock, 235
 diet, 105
 employee training, 87
 facility location, 258
 farm planning, 113
 file merge, 221
 inventory control, 181
 media selection, 96
 performance assessment, 134
 pooling, 123
 portfolio, 195
 power system expansion, 168
 telecommunication network, 245
 two-level decision, 144
 profit function
 contour, 18
 picturing, 18
 programming, 13
 integer, 26
 linear, 13, 63
 mathematical, 13
 mixed integer, 26
 multi-stage, 169, 182

 stochastic, 169, 176
 two-level, 144
 two-stage, 169, 182
 zero-one, 26

proportional blending, 126

Q

quantities, 108

R

random variable, 71, 200
 range constraint, 68, 107
 recursive probability, 189
 reduced cost, 41, 45, 230
 reduction
 a priori, 228
 redundant constraint, 262
 reference set, 138
 refinery model, 7, 123
 relaxed
 master problem, 264
 model, 26, 92
 right-hand-side
 range, 50
 risk, 197
 rounding, 24, 92

S

scaling, 142
 scenario, 169, 183, 209
 selection, 170
 security, 196
 sensitivity analysis, 22, 41, 121
 sensitivity range, 42
 constant basis, 49
 constant objective, 48
 separable function, 81
 separation
 model and data, 34
 symbols and values, 33
 set
 covering, 102
 packing, 103
 partitioning, 103
 reference, 138
 shadow price, 42, 121, 230
 binding constraint, 43
 degeneracy, 44
 equality constraint, 42
 limitation, 44
 nonbinding constraint, 43
 picturing, 44
 range, 49
 shortest path problem, 60
 SI unit, 109

simplex
 iteration, 230
 method, 21, 229
 solution
 corner, 20
 optimal, 20, 21
 picturing, 18, 25, 29
 rounding, 24, 92
 suboptimal, 91
 solver failure
 derivative error, 131, 151
 division error, 150
 SOS, *see* special ordered set
 special ordered set, 80
 SOS1, 80
 SOS2, 81
 state variable, 88, 169, 182
 stochastic programming, 169, 176
 multi-stage, 169, 182
 two-stage, 169, 182
 stochastic terminology, 184
 strategic investment model, 198
 symbolic
 formulation, 33

T

tactical investment model, 208
 tax & subsidy, 145
 telecommunication network problem, 245
 time
 beyond, 90
 horizon, 90
 past, 90
 planning interval, 90
 step, 209
 tolerance
 absolute optimality, 91
 optimality, 26
 relative optimality, 91
 transportation problem, 59
 transshipment problem, 59
 tree-based terminology, 184
 tutorials, xv
 two-level
 different knowledge levels, 145
 optimality conditions, 156
 two-level decision problem, 144
 two-stage, 169, 182

U

unboundedness, 22, 27
 uncertain demand, 170
 unconditional probability, 183
 unimodular, 27
 unit
 base, 109

consistency, 109
 convention, 111
 conversion, 110
 derived, 109
 SI, 109
 unit-valued parameter, 110
 units, *see* measurement units

V

variable, 14
 adjusting objective coefficient, 46
 basic, 229
 binary, 26
 bound, 107
 bound relaxation, 46
 candidate, 231
 control, 88, 169, 182
 decision, 15
 discontinuous, 75
 eliminating product, 84
 indicator, 76
 name, 15
 nonbasic, 229
 priority, 91
 random, 71, 200
 reduced cost, 41, 45
 shadow price range, 49
 state, 88, 169, 182
 status, 129
 variable range, 49
 variance, 200
 downside, 213
 one-sided, 213

W

waste water model, 144
 what-if analysis, 169, 174

Z

zero-one programming, 26

Bibliography

- [An91] R. Anbil, E. Gelman, B. Patty, and R. Tanga, *Recent advances in crew-pairing optimization at american airlines*, Interfaces **21** (1991), no. 1, 62–74. [1.5.4](#)
- [Ba66] F.M. Bass and R.T. Lonsdale, *An exploration of linear programming in media selection*, Journal of Marketing Research **3** (1966), 179–188. [9](#)
- [Ba79] M.S. Bazaraa and C.M. Shetty, *Nonlinear programming: Theory and algorithms*, John Wiley & Sons, New York, 1979. [2.3](#)
- [Be69] E.M.L. Beale and J.A. Tomlin, *Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables*, Proceedings of the 5th International Conference on Operations Research (Tavistock, London) (J. Lawrence, ed.), 1969. [7.5](#)
- [Bi82] J.J. Bisschop, W. Candler, J.H. Duloy, and G.T. O'Mara, *The indus basin model: A special application of two-level linear programming*, Mathematical Programming Study **20** (1982), 30–38. [14](#)
- [Bi99] J.J. Bisschop and M.R. Roelofs, *AIMMS, the language reference*, 1999. [10.3](#)
- [Bo93] R.A. Bosch, *Big mac attack*, OR/MS Today (1993). [10](#)
- [Bu89] G. Buxey, *Production scheduling: Practice and theory*, European Journal of Operational Research **39** (1989), 17–31. [1.5.2](#)
- [Ch55] A. Charnes and W.W. Cooper, *Generalizations of the warehousing model*, Operations Research Quarterly **6** (1955), 131–172. [1.5.2](#)
- [Ch59] A. Charnes and W.W. Cooper, *Change-constrained programming*, Management Science **6** (1959), 73–80. [6.6](#)
- [Ch62] A. Charnes and W.W. Cooper, *Programming with linear fractional functional*, Naval Research Logistics Quarterly **9** (1962), 181–186. [6.3](#)
- [Ch68a] A. Charnes, W.W. Cooper, J.K. DeVoe, D.B. Learner, and W. Reinecke, *A goal programming model for media planning*, Management Science **14** (1968), B431–B436. [9](#)
- [Ch68b] A. Charnes, W.W. Cooper, and R.J. Niehaus, *A goal programming model for manpower planning*, Management Science in Planning and Control (J. Blood, ed.), Technical Association of the Pulp and Paper Industry, New York, 1968. [1.5.4](#)
- [Ch78] A. Charnes, W.W. Cooper, and E. Rhodes, *Measuring the efficiency of decision making units*, European Journal of Operational Research **2** (1978), 429–444. [13](#)
- [Ch83] V. Chvátal, *Linear programming*, W.H. Freeman and Company, New

- York, 1983. 1.5.2, 1.5.4, 2.1.3, 4, 8, 10, 19, 20, 21
- [Ch96] A. Charnes, W.W. Cooper, A.Y. Lewin, and L.M. Seiford (eds.), *Data envelopment analysis: Theory, methodology and applications*, 2nd ed., ch. 4, Computational Aspects of DEA, pp. 63–88, Kluwer Academic Publishers, 1996. 13.4
- [Da63] G.B. Dantzig, *Linear programming and extensions*, Princeton University Press, Princeton, N.J., 1963. 2.1.3
- [Ep87] D.D. Eppen, F.J. Gould, and C.P. Schmidt, *Introductory management science*, 2nd ed., Prentice-Hall, Englewood Cliffs, N.J., 1987. 4, 8
- [Er94] E. Erkut, *Big mac attack revisited*, OR/MS Today (1994). 10
- [Fl89] C.A. Floudas, A. Aggarwal, and A.R. Ciric, *Global optimum search for nonconvex NLP and MINLP problems*, Computers & Chemical Engineering **13** (1989). 12
- [Ga72] R.S. Garfinkel and G.L. Nemhauser, *Integer programming*, John Wiley & Sons, New York, 1972. 2.2
- [Ge74] A.M. Geoffrion and G.W. Graves, *Multicommodity distribution system design by benders decomposition*, Management Science **20** (1974), no. 5, 822–844. 22
- [Gi61] P.C. Gilmore and R.E. Gomory, *A linear programming approach to the cutting stock problem part i*, Operations Research **9** (1961), 849–859. 20
- [Gi63] P.C. Gilmore and R.E. Gomory, *A linear programming approach to the cutting stock problem part ii*, Operations Research **11** (1963), 863–888. 20
- [Gl92] F. Glover, D. Klingman, and N.V. Phillips, *Network models in optimization and their applications in practice*, John Wiley & Sons, New York, 1992. 5, 19, 19.1, 19.1, 19.2
- [Go77] B. Golden and T. Magnanti, *Deterministic network optimizations: A bibliography*, Networks **7** (1977), 149–183. 5
- [Ha60] F. Hanssmann and S.W. Hess, *A linear programming approach to production and employment scheduling*, Management Technology **1** (1960), 46–52. 1.5.2
- [Ha78] C.A. Haverly, *Studies of the behaviour of recursion for the pooling problem*, ACM SIGMAP Bulletin **26** (1978). 12
- [Ha86] P.B.R. Hazell and R.D. Norton, *Mathematical programming for economic analysis in agriculture*, Macmillan, New York, 1986. 11
- [Hu69] T.C. Hu, *Integer programming and network flows*, Addison-Wesley, Reading, Massachusetts, 1969. 21
- [In94] G. Infanger, *Planning under uncertainty*, Boyd & Fraser, Danvers, Massachusetts, 1994. 16, 17
- [Ka94] P. Kall and S.W. Wallace, *Stochastic programming*, John Wiley & Sons, Chichester, 1994. 16, 17
- [Ke80] J. Kennington and R. Helgason, *Algorithms for network programming*, John Wiley & Sons, New York, 1980. 5
- [Ke81] D. Kendrick, A. Meeraus, and J.S. Suh, *Oil refinery modeling with the gams language*, Tech. Report Research Report Number 14, Center for

- Energy Studies, The University of Texas, Austin, 1981. 1.5.1
- [Ko87] J.S.H. Kornbluth and G.R. Salkin, *The management of corporate financial assets: Applications of mathematical programming models*, Academic Press Ltd., London, 1987. 1.5.3, 4
- [Ku88] G. Kutcher, A. Meeraus, and G.T. O'Mara, *Modeling for agricultural policy and project analysis*, Tech. report, The World Bank, Washington D.C., 1988. 1.5.5, 11
- [Lu73] D.G. Luenberger, *Introduction to linear and nonlinear programming*, Addison-Wesley, Reading, Massachusetts, 1973. 2.1.3
- [Ma52] H.M. Markowitz, *Portfolio selection*, Journal of Finance 7 (1952), 77–91. 1.5.3, 18, 18.1, 18.2
- [Ma56] A.S. Manne, *Scheduling of petroleum refinery operations*, Harvard Economic Studies, vol. 48, Harvard University Press, Cambridge, Massachusetts, 1956. 1.5.1
- [Ma76] A.S. Manne, *Eta: A model for energy technology assessment*, Bell Journal of Economics 7 (1976), 379–406. 1.5.1
- [Ma92] S.A. Malcolm and S.A. Zenios, *Robust optimization for power systems capacity expansion under uncertainty*, Tech. Report Report 92-03-07, Decision Sciences Department, The Wharton School, University of Pennsylvania, Philadelphia, 1992. 16
- [Me98] E.A. Medova, *Chance-constrained stochastic programming for integrated services network management*, Annals of Operations Research 81 (1998). 21
- [Ne88] G.L. Nemhauser and L.A. Wolsey, *Integer and combinatorial optimization*, John Wiley & Sons, New York, 1988. 2.2
- [No91] M. Norman and B. Stoker, *Data envelopment analysis: The assessment of performance*, John Wiley & Sons, Chichester, 1991. 13
- [Or93] J.B. Orlin, R.K. Ahuja, and T.L. Magnanti, *Network flows, theory, algorithms and applications*, Prentice Hall, New Jersey, 1993. 5
- [Re89] F.K. Reilly, *Investment analysis and portfolio management*, The Dryden Press, Orlando, Florida, 1989. 18
- [Sc91] L. Schrage, *Lindo: An optimization modeling system*, 4th ed., The Scientific Press, South San Francisco, 1991. 1.5.4, 3
- [Th92] G.L. Thompson and S. Thore, *Computational economics: Economic modeling with optimization software*, The Scientific Press, South San Francisco, 1992. 1.5.4
- [Va84] H.R. Varian, *Microeconomic analysis*, 2nd ed., W.W. Norton & Company, New York, 1984. 2.3
- [Wa75] H.M. Wagner, *Principles of operations research*, 2nd ed., Prentice-Hall, Englewood Cliffs, N.J., 1975. 8, 10
- [We67] *Webster's seventh new collegiate dictionary*, G. & C. Merriam Company, Springfield, Massachusetts, 1967. 3.3
- [Wi90] H.P. Williams, *Model building in mathematical programming*, 3rd ed., John Wiley & Sons, Chichester, 1990. 3, 6, 7, 7.6